RECALL OR RECHECK? EVALUATING MEMORY-AUGMENTED AGENTS IN EVOLVING MOBILE ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory allows agents to retain past experience as their environments continue to change. Historical information may disappear from the current state, while recent updates may be absent from memory. We introduce R^2-MobileMemory, an executable benchmark that separates persistent memory from the query-time state of the same evolving mobile world to study when agents should recall, recheck, or combine both sources. Our evaluation shows that memory provides its largest gains when task-critical historical evidence is no longer available in the environment, whereas it adds little when current-state search is already sufficient. Extending memory with later observations does not consistently improve task success. The added information can overlap with evidence already available in current applications or introduce competing versions of relevant facts. Even after retrieving a complete annotated support set, agents can still fail to select the relevant state or realize the requested outcome. These results frame memory-augmented interaction as a problem of temporal evidence arbitration, where success depends not only on what memory preserves, but also on selecting and using the right evidence relative to an evolving environment.

1 INTRODUCTION

As large language models (LLMs) evolve from one-shot assistants into persistent agents, their decisions increasingly depend on information accumulated across prior interactions. Long-term memory provides this continuity by preserving user-specific facts, prior decisions, and relevant experience beyond the current context (Zhong et al., 2024; Packer et al., 2023; Shinn et al., 2023; Zhao et al., 2024; Hu et al., 2025). However, memory is formed from past observations, while the world in which an agent acts may continue to change. As a result, what an agent remembers and what the current environment reveals need not coincide.

Existing evaluations have progressively expanded from testing whether agents can recall information from long interaction histories to whether retained experience can support reasoning, tool use, and multi-step behavior (Maharana et al., 2024; Wu et al., 2024; Tavakoli et al., 2026; Shen et al., 2026; He et al., 2026). Yet these evaluations largely treat accumulated memory or history as the primary source of information for downstream decisions (Chai et al., 2026; Bei et al., 2026; He et al., 2026; Liu et al., 2026a). In an evolving environment, relevant historical information may disappear from the current state, while newer observations may supersede what memory retains. Agents must therefore determine when memory provides sufficient evidence and when the environment needs to be consulted.

Mobile environments make this challenge particularly salient. Smartphones are continuously evolving personal information spaces, where messages arrive, plans are revised, files are replaced, orders progress, and records may eventually disappear. At the same time, task-relevant evidence is fragmented across applications, modalities, and time. Recent work has advanced long-horizon personal memory and personalized mobile assistance (Deng et al., 2026a; Chen et al., 2026; Nie et al., 2026), while increasingly realistic mobile benchmarks support cross-session interaction, GUI execution, and tool-augmented workflows (Rawles et al., 2025; Kong et al., 2026; Zhu et al., 2026; Liu et al., 2026b). Yet these settings rarely treat persistent memory and the query-time environment as dis-

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.

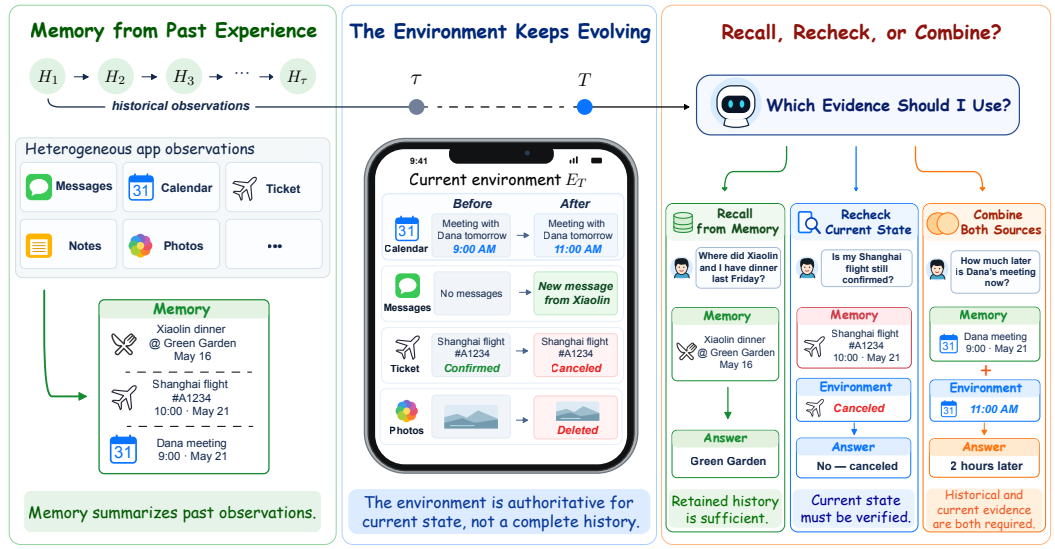

Figure 1: **Recall or recheck in evolving mobile environments.** Long-term memory is formed from historical observations, while the phone continues to evolve: new information may appear and old records may disappear. At query time, an agent must recall retained experience, recheck current application state, or combine both to obtain sufficient evidence to complete the executable task.

Table 1: Comparison of representative long-term memory and agent benchmarks. **Temporal Asynchrony** indicates that persistent memory and the query-time environment correspond to different temporal snapshots.

Benchmark	Data Setting			Agent Setting			Temporal Asynchrony
	Long-Horizon History	Personalization	Multimodal	Persistent Memory	Environment Access	Action Execution	
Long-term memory benchmarks							
LoCoMo (Maharana et al., 2024)	✓	✗	✓	✓	✗	✗	✗
LongMemEval (Wu et al., 2024)	✓	✗	✗	✓	✗	✗	✗
BEAM (Tavakoli et al., 2026)	✓	✗	✗	✓	✗	✗	✗
Agent memory benchmarks							
Mem2ActBench (Shen et al., 2026)	✓	✓	✗	✓	✗	✗	✗
MemoryArena (He et al., 2026)	✓	✗	✗	✓	✓	✓	✗
WorldMemArena (Liu et al., 2026a)	✓	✗	✓	✓	✗	✗	✗
Mobile memory and agent benchmarks							
MobileMem (Deng et al., 2026a)	✓	✓	✓	✓	✗	✗	✗
MemGUI-Bench (Liu et al., 2026b)	✓	✗	✓	✓	✓	✓	✗
KnowU-Bench (Chen et al., 2026)	✓	✓	✓	✓	✓	✓	✗
R^2-MobileMemory	✓	✓	✓	✓	✓	✓	✓

tinct evidence sources corresponding to different temporal views of the same world. The current environment provides the application state at query time, while memory may preserve earlier information that is no longer available there. Consequently, a nominal memory task may be solved by repeatedly searching the current device, while relying on memory alone may lead an agent to use outdated information. As illustrated in Figure 1, an agent must therefore determine when to recall from memory, when to recheck the current environment, and when to combine both before acting.

To study this problem, we introduce **R^2-MobileMemory**, an executable benchmark that separates persistent memory M_τ from the query-time environment E_T, while grounding both in the same evolving mobile world. Memory is formed from observations available up to time τ, whereas the environment continues to evolve until the evaluation time T. Tasks are grounded so that successful completion may require historical evidence retained in memory, current evidence available in the environment, or information from both sources. Agents retrieve evidence and interact with the current phone to complete tasks. Success is evaluated from the resulting application state. We construct

these worlds with R^2**-Forge**, which realizes longitudinal user histories as executable application records and grounds tasks in the realized evidence. Table 1 compares this setting with representative long-term memory and mobile-agent benchmarks.

Our contributions are threefold:

- We introduce R^2**-MobileMemory**, an executable benchmark for studying memory-augmented agents in evolving mobile environments. It explicitly separates persistent memory M_τ from the query-time environment E_T, enabling controlled evaluation of when an agent should rely on historical memory, current state, or both.
- We develop R^2**-Forge**, a state-grounded construction pipeline that realizes longitudinal user histories as evolving application records and grounds executable tasks in the resulting evidence. This design links memory formation, query-time environment state, and terminal-state evaluation within the same mobile world.
- Our experiments show that effective memory augmentation depends on more than retaining information. Memory provides its largest gains when task-critical historical evidence is no longer available in the current environment, while incorporating more recent observations does not consistently improve success. Trajectory-level analysis further reveals distinct failures in retaining, retrieving, selecting, and using the evidence required for execution.

2 Memory-Augmented Interaction in Evolving Mobile Environments

Let Q denote the space of user tasks, and let $E_t = (S_t, \mathcal{A})$ denote the mobile environment at time t, where S_t is the observable application state and $\mathcal{A}$ is the set of executable GUI actions. Unlike a static environment, S_t evolves as application records are created, updated, or removed.

Let M_τ denote the persistent memory formed from observations available up to time τ. At query time $T > \tau$, the agent interacts with E_T. Memory M_τ may preserve historical evidence no longer exposed in E_T. Later updates in E_T may be absent from M_τ.

Definition 2.1 (Memory-Augmented Mobile Agent Task) *Given a user task $q \in Q$, persistent memory M_τ, and the query-time environment E_T, a memory-augmented mobile agent follows a policy π to produce an interaction trajectory*

$$\xi = \pi(q, M_\tau, E_T).$$

At each step, the agent may invoke memory search over M_τ, search application records in E_T, or interact with E_T through the GUI. The objective is to acquire sufficient evidence and use it to reach a terminal environment state satisfying the task specification.

3 R^2-Forge: State-Grounded World-to-Task Construction

Operationalizing Definition 2.1 requires longitudinal mobile histories in which user behavior, application state, and multimodal evidence evolve over time. Collecting such histories at scale is costly and privacy-sensitive, while publicly available data provide limited coverage of sustained user-specific behavior (Deng et al., 2026a; Nie et al., 2026; Zhang et al., 2026). Unconstrained synthesis, however, can deviate from realistic user behavior (Xiao et al., 2026; Deng et al., 2026a). We therefore use controlled synthesis grounded in high-level behavioral priors.

As shown in Figure 2, R^2-Forge constructs each world in three stages. *World Planning* expands a user prior into a temporally ordered session history. *State Realization* instantiates these sessions as executable application records. *Task Grounding* derives evidence-backed tasks from the realized world. For the resulting application-record history R, a memory boundary τ and query time T define two temporal views of the same world:

$$M_\tau = \mathrm{Ingest}(R_{\leq\tau}), \qquad E_T = \mathrm{Project}_T(R).$$

Review and revision are applied throughout construction. Final tasks are organized by Memory Capability and Evidence Condition. Further details of benchmark construction and validation are provided in Appendix A.

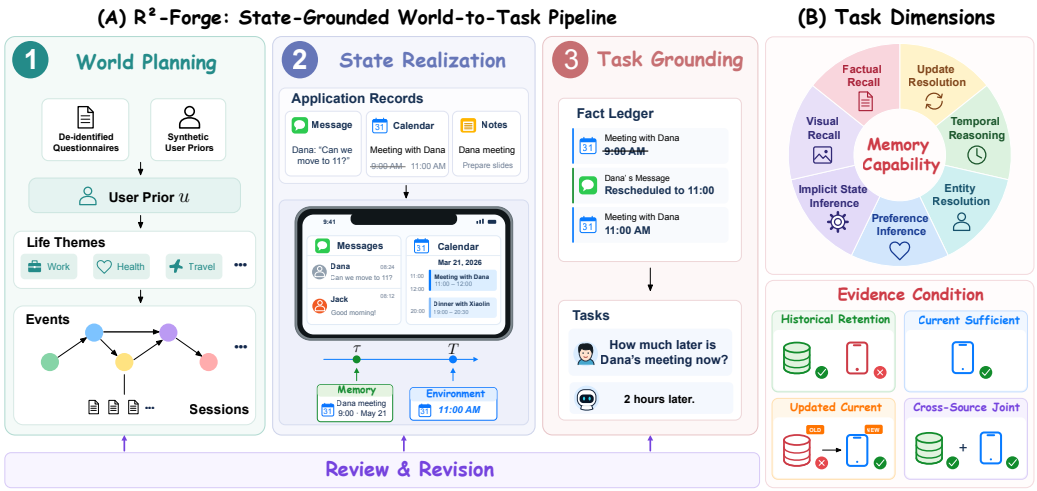

Figure 2: Overview of R^2-Forge and task dimensions. (A) R^2-Forge plans a longitudinal user world, realizes it as executable application state, and grounds tasks in the realized evidence. Persistent memory M_τ and the query-time environment E_T are derived from different temporal views of the same realized world, with review and revision applied throughout construction. (B) Tasks are organized by Memory Capability and Evidence Condition.

3.1 Prior-Grounded World Planning

Each world begins from a structured user prior u. Seven worlds are initialized from de-identified and fictionalized questionnaire priors describing smartphone habits and life patterns, while three use fully synthetic priors. All subsequent identities, events, application records, and media are generated within the simulated world.

R^2-Forge expands each prior into recurring life themes, concrete events, and temporally bounded sessions, producing an ordered session history

$$H = (h_1, \ldots, h_n).$$

Dependencies among events constrain their temporal ordering so that later activities can build on context established earlier. Once the timeline is fixed, we select a complete-session memory boundary τ and a world-level query time $T > \tau$. Sessions up to τ provide the observations later used for memory formation, while subsequent sessions continue to evolve the world before T. All tasks within the same world share this query time.

3.2 Application State Realization

Given the planned session history H, R^2-Forge realizes each session as application-specific records under the guidance of Fact Intents and Task Blueprints. Fact Intents specify candidate facts and state transitions to be expressed in the world, while Task Blueprints reserve the capability and evidence coverage needed by downstream tasks. Final task evidence and target outcomes are determined during task grounding.

Sessions are instantiated as heterogeneous application records, including messages, calendar events, notes, contacts, files, orders, and media. Records referring to the same people, events, or application objects are kept consistent across sessions. Application state also persists across sessions, allowing later activities to update objects created earlier. Together, these records form the realized multimodal history R.

At query time, R is projected according to application-specific state semantics to obtain E_T. Earlier revisions may therefore be absent from E_T. Memory M_τ, formed from $R_{\leq\tau}$, may preserve this historical information.

3.3 Task Grounding and Evidence Design

After state realization, R^2-Forge extracts atomic facts from the realized record history R into a Fact Ledger F. Each fact is linked to its source records and temporal context, allowing versions and relationships to be traced within the realized world. Task Blueprints are then resolved against F to bind the required evidence and target outcome of each task. Coverage is planned before realization. Ground truth is then bound to the realized records, excluding unsupported planned content. An LLM finally renders each grounded task specification as a natural user request.

We characterize each task along two orthogonal dimensions. *Memory Capability* describes the operation required over user-specific evidence: Factual Recall, Update Resolution, Temporal Reasoning, Entity Resolution, Preference Inference, Implicit State Inference, or Visual Recall. *Evidence Condition* describes how the task-critical evidence is distributed between persistent memory and the query-time environment. Historical Retention requires historical evidence that is no longer recoverable from E_T. Current Sufficient can be resolved entirely from E_T. Updated Current requires a post-τ update that supersedes earlier information while leaving the current state sufficient. Cross-Source Joint requires both historical evidence unavailable in E_T and current evidence introduced after τ. The four Evidence Conditions are mutually exclusive. Panel (B) of Figure 2 summarizes the two dimensions, with representative cases provided in Appendix B.3.

3.4 Review and Executable Validation

Review and revision are interleaved throughout R^2-Forge. Final tasks undergo independent source-level checks of target correctness, evidence sufficiency, potential leakage, and alternative support in the query-time environment. Evidence dependencies are further examined using the designated evidence, the absence of designated evidence, and task-specific evidence removals. Detailed validation procedures and coverage are provided in Appendix A.4.

Every task is executed in MobileGym (Wu et al., 2026) and evaluated from its terminal state. A task-specific rubric specifies the required outcome, permitted changes, and prohibited side effects. Task success requires the terminal state to satisfy the complete rubric. Figure 5 illustrates an execution from evidence retrieval to the resulting phone state.

Benchmark statistics. R^2-MobileMemory contains 453 executable tasks across ten user worlds, including five Chinese and five English worlds. The worlds span 462–547 days of simulated history, and the tasks cover 23 user applications, all seven Memory Capabilities, and all four Evidence Conditions. Detailed world composition, temporal coverage, and task distributions are reported in Appendix B.

4 Experiments

4.1 Experimental Setup

We evaluate three mobile agents: UI-Venus-2-9B (Venus Team et al., 2026), Gemini-3.7-Flash (Google DeepMind, 2026), and Qwen3.8-Max-0902 (Qwen Team, 2026a). Each agent is paired with eight memory backends, including the dense-retrieval baseline NaiveRAG (Lewis et al., 2020). The other backends are HippoRAG2 (Gutiérrez et al., 2025), A-MEM (Xu et al., 2025), MemOS (Li et al., 2025), EverMemOS (Hu et al., 2026a), Mem0 and Mem0^g (Chhikara et al., 2025), and LangMem (LangChain Team, 2025). All systems are evaluated on the same 453 tasks and query-time environments under a shared interaction protocol. Memory backends ingest the same ordered application-record stream, and persistent memory remains read-only during task execution. All agents use the same MobileGym GUI tools and are limited to at most 50 interaction steps per task.

We compare four evidence-access conditions: *GUI Only*, *GUI + App Search*, $M_\tau + E_T$, and $M_T + E_T$, where $M_T = \mathrm{Ingest}(R_{\leq T})$ denotes memory formed from the complete record history through query time. GUI + App Search adds structured retrieval over the query-time environment to GUI interaction. $M_\tau + E_T$ further provides historical memory formed at the boundary τ. $M_T + E_T$ extends memory formation through T. Both memory conditions retain GUI interaction and appli-

Table 2: Task Success Rate (%) with $M_\tau + E_T$, overall and by Memory Capability. Bold marks the best result for each agent, including ties.

Memory Backend	Overall	Factual Recall	Update Resolution	Temporal Reasoning	Entity Resolution	Preference Inference	Implicit State Inference	Visual Recall
UI-Venus-2-9B								
NaiveRAG	**20.5**	**24.4**	7.8	7.7	**26.3**	**50.0**	**31.1**	14.5
HippoRAG2	19.6	19.5	12.2	**11.5**	19.3	37.0	**31.1**	**20.0**
A-MEM	20.1	18.3	12.2	6.4	22.8	**50.0**	**31.1**	18.2
MemOS	12.4	9.8	4.4	1.3	21.1	37.0	20.0	9.1
EverMemOS	20.3	17.1	**14.4**	10.3	22.8	45.7	26.7	**20.0**
Mem0	14.3	13.4	11.1	7.7	14.0	30.4	28.9	5.5
Mem0^g	14.6	6.1	11.1	6.4	19.3	37.0	**31.1**	7.3
LangMem	15.2	11.0	8.9	3.8	17.5	43.5	26.7	12.7
Gemini-3.7-Flash								
NaiveRAG	60.3	57.3	**53.3**	48.7	**61.4**	76.1	64.4	**74.5**
HippoRAG2	58.7	58.5	50.0	47.4	59.6	73.9	60.0	**74.5**
A-MEM	**60.5**	**61.0**	**53.3**	47.4	59.6	78.3	68.9	69.1
MemOS	44.8	43.9	35.6	24.4	49.1	67.4	60.0	54.5
EverMemOS	57.6	46.3	47.8	**53.8**	59.6	**82.6**	64.4	67.3
Mem0	37.7	32.9	27.8	25.6	43.9	67.4	53.3	34.5
Mem0^g	38.2	29.3	26.7	29.5	47.4	71.7	53.3	32.7
LangMem	49.4	40.2	42.2	41.0	50.9	73.9	**71.1**	47.3
Qwen3.8-Max-0902								
NaiveRAG	**73.5**	**63.4**	71.1	59.0	**73.7**	87.0	**84.4**	**92.7**
HippoRAG2	66.9	59.8	70.0	48.7	66.7	84.8	73.3	78.2
A-MEM	73.1	**63.4**	**74.4**	**60.3**	68.4	87.0	80.0	90.9
MemOS	53.4	45.1	44.4	35.9	54.4	82.6	68.9	67.3
EverMemOS	66.2	52.4	58.9	59.0	66.7	**89.1**	82.2	76.4
Mem0	43.0	32.9	30.0	30.8	49.1	73.9	62.2	49.1
Mem0^g	45.0	32.9	32.2	32.1	50.9	80.4	66.7	49.1
LangMem	55.4	43.9	46.7	42.3	57.9	87.0	80.0	56.4

cation search. These comparisons separate access to current application state, access to historical memory, and the effect of extending memory with later observations.

We report *Task Success Rate* as the primary end-to-end metric. A task succeeds only when its terminal MobileGym state satisfies the complete task-specific rubric, including the requested outcome, permitted changes, and applicable side-effect constraints. Detailed memory configurations, tool interfaces, inference settings, and evaluation procedures are provided in Appendix C.

4.2 Performance across Agents, Memory Backends, and Task Dimensions

With evidence access fixed to $M_\tau + E_T$, we first examine how end-to-end performance varies with the downstream agent, memory backend, and the two task dimensions. Tables 2 and 3 report Task Success Rate overall and by Memory Capability and Evidence Condition, respectively.

Agent choice remains decisive under matched memory access. With NaiveRAG as a common backend, overall success is 20.5% for UI-Venus-2-9B, 60.3% for Gemini-3.7-Flash, and 73.5% for Qwen3.8-Max-0902. The largest absolute gaps appear on tasks that depend on historical evidence. Historical Retention increases from 10.9% for UI-Venus-2-9B to 66.7% for Qwen3.8-Max-0902. For the same agents, Cross-Source Joint increases from 8.3% to 71.2%, and Current Sufficient from 47.6% to 85.4%. Access to the same memory backend therefore does not imply comparable use of its evidence. The downstream agent must still identify the relevant state, relate it to the request, and carry that evidence through execution.

Table 3: Task Success Rate (%) by Evidence Condition with $M_\tau + E_T$. Bold marks the best result for each agent, including ties.

Memory Backend	Historical Retention	Current Sufficient	Updated Current	Cross-Source Joint
UI-Venus-2-9B				
NaiveRAG	10.9	**47.6**	31.8	8.3
HippoRAG2	**12.6**	37.9	27.3	**12.1**
A-MEM	10.3	46.6	31.8	8.3
MemOS	2.3	36.9	29.5	0.8
EverMemOS	12.1	43.7	**36.4**	7.6
Mem0	5.2	35.0	22.7	7.6
Mem0^g	2.3	35.9	34.1	7.6
LangMem	2.9	43.7	31.8	3.8
Gemini-3.7-Flash				
NaiveRAG	**54.0**	75.7	72.7	52.3
HippoRAG2	49.4	77.7	**75.0**	50.8
A-MEM	50.0	**79.6**	**75.0**	**54.5**
MemOS	31.0	76.7	70.5	29.5
EverMemOS	46.0	**79.6**	72.7	50.8
Mem0	14.9	72.8	68.2	30.3
Mem0^g	13.2	74.8	**75.0**	30.3
LangMem	30.5	75.7	72.7	46.2
Qwen3.8-Max-0902				
NaiveRAG	**66.7**	85.4	79.5	71.2
HippoRAG2	57.5	83.5	77.3	62.9
A-MEM	66.1	85.4	75.0	**72.0**
MemOS	37.9	83.5	75.0	43.2
EverMemOS	54.0	**88.3**	**81.8**	59.8
Mem0	20.1	79.6	70.5	35.6
Mem0^g	22.4	84.5	72.7	34.8
LangMem	34.5	85.4	79.5	51.5

Backend differences emerge most strongly when historical evidence matters. For Gemini-3.7-Flash, the spread across memory backends is 6.8 percentage points on Current Sufficient, compared with 40.8 on Historical Retention and 25.0 on Cross-Source Joint. For Qwen3.8-Max-0902, the corresponding spreads are 8.7, 46.6, and 37.2 percentage points. All configurations retain App Search, providing a shared route to records available in the query-time environment. Larger differences on Historical Retention and Cross-Source Joint therefore arise when task-critical evidence depends more directly on what the memory backend preserves and returns.

Capability scores reflect both reasoning and evidence burden. Memory Capability and Evidence Condition capture different sources of difficulty. Preference Inference is predominantly supported by evidence that remains available in the query-time environment, whereas Temporal Reasoning more often depends on historical or cross-source evidence. Capability scores therefore reflect both reasoning difficulty and the difficulty of accessing the required evidence. This distinction motivates the next experiment, which directly controls access to the query-time environment and persistent memory.

4.3 What Does Memory Add to Environment Access?

We next isolate the roles of query-time environment access and persistent memory using Qwen3.8-Max-0902, with NaiveRAG as the primary reference and A-MEM as a second strong backend. Table 4 compares four access conditions while holding the task set and query-time environment fixed.

Table 4: Task Success Rate (%) under controlled evidence-access conditions with Qwen3.8-Max-0902. Memory conditions retain GUI interaction and App Search. M_τ is formed at the memory boundary, whereas M_T incorporates records through query time.

Memory Backend	Access Setting	Overall	Historical Retention	Current Sufficient	Updated Current	Cross-Source Joint
—	GUI Only	7.5	0.0	30.1	6.8	0.0
—	GUI + App Search	30.9	8.0	83.5	72.7	6.1
NaiveRAG	$M_\tau + E_T$	73.5	66.7	85.4	79.5	71.2
NaiveRAG	$M_T + E_T$	70.4	60.9	82.5	84.1	68.9
A-MEM	$M_\tau + E_T$	73.1	66.1	85.4	75.0	72.0
A-MEM	$M_T + E_T$	72.6	62.6	85.4	79.5	73.5

Application search makes current evidence accessible. Adding App Search to GUI interaction raises overall Task Success Rate from 7.5% to 30.9%. The gain is concentrated on tasks whose required evidence remains available in the query-time environment. Current Sufficient rises from 30.1% to 83.5%, and Updated Current from 6.8% to 72.7%. In contrast, Historical Retention reaches only 8.0%, and Cross-Source Joint only 6.1%. Structured search therefore substantially reduces the difficulty of locating evidence already present in E_T, but it cannot recover historical information that the query-time environment no longer exposes.

Persistent memory supplies the missing historical evidence. Adding NaiveRAG M_τ on top of GUI and App Search raises overall success from 30.9% to 73.5%. The largest gains occur on Historical Retention, from 8.0% to 66.7%, and Cross-Source Joint, from 6.1% to 71.2%. By comparison, Current Sufficient changes only from 83.5% to 85.4%. A-MEM shows the same pattern, reaching 66.1% on Historical Retention and 72.0% on Cross-Source Joint. These results show that App Search and persistent memory play complementary roles. App Search improves access to evidence that still exists in the current environment, whereas memory provides historical evidence that E_T can no longer supply and enables its combination with current state.

Extending memory with later observations does not guarantee better execution. Replacing M_τ with M_T does not consistently improve end-to-end success. With NaiveRAG, overall success decreases from 73.5% to 70.4%, while A-MEM changes only slightly from 73.1% to 72.6%. The effect also varies across Evidence Conditions. Appendix E.1 compares retrieval at the two checkpoints using the same historical references.

4.4 From Evidence Acquisition to Task Completion

We next examine whether tasks fail before or after the annotated evidence is returned to the agent. For NaiveRAG and A-MEM, which preserve the original input records, we match memory and application returns against each task's annotated reference records. Table 5 reports evidence retrieval under $M_\tau + E_T$ and task success conditioned on complete joint coverage. The diagnostics measure records actually returned during execution, rather than all evidence stored in a source. Matching rules and metric definitions are provided in Appendix C.5.

Agents retrieve different evidence from the same memory state. With NaiveRAG, pre-τ reference recall is 24.0% for UI-Venus-2-9B, 55.6% for Gemini-3.7-Flash, and 64.9% for Qwen3.8-Max-0902. Joint complete coverage similarly increases from 23.4% to 52.5% and 62.3%. The memory backend and checkpoint are identical across these runs. These differences show how the downstream agent's search behavior affects which evidence is returned from memory and the query-time environment.

Complete evidence retrieval is not sufficient for task success. With Qwen3.8-Max-0902, task success conditioned on retrieval of a complete annotated support set is 86.9% for NaiveRAG and 87.8% for A-MEM. The agent must still select the relevant state, reason over the returned information, and realize the requested outcome. Conversely, a task may succeed without complete reference

Table 5: **Reference-record retrieval and task completion under $M_\tau + E_T$.** Pre-τ reference recall is macro-averaged over tasks with reference records within the M_τ ingestion scope. App-search and joint complete coverage use the full task set. Success is conditioned on joint coverage. Parentheses show task counts.

Backend	Evidence retrieval			Task completion
	Pre-τ reference recall (%)	App-search complete coverage (%)	Joint complete coverage (%)	Success given joint coverage (%)
UI-Venus-2-9B				
NaiveRAG	24.0	9.9 (45/453)	23.4 (106/453)	49.1 (52/106)
A-MEM	23.8	9.1 (41/453)	22.3 (101/453)	49.5 (50/101)
Gemini-3.7-Flash				
NaiveRAG	55.6	15.7 (71/453)	52.5 (238/453)	81.1 (193/238)
A-MEM	55.8	14.8 (67/453)	50.1 (227/453)	84.1 (191/227)
Qwen3.8-Max-0902				
NaiveRAG	64.9	18.5 (84/453)	62.3 (282/453)	86.9 (245/282)
A-MEM	65.1	18.1 (82/453)	63.1 (286/453)	87.8 (251/286)

coverage by using GUI observations or other valid evidence. Appendix E.2 illustrates these findings and the backend and checkpoint comparisons through representative execution cases.

5 Related Work

Long-term memory systems and evaluation. Long-term memory systems organize past experience through context management (Packer et al., 2023), fact extraction and consolidation (Chhikara et al., 2025; Chen et al., 2025a), linked memories (Xu et al., 2025), and associative retrieval (Gutiérrez et al., 2025). Memory evaluation has expanded from long-history recall and reasoning (Maharana et al., 2024; Wu et al., 2024; Tavakoli et al., 2026) to incremental memory management and failure diagnosis (Hu et al., 2026b; Chen et al., 2025b; Deng et al., 2026b). It also examines memory-supported agent behavior (Shen et al., 2026; He et al., 2026; Liu et al., 2026a). These works establish that memory quality depends on both what is retained and how it is retrieved and used. Our evaluation extends this perspective to settings where persistent memory and the query-time environment provide distinct, temporally separated evidence sources.

Personalized and executable mobile agents. Recent benchmarks study personalization from behavioral histories (Chen et al., 2026; Nie et al., 2026) and increasingly evaluate agents through stateful mobile interaction (Kong et al., 2026; Liu et al., 2026b; Zhu et al., 2026). MobileMem (Deng et al., 2026a) evaluates longitudinal mobile memory from accumulated experience. MobilePA-Bench (Zhu et al., 2026) combines persistent-memory access with stateful tools. R^2-MobileMemory differs by explicitly controlling when memory is formed relative to the query-time world. It also grounds tasks according to whether the required evidence resides in memory, the current environment, or both.

6 Conclusion

We introduced R^2-MobileMemory to evaluate agents whose persistent memory and current environment provide different temporal views of the same world. Application search is effective when the required evidence remains available, while memory provides its largest benefits when historical evidence has disappeared or must be combined with current information. Extending memory through query time does not consistently improve success, and retrieving a complete annotated support set can still leave failures in state selection and execution. These findings motivate temporal evidence arbitration as a central problem for memory-augmented agents.

AI USE STATEMENT

We used generative AI tools to polish and restructure manuscript text, translate task descriptions, identify relevant literature, and assist with implementation and analysis code. Generative models also supported synthetic dataset construction, including world and record synthesis, image generation, fact extraction, and task authoring and review, as well as semantic outcome evaluation. The models and procedures are documented in Appendices A and C. Generated dataset artifacts were checked through the source reviews and executable checks described in Section 3.4 and Appendix A.4. The authors take responsibility for the final manuscript, dataset, code, and reported findings.

ETHICS STATEMENT

All questionnaire contributors consented to the use of their data for a public research dataset. Seven worlds use questionnaire-derived priors, with identifying details removed and profiles transformed into fictionalized backgrounds; three additional worlds use fully synthetic priors. Downstream identities, events, application records, personal media, and benchmark tasks are generated from these priors. Evaluation runs in isolated MobileGym environments and operates on simulated application state.

REPRODUCIBILITY STATEMENT

Section 4.1 identifies the evaluated agents, memory backends, and evidence-access conditions. Appendices A and B document benchmark construction, validation, dataset composition, and task design. Appendix C describes memory formation, checkpoint selection, backend configurations, and the agent execution protocol. Appendix D provides tool interfaces and prompt excerpts. Task-success evaluation and reference-record matching and retrieval metrics are specified in Appendices C.4 and C.5, respectively. We will provide reviewers and area chairs with an anonymous code repository through OpenReview during the discussion period.

REFERENCES

Yuanchen Bei, Tianxin Wei, Xuying Ning, Yanjun Zhao, Zhining Liu, Xiao Lin, Yada Zhu, Hendrik Hamann, Jingrui He, and Hanghang Tong. Mem-gallery: Benchmarking multimodal long-term conversational memory for mllm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 40750–40784, 2026.

Huacan Chai, Yukai Wang, Yingxuan Yang, Dan Peng, Yuanyi Song, Zhihui Fu, Weiwen Liu, Jianghao Lin, Jun Wang, and Weinan Zhang. Smmbench: A benchmark for source-distributed multimodal agent memory. *arXiv preprint arXiv:2605.15710*, 2026.

Chunliang Chen, Ming Guan, Xiao Lin, Jiaxu Li, Luxi Lin, Qiyi Wang, Xiangyu Chen, Jixiang Luo, Changzhi Sun, Dell Zhang, et al. Telemem: Building long-term and multimodal memory for agentic ai. *arXiv preprint arXiv:2601.06037*, 2025a.

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. *arXiv preprint arXiv:2511.03506*, 2025b.

Tongbo Chen, Zhengxi Lu, Zhan Xu, Guocheng Shao, Shaohan Zhao, Fei Tang, Yong Du, Kaitao Song, Yizhou Liu, Yuchen Yan, et al. Knowu-bench: Towards interactive, proactive, and personalized mobile agent evaluation. *arXiv preprint arXiv:2604.08455*, 2026.

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

Xinle Deng, Yida Xue, Xiangyuan Ru, Haoming Xu, Shuofei Qiao, Mengru Wang, Yijun Chen, Buqiang Xu, Chen Jiang, Yuchen Eleanor Jiang, et al. Mobilemem: Learning from a year of mobile experiences. *arXiv preprint arXiv:2608.13606*, 2026a.

Xinle Deng, Ruobin Zhong, Hujin Peng, Xiaoben Lu, Yanzhe Wu, Guang Li, Buqiang Xu, Yunzhi Yao, Jizhan Fang, Haoliang Cao, et al. Memtrace: Tracing and attributing errors in large language model memory systems. *arXiv preprint arXiv:2605.28732*, 2026b.

Google DeepMind. Gemini 3.7 flash model card, August 2026. URL `https://deepmind.google/models/model-cards/gemini-3-7-flash/`.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, et al. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. *arXiv preprint arXiv:2601.02163*, 2026a. URL `https://arxiv.org/abs/2601.02163`.

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

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6142–6167, 2026.

LangChain Team. LangMem SDK for agent long-term memory. Official documentation and release announcement, 2025. URL https://www.langchain.com/blog/langmem-sdk-launch.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2005.11401.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 986–995, 2017.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, et al. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025. URL https://arxiv.org/abs/2507.03724.

Chengzhi Liu, Yuzhe Yang, Sophia Xiao Pu, Yepeng Liu, Lin Long, Yichen Guo, Nuo Chen, Zhaotian Weng, Elena Kochkina, Simerjot Kaur, et al. Worldmemarena: Evaluating multimodal agent memory through action-world interaction. *arXiv preprint arXiv:2605.29341*, 2026a.

Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Qinyi Luo, Shunye Tang, Yuxiang Chai, Weifeng Lin, Han Xiao, WenHao Wang, Siheng Chen, et al. Memgui-bench: Benchmarking memory of mobile gui agents in dynamic environments. *arXiv preprint arXiv:2602.06075*, 2026b.

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

Hongyi Nie, Xunyuan Liu, Yudong Bai, Yaqing Wang, Yang Liu, Quanming Yao, and Zhen Wang. Pspa-bench: A personalized benchmark for smartphone gui agent. *arXiv preprint arXiv:2603.29318*, 2026.

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

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026a. URL https://qwen.ai/blog?id=qwen3.8.

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

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pp. 406–441, 2025.

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.

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

Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J. Ross Mitchell. Beyond a million tokens: Benchmarking and enhancing long-term memory in llms. In *International Conference on Learning Representations*, volume 2026, pp. 133343–133429, 2026.

Venus Team, Zhuohan Cai, Haoxing Chen, Jiaxuan Chen, Weizhi Chen, Changlong Gao, Zhangxuan Gu, Yuan Guo, Yusong Hu, Jianrong Jiang, et al. Ui-venus-2 technical report. *arXiv preprint arXiv:2609.00028*, 2026.

Xiaoyang Wang, Chen Li, Jianqiao Zhao, and Dong Yu. Naturalconv: A chinese dialogue dataset towards multi-turn topic-driven conversation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14006–14014, 2021.

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

Dingbang Wu, Rui Hao, Haiyang Wang, Shuzhe Wu, Han Xiao, Zhenghong Li, Bojiang Zhou, Zheng Ju, Zichen Liu, Lue Fan, et al. Mobilegym: A verifiable and highly parallel simulation platform for mobile gui agent research. *arXiv preprint arXiv:2605.26114*, 2026.

Jianfei Xiao, Xiang Yu, Chengbing Wang, Wuqiang Zheng, Xinyu Lin, Kaining Liu, Hongxun Ding, Yang Zhang, Wenjie Wang, Fuli Feng, et al. Alpsbench: An llm personalization benchmark for real-dialogue memorization and preference alignment. *arXiv preprint arXiv:2603.26680*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Jiahao Zhang, Yongzhi Tong, Zelin Fu, Pengde Zhao, Yanmei Jiang, Feng Jiang, and Min Yang. Lunar: Benchmarking personalized large language models on universal user behavior logs. *arXiv preprint arXiv:2608.05246*, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

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

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

Yi Zhu, Xiongwei Wu, Qiyi Wang, Tingyu Qu, Jiajun Liu, Sihan Cao, Long Chen, Weigao Sun, Feida Zhu, Yiran Zhong, et al. Mobilepa-bench: Benchmarking mobile planner agents on complex real-world tasks. *arXiv preprint arXiv:2608.23035*, 2026.

A Benchmark Construction and Validation

A.1 World Planning and Behavioral Priors

Each world is initialized from a structured user prior u. Seven priors are derived from volunteer questionnaires describing smartphone habits, recurring activities, relationships, and broader life patterns. Three are generated synthetically to broaden world coverage. Identifying details are removed before synthesis, and each questionnaire-derived profile is transformed into a fictionalized prior. All downstream identities, events, application records, and personal media are generated from these priors rather than copied from the questionnaire responses.

World planning organizes each prior hierarchically. Recurring activities, relationships, routines, and patterns of application use are first grouped into life themes. Each theme is expanded into concrete events, and each event is decomposed into one or more temporally bounded sessions. A session represents a coherent unit of user activity that can later be realized through one or more mobile applications.

Events may depend on information or state established by earlier events. We represent these dependencies as a directed acyclic graph, where an edge $e_i \to e_j$ indicates that event e_j depends on context introduced by e_i. Sessions are placed on a shared timeline while respecting these dependencies and their temporal constraints, producing the ordered history

$$H = (h_1, \ldots, h_n).$$

After the timeline is fixed, we select a complete-session boundary τ and a world-level query time T, with $\tau < T$. The boundary partitions the planned history into sessions whose realized records are available for memory formation and later sessions that continue to evolve the world before evaluation. All benchmark tasks constructed from the same world share the same query time T.

A.2 Application State Realization

Each planned session is realized as a collection of application-specific records. Fact Intents specify candidate facts and state transitions that should be expressed during realization, while Task Blueprints specify the Memory Capability and Evidence Condition coverage targeted by later task construction. These planning artifacts constrain generation. Final evidence, target values, and task outcomes are determined during task grounding.

Sessions may be realized through one or more applications, producing records such as messages, calendar events, notes, contacts, files, orders, and media. Records that refer to the same people, events, or application objects share consistent identities and references across sessions. Application state persists over the session timeline, so later sessions may update, replace, or otherwise modify objects created earlier. The resulting ordered collection of realized records forms the multimodal application-record history R.

Conversational records are generated with prompt designs informed by dialogue corpora (Wang et al., 2021; Li et al., 2017). Personal images are generated from session-conditioned scene descriptions, while application screenshots are rendered from the corresponding MobileGym state. These media are bound to the same underlying events and application objects as the textual records.

The query-time environment E_T is obtained by projecting R at time T according to application-specific state semantics. The projection retains the records and object states visible at query time but need not preserve overwritten or removed historical versions. In contrast, persistent memory M_τ is constructed only from records available through the memory boundary τ. This construction allows historical information to remain available through memory even when it is no longer exposed by the query-time environment.

A.3 Fact Ledger and Task Grounding

After application-state realization, R^2-Forge extracts atomic facts from the reviewed records into a Fact Ledger F. Each fact represents an independently interpretable property of an entity or event. It retains references to supporting records together with its temporal context. Facts referring to the same subject across sessions can therefore be linked as different versions or related states.

The construction can be summarized as

$$u \xrightarrow{\text{plan}} H \xrightarrow{\text{realize}} R \xrightarrow{\text{extract}} F \xrightarrow{\text{ground}} \{q_i\}.$$

For task construction, each Task Blueprint is resolved against the realized Fact Ledger. Candidate facts that are not supported by the realized records are discarded. The resulting grounded task specification binds the required evidence, target outcome, and applicable state-preservation constraints to concrete source records. Only after this grounding is fixed does an LLM render the specification as a user-facing request. This separation prevents planned but unrealized content from becoming benchmark ground truth. Each final task is assigned one primary Memory Capability and one Evidence Condition according to its grounded evidence requirements.

A.4 Task Validation Procedure and Coverage

We validate each task for source grounding, evidence dependency, and executable outcome.

Source grounding. Each task is independently checked against the frozen realized records. The review verifies that the target outcome, temporal relations, entity references, and designated evidence are supported by the underlying world. Reviewers also inspect the query-time environment for alternative support, especially when a task is intended to depend on historical information that is no longer available in E_T.

Evidence dependency. For each task q_i, let

$$G_i = \{g_{i1}, \ldots, g_{ik_i}\}$$

denote its designated evidence. For targeted counterfactual checks, we additionally define removal sets

$$C_i = \{C_{ij}\}_{j=1}^{r_i}, \qquad C_{ij} \subseteq G_i.$$

We examine the task under three evidence settings:

$$G_i, \qquad \emptyset, \qquad \{G_i \setminus C_{ij}\}_{j=1}^{r_i}.$$

The full-evidence setting checks that the intended support is sufficient. The empty-evidence setting checks whether the target remains recoverable without that support. Targeted removals test whether deleting task-critical evidence changes answerability as annotated. Across the benchmark, 1,161 evidence-removal conditions are structurally validated against the frozen evidence specification.

Executable outcome. Every task is evaluated with a terminal-state rubric that specifies the required outcome, permitted changes, and applicable state-preservation constraints. For each rubric, we instantiate a terminal state that satisfies the task and verify that it is accepted. We also evaluate the unchanged initial state to ensure that the task is not credited without realizing the requested outcome. Together, these checks connect grounded evidence to a verifiable executable result.

A.5 Construction and Review Models

Table 6 lists the models used at each stage of the construction and validation pipeline.

MobileGym renders GUI screenshots from application states. Deterministic code handles identifiers, timestamps, reference binding, state projection, and applicable authorization and side-effect checks. Appendix D.3 provides excerpts of the fact-extraction, task-authoring, source-review, and semantic-judging prompts.

B Benchmark Composition and Task Design

B.1 World and Task Composition

Table 7 summarizes the frozen 453-task release. Each world provides a temporally ordered history of application records and a query-time mobile environment. The Chinese and English portions

Table 6: Models used in benchmark construction and validation.

Role	Model
World planning and task authoring	GPT-5.6-sol
Record generation and grounding	Gemini 3.1 Pro Preview
Stage review	Claude Opus 5
Image generation	GPT Image 2
Captioning and OCR	Gemini 3.6 Flash
Image and visual-fact review	Gemini 3.6 Flash
Final task source review	Gemini 3.6 Flash
Semantic field validation	Qwen3.6-27B

Table 7: Dataset composition. Means weight the ten worlds equally.

Statistic	Value
Worlds (Chinese / English)	10 (5 / 5)
Tasks (Chinese / English)	453 (211 / 242)
Applications covered	23
Application records	85,840
Records per world, mean	8,584
Records per world, range	7,603–11,393
History span per world (days)	462–547
Mean record share for M_τ (%)	56.89

contain five worlds each, with 211 and 242 tasks, respectively. All 453 tasks are executable through the GUI. Each task specifies a target terminal state and the permitted scope of changes. Success is determined by the corresponding state-based rubric. The ten world histories span 462–547 days. Figure 3 shows their average cumulative record distribution after normalizing time and record counts within each world. The memory boundary falls at 39.9% of the observation window on average, with 56.89% of records available for forming M_τ. M_T incorporates the remaining records through query time. The blue curve and orange dashed line show the mean record share and memory boundary, respectively, with shaded bands indicating one standard deviation across worlds.

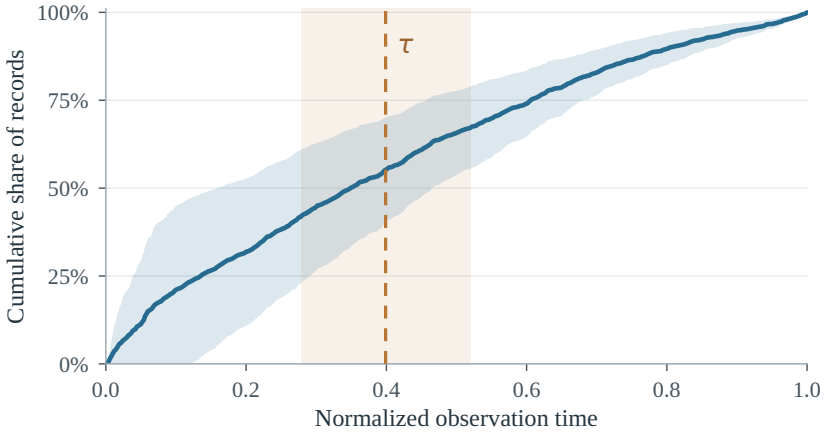

Figure 3: **Record accumulation and memory boundary across the ten worlds.**

B.2 Memory Capabilities and Evidence Conditions

Each task has one primary Memory Capability and one Evidence Condition, the two dimensions introduced in Section 3.3. Memory Capability specifies the operation required over user-specific evidence, whereas Evidence Condition specifies how task-critical evidence is distributed between persistent memory and the query-time environment. Table 8 reports their distributions over all 453 tasks. The counts in each block sum to 453. Capability counts range from 45 to 90 tasks (9.9–19.9%), with Update Resolution the largest category. Historical Retention and Cross-Source Joint together account for 306 tasks (67.5%), providing substantial coverage of requests whose required historical evidence is no longer fully available in the query-time environment. Current Sufficient and Updated Current cover requests supported by the current environment, with the latter explicitly involving an update to earlier information. Appendix B.3 provides representative tasks illustrating each Memory Capability and Evidence Condition.

Table 8: Task composition by Memory Capability and Evidence Condition. Percentages use all 453 tasks as the denominator and are rounded independently.

Category	Tasks	Share (%)
Memory Capability		
Factual Recall	82	18.1
Update Resolution	90	19.9
Temporal Reasoning	78	17.2
Entity Resolution	57	12.6
Preference Inference	46	10.2
Implicit State Inference	45	9.9
Visual Recall	55	12.1
Evidence Condition		
Historical Retention	174	38.4
Current Sufficient	103	22.7
Updated Current	44	9.7
Cross-Source Joint	132	29.1

B.3 Representative Task Cases

The two task dimensions describe different properties of the same executable task. Memory Capability specifies *what operation* is required over user-specific evidence, whereas Evidence Condition specifies *where the task-critical evidence is available* at query time. We illustrate each Memory Capability with one representative task and then reuse selected tasks to illustrate the four Evidence Conditions. All user requests below are taken directly from the frozen English tasks.

B.3.1 Memory Capability Cases

Factual Recall. *User request: "Set my recurring Work alarm back to the time it had on February 3. Keep it enabled and preserve all its current settings other than the time."*

The Work alarm was set to **8:00 AM** on February 3. The task requires recovering this specific historical value and applying it to the existing alarm. It is therefore a Factual Recall task.

Update Resolution. *User request: "I'm checking how my pepper stir-fry seasoning was resolved from the original unfinished draft to the current tested recipe. What did the draft say still needed testing, and how much dark soy sauce does the current seasoning use?"*

The original draft says that **the exact seasoning amounts** still need testing, while the current recipe uses **half a tablespoon of dark soy sauce**. The task requires distinguishing two versions of the same recipe and recovering the value associated with each version. It is therefore an Update Resolution task.

Temporal Reasoning. *User request: "Compare my initial camera-cleaning reminder with the latest scheduled cleaning. What time did the initial reminder start, what time does the latest cleaning start, and how many calendar days separate their dates?"*

The initial reminder starts at **10:00 AM**, and the latest cleaning starts at **10:30 AM**. Their dates are **244 calendar days apart**. The task requires relating records from two points in time and computing the interval between them. It is therefore a Temporal Reasoning task.

Entity Resolution. *User request: "Find the mechanic who gave me the chain-cleaning advice, then message him to confirm that I'll follow his current bike handoff routine and notify him as requested."*

The mechanic is not named directly in the request. The agent must identify him from the earlier chain-cleaning interaction and use that identity to send the requested message. It is therefore an Entity Resolution task.

Preference Inference. *User request: "Schedule a 45-minute Third Work Review on August 1, 2026, at the time I usually use for work review meetings."*

Previous work-review meetings repeatedly start at **10:30 AM**. The task requires inferring this recurring pattern as the user's usual meeting time and applying it to a new event. It is therefore a Preference Inference task.

Implicit State Inference. *User request: "Across the September 2, 2025 Q2 correction and the May 15, 2026 billing period, what prerequisite do I appear to use before paying a property-fee notice?"*

The two billing episodes indicate the unstated prerequisite **verify the statement before paying**. This condition is not explicitly recorded as a user rule and must instead be derived from the observed sequence of corrections and payments. It is therefore an Implicit State Inference task.

Visual Recall. *User request: "In the CloudPurse chat where you supplied the requested 15-digit order number for an electronic refund receipt, what color is the paper in the receipt image sent immediately afterward?"*

The chat identifies the relevant receipt image, and the paper in that image is **white**. The task requires using the conversational context to locate the image and recover an attribute grounded in its visual content. It is therefore a Visual Recall task.

B.3.2 Evidence Condition Cases

Evidence Condition describes where the task-critical evidence is available at query time. To make this dimension concrete, we revisit four representative tasks from the Memory Capability cases above. Each example is identified by both its task and its Memory Capability so that the two dimensions can be read independently.

Historical Retention. *Example: the Work-alarm restoration task from Factual Recall.*

The required February 3 alarm time is no longer available in E_T. The agent must therefore recover this historical value from information retained before the query-time state. The task is classified as Historical Retention.

Current Sufficient. *Example: the Third Work Review task from Preference Inference.*

The meeting records needed to infer the usual **10:30 AM** start time remain available in E_T. The query-time environment therefore contains all task-critical evidence. The task is classified as Current Sufficient.

Updated Current. *Example: the bike-handoff task from Entity Resolution.*

The handoff routine is updated after τ, and the latest instructions are available in E_T. These current instructions supersede the earlier routine and provide the authoritative evidence for the requested message. The task is classified as Updated Current.

Cross-Source Joint. *Example: the pepper stir-fry task from Update Resolution.*

The unresolved seasoning detail comes from the original draft and is no longer available in E_T. The tested dark-soy quantity comes from the later recipe introduced after τ. Both pieces are required, so neither historical memory nor the query-time environment is sufficient alone. The task is classified as Cross-Source Joint.

B.4 Thematic Coverage

Figure 4 shows the distribution of life domains and concrete scenarios linked to the tasks. Domain labels are derived from each task's anchor event, with synonymous Chinese and English labels merged into 11 domains. The original event theme titles are grouped into 51 scenarios, such as meeting coordination, household repairs, and family contact. By describing the source events associated with tasks, these labels complement the capability and evidence annotations.

Tasks associated with multiple domains are weighted equally across those domains.

B.5 Extended Comparison with Existing Benchmarks

Table 9 extends Table 1 with dataset size, task type, language, and history length. History lengths count visible text, roles, timestamps, and available image descriptions using `tiktoken` 0.13.0. Queries, answers, profile metadata, generation logs, and visual-token costs are excluded. Each history is counted once using the unit listed in the table. MobileMem-Omni follows the paper's collection and reported mean. KnowU-Bench also uses the author-reported task count. Other counts come from the inspected releases.

Table 9: **Benchmark scale and history length.** Lengths use integer k below one million tokens and M with one decimal place otherwise (k = 1,000; M = 1,000,000).

Benchmark	Histories	Tasks	Task type	Language	History unit	History length		
						Min.	Mean	Max.
LoCoMo-10	10	1,986	QA	EN	Conversation	13k	21k	25k
LongMemEval-S	500	500	QA	EN	Query context	97k	103k	106k
BEAM-100K	20	400	QA	EN	Conversation	94k	116k	123k
BEAM-500K	35	700	QA	EN	Conversation	413k	506k	718k
BEAM-1M	35	700	QA	EN	Conversation	807k	960k	1.0M
BEAM-10M	10	200	QA	EN	Conversation	9.6M	9.7M	9.8M
Mem2ActBench	2,029	400	Tool use	EN	Session	1k	3k	16k
MemoryArena	—	701	Interactive	EN	Online rollout	—	—	—
WorldMemArena	461	461	QA	EN	Sample history	1k	12k	90k
MobileMem (Text)	2	1,319	QA	ZH	User history	186k	227k	269k
MobileMem (Omni)	16	7,415	QA	EN/ZH	User history	—	1.7M*	—
MemGUI-Bench	—	128	GUI	EN	Online rollout	—	—	—
KnowU-Bench	4	192	GUI	ZH	Profile log	6k	9k	12k
R^2-MobileMemory	10	453	GUI	EN/ZH	World history	464k	594k	952k

EN: English; ZH: Chinese. A dash indicates an unavailable value or no fixed history.
*Author-reported mean from MobileMem Table 3 (Deng et al., 2026a); all other non-empty lengths use `o200k_base`.



Figure 4: **Thematic coverage of the benchmark.** The inner ring shows 11 life domains, and the outer ring shows 51 scenarios. Sector sizes reflect weighted task counts.

C Experimental Protocol

C.1 Memory Formation and Checkpoints

Backends ingest the same ordered stream of application records, one record at a time, using the common textual projection of record content and accepted visual facts. The shared text input enables comparison across backends with different modality support. MobileMem-Omni similarly converts images to captions when evaluating text-based memory methods (Deng et al., 2026a). For each world, we save $M_\tau = \mathrm{Ingest}(R_{\leq \tau})$ at the memory boundary and continue ingestion through the query time to obtain $M_T = \mathrm{Ingest}(R_{\leq T})$. Where a backend requires these components, we use Qwen3.6-27B (Qwen Team, 2026b) for memory operations and Qwen3-Embedding-8B (Zhang et al., 2025) with 4,096-dimensional embeddings. Each evaluation task runs independently with the selected memory checkpoint and the same query-time environment E_T. The agent can retrieve information from memory but cannot modify it during execution.

C.2 Memory Backends

All eight backends ingest the same application-record stream and expose retrieval through `memory_search`. They differ in how they organize incoming records and retrieve evidence for the agent.

- **NaiveRAG** is a dense-retrieval baseline following the retrieval-augmented paradigm (Lewis et al., 2020). It stores the original record text in an embedding index and returns the memory units most relevant to each query.
- **HippoRAG2** (Gutiérrez et al., 2025) builds a graph of extracted phrases and source passages. Dense retrieval and LLM-based relevance filtering identify query-relevant evidence, which initializes Personalized PageRank over the graph. The resulting ranking retrieves passages through associative connections.
- **A-MEM** (Xu et al., 2025) organizes records as interconnected notes with contextual descriptions, keywords, and tags. During ingestion, it links new notes to related memories and updates their contextual attributes, allowing the memory structure to evolve with the record stream.
- **MemOS** (Li et al., 2025) manages memory through MemCubes that package content with provenance and lifecycle metadata. We use its `TreeTextMemory` component to organize textual records and retrieve relevant memory content.
- **EverMemOS** (Hu et al., 2026a) extracts episodic MemCells containing facts and temporally scoped information, then consolidates related cells into thematic MemScenes. We use hybrid retrieval over this memory.
- **Mem0 and Mem0^g** (Chhikara et al., 2025) maintain fact-based memory. Mem0 extracts salient facts from incoming records and reconciles them with existing memory through addition, update, and deletion. Mem0^g adds entities and relations to this fact store and combines graph-based context with vector retrieval. We evaluate each configuration as a separate backend.
- **LangMem** (LangChain Team, 2025) uses an LLM-based memory manager to extract, consolidate, and update persistent information. Its memory-store manager maintains record-derived memories in an embedding-indexed store and retrieves relevant entries for each query.

C.3 Agent Execution Protocol

Each task runs independently in MobileGym, starting from the same query-time environment E_T across compared conditions. The agent receives the user request, the current screenshot, and the tools available under its assigned evidence-access condition. Each run is limited to 50 interaction steps.

One interaction step corresponds to one model response. Within a step, the agent may issue multiple read-only retrieval calls or one GUI action. GUI actions and retrieval calls cannot be combined in the same step. Previous tool calls and their results remain in the conversation history, while each new screenshot replaces the previous screenshot. Screenshots are used without resizing.

GUI Only provides GUI interaction. GUI + App Search adds read-only search and result reading over E_T. The two memory conditions retain these tools and add `memory_search` over either M_τ or M_T. They use identical agent instructions, with the evaluator selecting the memory checkpoint for each run. Memory retrieval returns at most ten entries per call, and the loaded checkpoint remains read-only throughout execution. All changes to the environment are performed through GUI interaction. Detailed tool schemas and agent instructions are provided in Appendix D.

Example execution. *User request: "Please use the departure lead time I used for the confirmed September 2025 after-work class and apply it to the latest after-work class start time. Update my current 'Leave for class' alarm accordingly, keeping its current label, weekday repeat, and enabled state."*

In Figure 5, Qwen3.8-Max-0902 with NaiveRAG M_τ retrieves the historical class time of 19:00 and departure alarm of 18:00, establishing a one-hour lead time. Application search returns the latest class time of 19:30, so the agent changes the alarm from 18:45 to 18:30 in Clock. The update passes evaluation and preserves the alarm's label, weekday repeat, and enabled state.

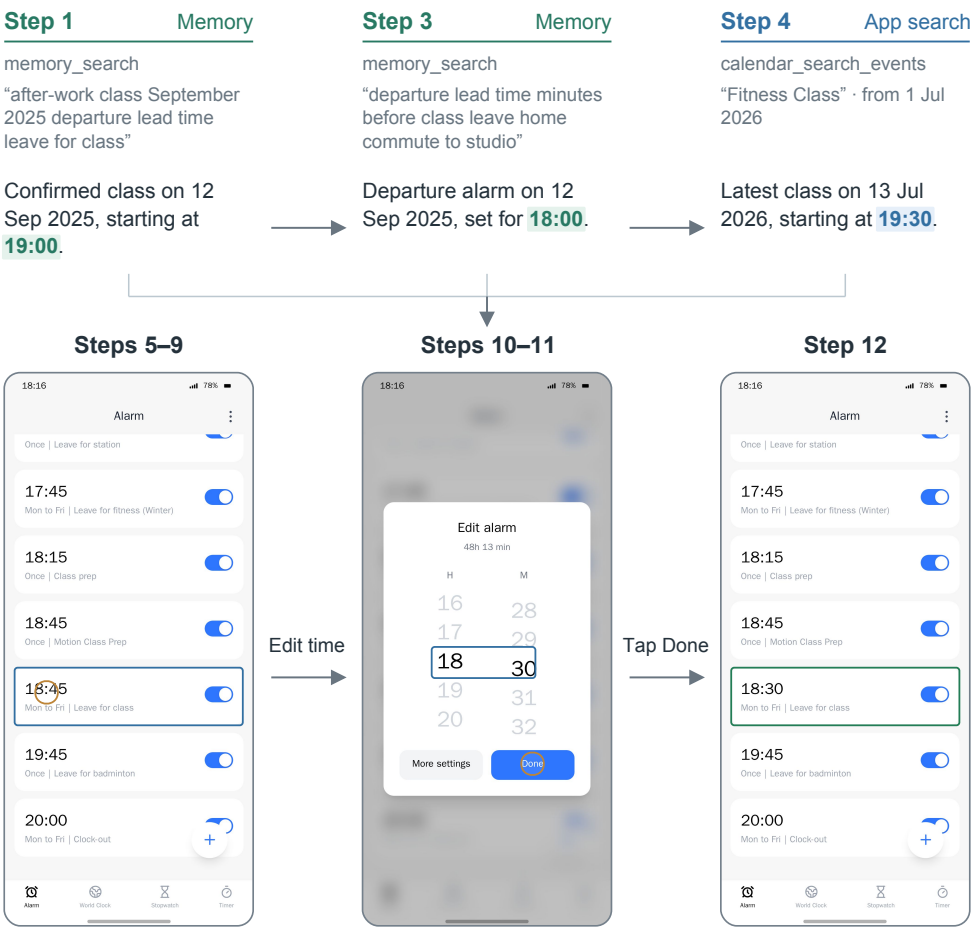

Figure 5: **Executing a memory-grounded alarm update.** GUI frames correspond to steps 9, 11, and 12. Outlines mark the target; circles mark tap locations.

C.4 TASK EVALUATION

Task Success Rate is the primary end-to-end metric. Each task specifies a terminal-state rubric covering the requested outcome, authorized changes, state-preservation requirements, and applicable side-effect constraints. All tasks require GUI actions to realize the requested terminal state. A task succeeds when the final MobileGym state satisfies the complete rubric, including any task-required content submitted through the GUI.

The evaluator checks the realized outcome. An agent's declaration of completion alone does not establish success. Structured state properties are checked deterministically. Criteria requiring semantic equivalence are judged against the expected values and observed terminal-state content. These judgments are combined with the deterministic checks and the task's applicable authorization and side-effect constraints.

A run ends when the agent calls `gui_complete` or `gui_abort`, or reaches the 50-step limit. Runs that end without satisfying the complete rubric are counted as failures. The same task specification and evaluator are used across all agents, memory backends, and evidence-access conditions. Prompt excerpts for semantic outcome evaluation are provided in Appendix D.3.

C.5 REFERENCE-RECORD EVALUATION

Annotated support sets. For each task, we use the frozen evidence records and records cited by its field-level proofs to identify the source records supporting the target. Complete reference coverage means that all records in at least one annotated support set are returned during the evaluated trajectory. One task has three alternative support sets. Returning any one of these sets in full satisfies this criterion. We count memory and application returns that are available before a subsequent agent decision. GUI observations are excluded. Coverage therefore describes what the agent retrieved during a run, rather than everything available in memory or the environment.

Reference matching. We match retrieved items to the original source records. For memory results, we check whether the source-record content is present. For application results, we compare the object's relevant public fields, version, and content with the reference.

For text, we combine reads across pages, normalize spaces and line breaks, and check for the complete source text. For files, the comparison follows the annotated evidence: file attributes are matched against all quoted metadata, and document contents are matched against the full body text. An image reference is covered when the corresponding complete, uncropped image is returned.

Matching excludes generated summaries and agent edits. Joint coverage counts each matched source record once, including records retrieved through both memory and application search.

Retrieval metrics. *Pre-τ reference recall* measures retrieval of reference records within the M_τ ingestion scope. For each task, we compute the fraction of eligible reference records returned by memory search, then average across tasks with at least one eligible reference. Of the 453 tasks, 426 contain at least one reference available by τ. The remaining 27 tasks have no eligible references, so their recall is undefined (N/A) and they are omitted from this average rather than assigned zero. References introduced after τ are outside the checkpoint's ingestion scope and are excluded rather than counted as retrieval failures. The full M_T ingestion horizon includes all task references, making all 453 tasks eligible for recall over that horizon. The paired checkpoint analysis instead holds the pre-τ reference set fixed at both checkpoints.

Memory complete coverage is the fraction of tasks for which memory retrieval alone returns a complete annotated support set. *App-search complete coverage* applies the same criterion to application returns from E_T. *Joint complete coverage* uses the union of memory and application returns from the same trajectory. All three complete-coverage metrics use 453 tasks as the denominator. *Success given joint coverage* is Task Success Rate among tasks with joint complete coverage.

These metrics distinguish record retrieval from successful task execution. A task can succeed without complete reference coverage by using GUI observations or other valid evidence. Complete coverage does not establish that the agent uses the returned records correctly.

D TOOL INTERFACES AND PROMPT SPECIFICATIONS

D.1 TOOL INTERFACES

Tables 10 and 11 list the GUI and read-only retrieval tools. GUI Only exposes the GUI tools. GUI + App Search adds application search and result reading. Both memory conditions also expose `memory_search`. GUI tools perform all state changes, while retrieval reads the current environment E_T or the selected memory checkpoint. Screen coordinates are normalized to $[0, 1000]$ on each axis, with points specified as (x, y).

Table 10: **GUI control tools.**

Tool	Action
`gui_click`	Tap a location
`gui_double_tap`	Double-tap a location
`gui_long_press`	Press and hold
`gui_type`	Enter text; optionally clear first
`gui_swipe`	Swipe between locations
`gui_drag`	Drag between locations
`gui_back`	Go back
`gui_home`	Return to the home screen
`gui_recent`	Show recent apps
`gui_enter`	Press Enter
`gui_wait`	Wait for 0–10 seconds
`gui_awake`	Open an app by its name
`gui_complete`	Mark completion
`gui_abort`	Stop and provide a reason

Table 11: **Read-only retrieval tools.** Inputs are summarized by function.

Tool	Inputs
`memory_search`	Required query; up to 10 memory entries
`wechat_search_messages`	Query, conversation, sender, type, direction, time
`sms_search_messages`	Query, counterparty, phone, direction, time
`calendar_search_events`	Query, location, start time
`tencent_meeting_search_meetings`	Query, kind, host, status, start time
`contacts_search`	Query, scope, call direction, call time
`notes_search_notes`	Query, folder, update time
`file_manager_search_files`	Query, path prefix, media type, saved/modified time
`gallery_search_media`	Query, album, media type, saved time
`railway12306_search_orders`	Train, stations, passenger, status, travel date
`phone_read_result`	Result reference; offset, limit, context, crop

Application search combines literal text matching with application-specific filters. Time and date filters accept ranges. Queries allow up to 64 characters and may be omitted when explicit filters are provided. Results are paginated, with at most 20 hits per page and a target size of approximately 16 KiB. The fields `next_cursor` and `has_more` indicate further results. Memory search requires a non-empty query of at most 1,000 characters, with the retrieval limit fixed by the evaluator.

The agent passes a returned `result_ref` to `phone_read_result` to read full text, surrounding messages, or an image. Only `result_ref` is required. The parameters `offset`, `limit`, `context`, and `crop` are optional. Text reads use character offsets and return 4,000 characters by default, up to 8,000. Conversational context uses message offsets. Image crops use normalized coordinates $[l, t, r, b] \in [0, 1000]^4$.

D.2 AGENT PROMPT

Table 12 provides excerpts of the agent instructions implementing the execution protocol in Appendix C.3, grouped by access condition and followed by a summary of task-completion requirements.

Table 12: **Agent instructions.** Prompt excerpts grouped by access condition, followed by a summary of completion requirements.

Role and interaction	*Search-enabled conditions*
You are a mobile-phone agent. Retrieve information and operate the current phone only through the registered function tools. You may call multiple read-only retrieval tools in one turn. GUI tools change the screen, so call at most one GUI tool per turn and do not combine it with retrieval tools in the same turn. The conversation retains your previous tool calls and their results; only the current screen screenshot is kept. Search returns `result_ref` values. Use `phone_read_result` to read full text, nearby messages, or actual images; crop images to inspect details when needed.	
Memory sources	$M_\tau + E_T$ and $M_T + E_T$
`memory_search` returns long-term memories formed from past experiences. Phone records may have been updated or overwritten since those memories were formed, and the phone displays only its current state. Earlier historical information may therefore exist only in memory, while recent changes may exist only on the phone. Both are available read-only evidence sources, and neither is guaranteed to be sufficient on its own. Choose sources according to the question. `memory_search` returns up to 10 memories per call. The retrieval count is fixed by the evaluation; provide only query and choose its wording for the question.	
Role and access	*GUI Only*
You are a mobile-phone agent. Operate the current phone through the registered GUI function tools. Call at most one GUI tool per turn. Read information on the screen; you may also use search interfaces inside apps.	
Completion and output	*Summary*
Verify the information needed for the user request, then use the GUI to realize the required terminal state. Follow the task-specific completion instruction and modify only authorized targets. Coordinates are normalized to 0–1000 in (x, y) order. Preserve exact names, source quotations, numbers, and required field values.	

D.3 Construction and Evaluation Prompt Excerpts

Table 13 presents edited English excerpts of the instructions used for fact extraction, task authoring, source review, and outcome evaluation. Fact extraction grounds the supplied Fact Intents in reviewed session records and the subject catalog. The extracted facts undergo automated checks of their structure and source references, followed by a separate model review against the records. Construction and validation use the task targets and supporting evidence. The evaluated agent receives the user request and permitted tools.

Table 13: **Construction and evaluation instructions.** Edited prompt excerpts with the inputs and outputs of each stage.

Fact extraction
Extract one to six atomic facts from the reviewed, user-visible records in the current session. Each fact must express one independently updateable property, identify its subject and value, and cite supporting records. Cover the supplied Fact Intents only when realized in the records. Preserve uncertainty and pending status; use the latest supported state within the session. Leave cross-session version linking to the subsequent stage.
Input: Session records, subject catalog, Fact Intents, and review feedback.
Output: structured facts with values, dimensions, and source references.
Task authoring
Construct one executable mobile task from the supplied frozen records. Focus on one coherent user goal. Preserve record identities, timestamps, and the execution contract. Ground every target value in source evidence; do not reveal it in the user request. Specify the required terminal state, authorized changes, and state-preservation constraints. Assign the memory capability and evidence condition from the actual dependency on historical and current sources.
Input: task blueprint and source packet.
Output: user request, target contract, evidence references, and field-level source proofs.
Independent source review
Check every target value against the frozen records. Verify temporal calculations, entity uniqueness, evidence sufficiency, and the permitted state changes. Search current records for alternative sources for information labeled as historical. Check that the assigned capability is required and that removing the designated evidence changes answerability as specified. Return accept, revise, or reject with concrete findings.
Input: candidate task and independent access to its source packet.
Output: source findings, evidence-condition assessment, and revision issues.
Semantic outcome evaluation
Evaluate the observed result against the requested target and supplied criterion. Accept semantically equivalent expressions when they preserve the required entity, time, quantity, and meaning. Reject unsupported or conflicting content. Return a structured decision and its reason.
Input: criterion, expected value, and observed terminal-state content.
Output: criterion-level judgment. Deterministic checks and applicable authorization and side-effect checks are combined with these judgments to determine task success.

E Additional Experimental Diagnostics and Case Studies

We compare retrieval at the two memory checkpoints using the same historical references, then examine representative trajectories showing backend differences, the effects of later observations, and errors in using retrieved evidence.

E.1 Paired Memory-Checkpoint Analysis

The eligible reference sets differ between M_τ and M_T. To compare retrieval of historical evidence, we evaluate the same references already available by τ at both checkpoints. Following the Qwen3.8-Max-0902 checkpoint comparison in Section 4.3, we hold the tasks and query-time environments

fixed while allowing the agent's retrieval and action trajectories to change. Paired pre-τ reference recall uses the same 426 eligible tasks. Complete-coverage transitions are paired over all 453 tasks. Table 14 summarizes both comparisons.

Table 14: Paired memory-checkpoint diagnostics for Qwen3.8-Max-0902. Pre-τ reference recall uses the same 426 tasks and historical references at both checkpoints. Newly memory-covered tasks attain complete memory coverage at M_T but not at M_τ. The final column reports how many of these tasks already had joint complete coverage under $M_\tau + E_T$.

Backend	**Pre-τ reference recall (%)**	**Newly memory covered**	**Already jointly covered at M_τ**
NaiveRAG	64.9 $\rightarrow$ 61.9	89	71 (79.8%)
A-MEM	65.1 $\rightarrow$ 63.0	93	81 (87.1%)

The paired executions show two patterns. Pre-τ reference recall decreases from 64.9% to 61.9% with NaiveRAG and from 65.1% to 63.0% with A-MEM. At the same time, M_T provides complete memory-only coverage for additional tasks. Most of these tasks were already jointly covered by M_τ and E_T, including 71 of 89 tasks for NaiveRAG and 81 of 93 for A-MEM. Thus, much of the new memory-only coverage overlaps with evidence already returned through the combination of historical memory and application retrieval. Extending memory increases memory-only coverage while reducing recall of the fixed historical references in these runs, helping interpret the mixed end-to-end results in Section 4.3.

E.2 Representative Cases from the Main Experiments

Three cases illustrate concrete behaviors behind the experimental findings in Sections 4.2–4.4. Each presents the original user request, the observed behavior under the compared conditions, and the mechanism illustrated by that comparison.

E.2.1 Historical Evidence across Memory Backends

Request.
"For the office-supply item now listed in a 12-pack configuration, what quantity did I initially request before the stock checks and later revisions?"

Observed behavior. The source ledger records 100 boxes of black gel pens, while a later conversation requests 60 boxes. With M_τ, NaiveRAG retrieves the original ledger and LangMem retrieves a consolidated memory preserving the 100-box quantity. Both runs succeed. Mem0's retrievals do not return the original quantity. Application search returns the later 60-box request, which the agent submits, failing the task. A targeted textual inspection of Mem0's export also finds no matching item–quantity fact.

What it shows. This case illustrates the backend differences in Section 4.2. The returned evidence connects the successful runs to the requested original quantity, while the failed run uses a later revision. Different memory backends can therefore expose different historical evidence to the agent for the same task.

E.2.2 Temporal Version Selection

Request.
"For the calendar entry that now says it was pushed back due to maintenance, what was its start time before that change?"

Observed behavior. Both checkpoints return the requested historical value, 20:00. The M_τ run submits 20:00 and succeeds. The M_T run additionally retrieves the later 21:00 schedule and submits that value instead, failing the task.

What it shows. This case illustrates one way that extending memory can fail to improve task success, as observed in Section 4.3. The requested historical value is retrieved at both checkpoints. The additional record introduces a competing version, and the M_T run selects that later state instead of the one requested by the user.

E.2.3 EXECUTING THE INTENDED COMMITMENT

Request.
"Find the mechanic who gave me the chain-cleaning advice, then message him to confirm that I'll follow his current bike handoff routine and notify him as requested."

Observed behavior. The M_τ run succeeds by promising to notify the mechanic after the bike has been left at the main South Gate security desk. The M_T run instead sends "I'm texting you now to let you know it's there." The message passes the authorization and side-effect checks but fails the requested outcome because it states that the handoff has already occurred.

What it shows. This case illustrates the gap between evidence retrieval and task completion in Section 4.4. The executed message changes a future notification into an assertion about a completed event. The error lies in realizing the user's intended commitment through the message sent to the mechanic.

F INTERACTION COST ANALYSIS

We provide supplementary interaction-cost statistics for Qwen3.8-Max-0902 under the access conditions analyzed in Section 4.3. A step corresponds to one agent decision and may contain multiple read-only tool calls or one GUI action. Input and output tokens sum the saved per-step API usage, including repeated conversation context. Memory construction, backend-internal model calls, and evaluation are excluded from these costs.

F.1 COST ON SHARED-SUCCESS TASKS

Table 15 compares interaction cost on tasks successfully completed under both conditions in each pair. Each row therefore uses its own paired subset.

Table 15: Mean interaction cost on tasks successfully completed under both compared conditions. Values follow the order From → To. N is the number of paired tasks, and k denotes 1,000 tokens.

From	To	N	Steps	Input (k)	Output (k)
Access conditions					
GUI Only	GUI + App Search	33	25.76→11.64	166.10→260.50	22.25→13.85
GUI + App Search	NaiveRAG M_τ	133	15.50→13.23	459.80→249.44	33.92→15.77
GUI + App Search	A-MEM M_τ	133	15.88→12.93	467.72→299.61	37.49→17.20
Memory checkpoints					
NaiveRAG M_τ	NaiveRAG M_T	297	15.40→15.31	391.84→380.72	26.75→25.30
A-MEM M_τ	A-MEM M_T	302	14.49→14.36	402.00→396.00	22.66→23.06

Adding App Search reduces mean interaction steps on the shared GUI-only successes from 25.76 to 11.64. Input tokens increase from 166.10k to 260.50k. Thus, fewer agent decisions do not necessarily imply lower model-input cost. On their respective shared-success subsets, adding either NaiveRAG or A-MEM M_τ reduces both interaction steps and token usage relative to GUI + App Search. By contrast, extending M_τ to M_T produces only small changes in interaction cost for tasks that succeed under both checkpoints.

F.2 COST OF FAILED EXECUTIONS

Table 16 summarizes the interaction length of failed executions. These comparisons are descriptive rather than paired: the set of failed tasks differs across access conditions.

Budget exhaustion dominates GUI-only failures, accounting for 90.9% of them, and falls to 48.9% with App Search and 24.6–32.5% under the memory conditions. The remaining failures include completion without satisfying the task rubric, explicit aborts, and output-format errors. Because different conditions fail on different task subsets, these results describe observed stopping behavior rather than a controlled efficiency comparison.

Table 16: Interaction steps used by failed Qwen3.8-Max-0902 executions. Budget exhaustion is the fraction of failures terminating with `MAX_STEPS` under the 50-step limit.

Condition	Failures	Mean steps	Median steps	Budget exhaustion (%)
GUI Only	419	48.46	50.00	90.9 (381/419)
GUI + App Search	313	37.25	47.00	48.9 (153/313)
NaiveRAG M_τ	120	31.38	29.00	32.5 (39/120)
NaiveRAG M_T	134	29.64	25.50	24.6 (33/134)
A-MEM M_τ	122	29.93	25.00	28.7 (35/122)
A-MEM M_T	124	27.69	22.50	26.6 (33/124)

G Human Review and Judge Agreement

Four annotators who were not paper authors independently reviewed the 453 Qwen3.8-Max-0902–NaiveRAG $M_T + E_T$ executions. Each execution was assigned to two annotators. All final ratings were definite pass or fail labels. Human consensus requires matching labels from both annotators. Table 17 summarizes agreement using the final ratings.

Table 17: Final human agreement and agreement between the judge and human consensus.

Metric	Result
Reviewed tasks	453, two annotators per task
Human–human agreement	435/453 (96.03%)
Nominal Krippendorff's α	0.903
Judge–consensus agreement	432/435 (99.31%)

The annotators reached consensus on 435 tasks, with disagreements remaining on 18 tasks. The judge agreed with 432 consensus verdicts. The three discrepancies concern timestamp precision and the range of accepted semantic equivalence. One task was accepted by both annotators and rejected by the judge. Two tasks were rejected by both annotators and accepted by the judge. These results show high agreement between automatic judgments and human consensus on the reviewed executions, with a small number of disagreements on borderline decisions.

H Limitations

R^2-MobileMemory contains ten simulated Chinese and English user worlds within MobileGym. Although the worlds span long histories, heterogeneous applications, and multimodal evidence, their records and task distribution remain products of the controlled construction process. Performance on these environments therefore does not directly establish generalization to real users or physical devices.

Our memory-backend comparison uses a shared textual projection of application records and reviewed visual facts so that heterogeneous memory systems receive comparable inputs. Agents can still inspect screenshots and application images during execution, but the benchmark does not compare fully native multimodal memory architectures under their individual input representations.

Finally, the main protocol forms memory from a fixed ordered record stream and keeps memory read-only during evaluation. This design isolates memory formation, retrieval, evidence selection, and downstream use under controlled inputs. It does not evaluate autonomous decisions about which experiences to store or modify during task execution.