PAST-BENCH: BENCHMARKING THE FOUNDATIONS OF RECURSIVE SELF-IMPROVEMENT IN PERSONAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improvement requires agents to turn accumulated experience into better future behavior. Personal AI agents retain memories, skills, and session histories across sessions on the premise that retained experience makes them better over time. Whether it does has not been tested under controlled conditions. We introduce **PAST-Bench**, a benchmark that isolates the contribution of retained experience to later-task performance. Each agent runs through ordered sequences of fresh-session tasks under matched conditions that turn retained experience on and off, across **26 scenarios and 204 episodes** spanning memory, procedural reuse, information gathering, and update. For each we report the later-task gain and whether that gain flowed through the intended save, retrieve, and update pathway. Across seven base models and four agent frameworks, retained experience helps, but the gain concentrates in different capabilities for different models, and frameworks with the same headline gain can differ substantially in whether that gain is supported by evidence of the intended pathway. Using the benchmark as a diagnostic, we develop **Hermes+**, which adds five targeted interventions to Hermes, one per stage of the agent loop. Each intervention improves its target capability in isolation, yet they do not compose: the full system doubles the gain on tasks that require replacing outdated state, regresses on procedural reuse, and leaves the overall gain within run-to-run variation of the baseline. Improvement from experience is not a single capability, and a benchmark that reports one number cannot tell apart the ways an agent can appear to learn. **PAST-Bench** and **Hermes+** together provide an evaluation and diagnostic foundation for studying how persistent agents progress from retaining experience to improving through it.

1 INTRODUCTION

Recursive self-improvement (RSI) concerns the ability of an AI system to use experience generated through its own operation to improve its future capabilities (Wang et al., 2026b; Ren et al., 2026; Lee et al., 2026; Qu et al., 2024; Yin et al., 2025). Self-modifying agent code is one route (Zhang et al., 2025a); here we study the persistent-state layer already deployed in personal AI agents (Wang et al., 2026b; Gao et al., 2025; Sarukkai et al., 2025). These agents persist across sessions: they read messages, operate over user workspaces, call tools, and accumulate files, memories, skills, and session histories over days and months (OpenClaw, 2026; Nous Research, 2026). Frameworks such as Hermes (Nous Research, 2026) and OpenClaw (OpenClaw, 2026) treat these persistence surfaces as first-class runtime components, and memory-layer systems such as Mem0 (Chhikara et al., 2025) and LangGraph (LangChain, 2026) supply the substrate. The premise behind this infrastructure is that an agent which retains experience will serve the same user better over time, and that this loop is the first thing an agent must close before it can improve anything else: identify useful experience, preserve it beyond the current session, retrieve it when relevant, apply it correctly, and revise it when it becomes stale (Xu et al., 2025; Suresh et al., 2026). We call this capability *online self-evolution*: a change in future behavior driven by experience from prior interactions, without retraining (Xia et al., 2026; Buening et al., 2026), prompt optimization (Ou et al., 2025; Khattab et al., 2024; Yuksekgonul et al., 2025; Ding et al., 2026a;b; Agrawal et al., 2025), or long-context adaptation (Agarwal et al., 2024).

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.

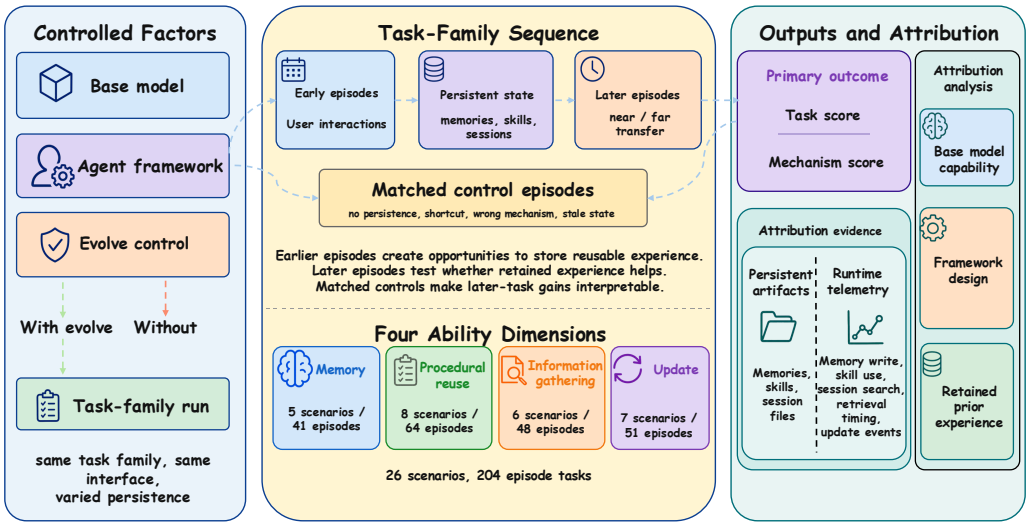

Figure 1: Overview of PAST-BENCH. Left: the three factors a run controls (base model, framework, persistence on or off). Center: a task family, where early episodes create the opportunity to store experience, later episodes test its reuse, and matched control episodes strip persistence. Right: the two reported quantities, task score and mechanism evidence, and the attribution each supports.

It is not RSI in full, but it is the substrate on which stronger forms would have to be built (Zhang et al., 2026a; Fang et al., 2025).

On one hand, the premise is plausible and the infrastructure to realize it is widely deployed. On the other hand, accumulating experience does not guarantee using it well: an agent may store the wrong evidence, retrieve an irrelevant memory, reuse a brittle procedure, or apply state the user has already corrected. And if later-session performance does improve, the improvement might come from the retained experience, or from the base model, the runtime, the prompt, a retrieval shortcut, task difficulty, or scoring noise. It is thus natural to ask:

When a persistent agent performs better on later tasks, is that because of what it retained, and did the gain flow through the mechanism the framework claims credit for?

Existing evaluations address different parts of this question. Interactive benchmarks measure task completion (Liu et al., 2024; Koh et al., 2024; Drouin et al., 2024; Xie et al., 2024; Merrill et al., 2026; Mialon et al., 2024; Zhang et al., 2026b); memory and skill benchmarks test recall, reuse, and adaptation (Wu et al., 2025; Maharana et al., 2024; Li et al., 2026a; Zhang et al., 2026c). Recent work also controls experience histories and verifies task dependence on memory (Yang et al., 2026a; Rathi et al., 2026). Our focus is the joint evaluation of four persistence capabilities under matched fresh-session controls, with mechanism evidence alongside outcome gains.

Contributions. We develop a benchmark whose unit of evaluation is an agent's trajectory through a family of related tasks, rather than a one-shot task score (Figure 1). Earlier episodes in a family give the agent an opportunity to store reusable experience; later episodes test whether that experience is used; and each later episode is graded under a matched control in which persistence is turned off. Holding the model, the task family, and the evaluation interface fixed while toggling persistence makes the resulting gap directly interpretable, and saved artifacts together with execution traces reveal whether the agent wrote, retrieved, applied, or revised the state it was supposed to.

(1) Isolating retained experience (Section 3). PAST-BENCH contains 26 task families and 204 episodes spanning four capabilities that place distinct demands on persistent state: memory, procedural reuse, information gathering, and update. Every family includes control episodes that rule out prompt

shortcuts, surface memorization, stale reuse and writes to the wrong substrate, so a persistence gap is accepted as evidence of self-evolution only when it clears the bound these controls set.

(2) Persistence gains are real but not one thing (Section 4). Across seven base models under a fixed framework, we find that every model gains from persistence, with overall gaps from $+0.13$ to $+0.24$. However, where the gain concentrates varies sharply by model: GPT-5.4 spreads it across memory and update, GLM-5.1 places nearly half of its movement on update and Kimi K2.6 nearly half on memory. Across four frameworks under a fixed model, we further find that frameworks with identical overall gaps can differ substantially in mechanism evidence. Hermes and nanobot both reach $\Delta = +0.13$, yet nanobot earns it from a single capability with no consistent write-then-read trace. A single number can hide both distinctions.

(3) Mechanisms that help alone do not compose (Sections 4.3 and 4.4). Using the benchmark as a diagnostic, we trace the low-gain cases to five failures, each owned by one stage of the agent loop, and add one targeted intervention per stage to Hermes. In isolation, each intervention delivers the largest single-mechanism gain on its target capability. Enabled together, they double the update gain ($+0.12$ to $+0.24$) and leave the overall gain within run-to-run variation of the baseline, while procedural reuse regresses. A full-minus-one ablation traces the regression to an interaction between the memory-rendering and skill-routing interventions: removing the former raises the procedural gap. This indicates that persistence mechanisms should not be viewed as independently composable, and that a framework can improve on the capability it was designed for while losing ground elsewhere.

2 Related Work

Related work spans task evaluation, persistent memory, and reusable skills. Table 1 compares representative benchmarks; Appendix F expands the comparisons, including recent lifelong-agent evaluations.

Task and trajectory evaluation. Interactive benchmarks (Liu et al., 2024; Koh et al., 2024; Drouin et al., 2024; Xie et al., 2024; Merrill et al., 2026) measure task completion, while AgentBoard, TRAJECT-Bench, and ATBench examine within-task trajectories (Ma et al., 2024; He et al., 2026; Li et al., 2026b). PAST-BENCH combines later-task outcomes with evidence of cross-session state use. Benchmark-validity studies further motivate separating successful execution from the mechanism credited for it (Liang et al., 2026; Wang et al., 2026a).

Memory and lifelong evaluation. LongMemEval and LoCoMo evaluate conversational recall (Wu et al., 2025; Maharana et al., 2024); AgentMemoryBench and PATH-Bench extend evaluation to transfer, forgetting, and memory coordination (Ma et al., 2026; Yang et al., 2026a). PASB traces harmful commitments across cleared sessions (Mao et al., 2026), while DolphinBench verifies memory-dependent actions with and without relevant history (Rathi et al., 2026). These are complementary to our four-capability protocol, which pairs fresh-session persistence controls with mechanism evidence under fixed-model and fixed-framework comparisons.

Skill learning and agent adaptation. Reflexion and ExpeL retain feedback and extracted experience (Shinn et al., 2023; Zhao et al., 2024); Voyager and Agent Workflow Memory reuse executable skills and workflows (Wang et al., 2023; 2024). SkillsBench, SkillLearnBench, and SkillFlow evaluate skill use and evolution (Li et al., 2026a; Zhong et al., 2026; Zhang et al., 2026c), while SkillOpt and SkillOS improve skill execution and curation (Yang et al., 2026b; Ouyang et al., 2026). PAST-BENCH studies how memory, procedural reuse, retrieval, and update interact within the same runtime, complementing architectural comparisons (Bogavelli et al., 2025).

3 PAST-Bench

Persistence-aware evaluations use conversational histories (Maharana et al., 2024; Letta, 2025; Hu et al., 2026a), sequential task streams (Wei et al., 2025; Zheng et al., 2025), and controlled experience paths (Yang et al., 2026a). Related protocols also clear sessions or verify dependence on prior history (Mao et al., 2026; Rathi et al., 2026). PAST-BENCH evaluates an ordered task family whose episodes share a latent rule, reusable artifact, correction, or pre-seeded reference. We wipe volatile

Table 1: Comparison with representative benchmarks. PAST-BENCH is designed for retained-experience attribution: it evaluates longitudinal task families and supports both model-side and framework-side isolation. ✓: axis directly supported. ✗: not supported. △: related proxy tested, not the matched retained-experience comparison.

Benchmark	Retained experience	Model comparison	Framework comparison	Trajectory diagnosis
GAIA (Mialon et al., 2024)	✗	✗	✗	✗
AGENTBENCH (Liu et al., 2024)	✗	✓	✗	✗
VISUALWEBARENA (Koh et al., 2024)	✗	✓	✗	✗
WORKARENA (Drouin et al., 2024)	✗	✓	✗	✗
OSWORLD (Xie et al., 2024)	✗	✓	✗	✗
LONGMEMEVAL (Wu et al., 2025)	✓	✓	✗	✗
LOCOMO (Maharana et al., 2024)	✓	✓	✗	✗
SKILLSBENCH (Li et al., 2026a)	△	✓	△	△
AGENTBOARD (Ma et al., 2024)	✗	✓	✗	✓
PAST-BENCH	✓	✓	✓	✓

context between episodes and pair evaluation outcomes with persistence-off controls and mechanism evidence. This separates transfer through the persistent substrate from residual conversation context across four capability categories.

3.1 BENCHMARK CONSTRUCTION

PAST-BENCH targets four core **capabilities** of online self-evolution, each requiring retention across sessions and active reuse in later ones: **Memory**, **Procedural Reuse**, **Information Gathering**, and **Update**. For each capability, we curate a diverse set of carefully designed *task families*. Each task family is an ordered sequence of fresh-session episodes sharing a latent rule (**memory**), a reusable artifact (**procedural reuse**), a correction (**update**), or a pre-established reference whose retrieval must be triggered at the right moment (**information gathering**). For the first three capabilities, earlier episodes expose information, a procedure, or an updated value to be retained; for information gathering, the reference is preseeded before the family begins. In every case, later episodes test whether the agent reuses or consults the relevant state without restating the decisive rule. Each family also contains control episodes: *no-retention* controls that remove the earlier state, *distractor* controls that inject irrelevant or superficially similar information, *stale* controls that expose obsolete memories, and *wrong-mechanism* controls that surface incorrect skills or the wrong evidence source. The distribution of families across capabilities and scenario groups is shown in Fig. 2. The per-capability breakdown is reported in Table 5 in Appendix A.1. Appendix A.3 presents detailed trace-backed examples in Figures 4–8.

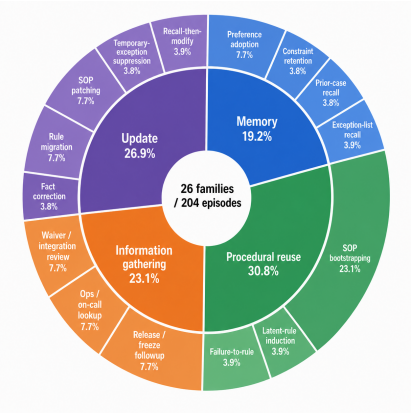

Figure 2: **Task family distribution of PAST-BENCH** across the four capability categories.

Capability 1: General-knowledge memory—can the framework retain and look up an everyday clause? Memory families isolate the *declarative* pathway (Sumers et al., 2024; Squire, 1992) for casual, user-facing facts (preferences, constraints, one-line policies, exceptions, prior-case decisions). Each family seeds a single read-mostly clause and never revises it; evaluate episodes succeed iff the agent recovers and applies it as a *one-shot lookup-and-apply*, with the trigger wording removed from the prompt. *When* to query is deferred to Capability 3; *revising* a clause is deferred to Capability 4.

Capability 2: Professional procedural reuse—can the framework retain and re-execute a multi-step technical workflow? Procedural-reuse families isolate the *imperative* path-

way (Sumers et al., 2024; Anderson & Lebiere, 2014) for domain-specific, technical routines—SOPs, playbooks, build/deploy pipelines, incident-triage flows, induced engineering workflows (Hong et al., 2024). Where Capability 1 tests *value lookup*, Capability 2 tests *ordered execution* with the right tool composition; order errors, skipped steps, and wrong-tool substitutions are graded as failures. Procedure *revision* is deferred to Capability 4, so this capability isolates first-time procedure formulation.

Capability 3: Information gathering—given that the answer is already in the substrate, does the framework consult it at the right moment? The relevant artifact is *pre-seeded* into the runtime's persistence layer (long-term memory, registered skills, indexed transcripts, or home-state fixtures) before the family begins. The test is not retention but whether the agent *proactively retrieves* under noisy context; each family also plants a generic default that yields a plausible-but-wrong answer if used in place of retrieval.

Capability 4: Update—can a second write override a first one without leaking the first? Update families begin with an existing entry (stale fact, old rule, outdated SOP, or temporary exception) and then deliver an authoritative second write; evaluate episodes score whether the new state is used *and* the old state does not leak. Seven families cover fact correction, global and scoped rule migration, temporary-exception expiry, incident and error-correction SOP patching, and recall-then-modify of a prior session artifact. Figures 4–8 in Appendix A.3 give one representative example per capability.

3.2 Evaluation Pipeline

Episode roles. Every task family is an ordered sequence of fresh-session episodes playing one of four roles. **Cold** episodes measure first-contact behavior before any persistence can exist. **Learn** episodes (and, in Update families, an **Update** episode) deposit the target (a clause, procedure, or correction) into the persistence substrate. **Evaluation** episodes probe reuse of that state in a later fresh session with the trigger wording removed. **Control** episodes check that any gain cannot be explained by prompt shortcuts, surface-memorization, stale reuse, or writes to the wrong substrate.

Persistence and the matched ablation. By *persistence* we mean benchmark-managed access to state produced or modified by earlier episodes in the same family: memory records, skills, profile entries, session-history indices, saved artifacts, and home-state fixtures. Each evaluation episode is graded under a matched ablation. The **w/o-evolve** condition denies the runtime any access to family-produced state; the **w/-evolve** condition permits it. The two runs share the same prompt, grader, tool stack, and seed, so any score gap is most plausibly attributable to the persistence layer rather than to model variance or task drift. We treat this as a strong design control rather than a causal proof; the mechanism-evidence score below provides a complementary substrate-level check. Cold scores are reported for calibration and headroom analysis but are *not* the persistence-off baseline; the matched w/o-evolve condition is.

What we report. The primary metric for a family f is the self-evolution gap $\Delta_f = S_f^{\text{w/-evolve}} - S_f^{\text{w/o-evolve}}$, where S_f is the within-family mean of the per-episode task score. The capability-level Δ macro-averages over families. We accept Δ_f as evidence of self-evolution only when it clears the bound set by the family's control episodes. Alongside the gap, we report a mechanism-evidence score (Mech) computed from saved artifacts and runtime telemetry: memory writes and reads, skill creation and patching, session-search calls, artifact diffs, and update-correctness signals (Appendix B.3). An agent that writes artifacts but never reads them, or that succeeds through the wrong substrate, does not earn credit on Δ alone.

Note that Mech is a *consistency* signal rather than a *causal* test. It measures whether the trace matches the expected persistent pathway; it does not establish that the pathway was necessary for the outcome. Appendix B.5 shows that the score is stable under reweighting its components and under removing individual signals, and that across the Hermes traces it is only weakly correlated with Δ itself (Spearman 0.23). The weak correlation is the point: whether a gain occurred and whether it went through the intended mechanism are different questions, and PAST-BENCH reports both. Open-ended episodes are graded by an LLM judge whose scores fall within 0.5 of the mean of two blinded human raters on 91.7% of a 48-sample audit (Appendix B.4).

Table 2: **Overall performance of PAST-BENCH with Hermes agent.** w/o evolve and w/ evolve denote persistence off and on; Δ is their family-balanced difference. Results are reported in average across 3 runs. Subscripts on capability Δ cells give each capability's signed share of the row's total absolute movement $\sum_c |\Delta_c|$ (per-row absolute values sum to 100%); the Overall Δ column is the macro-average and carries no subscript. Mech. is the mechanism-evidence score when applicable.

Model	State	Memory	Procedural	Info.	Update	Overall	Mech.
GLM-5.1	w/o evolve	0.51	0.44	0.59	0.53	0.52	–
	w/ evolve	0.74	0.53	0.69	0.90	0.71	0.70
	Δ	$+0.23_{(29\%)}$	$+0.09_{(11\%)}$	$\mathbf{+0.11}_{(14\%)}$	$\mathbf{+0.36}_{(46\%)}$	$\underline{+0.20}$	–
Kimi K2.6	w/o evolve	0.51	0.60	0.65	0.50	0.57	–
	w/ evolve	0.83	0.63	0.70	0.78	0.73	0.72
	Δ	$\underline{+0.33}_{(49\%)}$	$+0.03_{(4\%)}$	$+0.05_{(7\%)}$	$+0.27_{(40\%)}$	$+0.17$	–
DeepSeek-V4-Pro	w/o evolve	0.46	0.42	0.60	0.51	0.50	–
	w/ evolve	0.79	0.54	0.66	0.69	0.67	0.71
	Δ	$\underline{+0.33}_{(48\%)}$	$+0.12_{(17\%)}$	$+0.06_{(9\%)}$	$+0.18_{(26\%)}$	$+0.17$	–
MiniMax-M2.7	w/o evolve	0.50	0.50	0.62	0.50	0.53	–
	w/ evolve	0.77	0.55	0.71	0.62	0.66	0.64
	Δ	$+0.26_{(50\%)}$	$+0.05_{(10\%)}$	$+0.09_{(17\%)}$	$+0.12_{(23\%)}$	$+0.13$	–
GPT-5.4	w/o evolve	0.48	0.49	0.62	0.46	0.51	–
	w/ evolve	0.85	0.68	0.69	0.80	0.75	**0.80**
	Δ	$\mathbf{+0.37}_{(38\%)}$	$\mathbf{+0.19}_{(20\%)}$	$+0.07_{(7\%)}$	$\underline{+0.34}_{(35\%)}$	$\mathbf{+0.24}$	–
Claude Sonnet 4.6	w/o evolve	0.51	0.43	0.59	0.48	0.50	–
	w/ evolve	0.83	0.53	0.69	0.77	0.71	$\underline{0.76}$
	Δ	$+0.32_{(40\%)}$	$+0.10_{(12\%)}$	$\underline{+0.10}_{(12\%)}$	$+0.29_{(36\%)}$	$\underline{+0.20}$	–
Claude Opus 4.6	w/o evolve	0.32	0.50	0.62	0.50	0.49	–
	w/ evolve	0.58	0.64	0.68	0.78	0.67	0.70
	Δ	$+0.26_{(35\%)}$	$\underline{+0.14}_{(19\%)}$	$+0.06_{(8\%)}$	$+0.28_{(38\%)}$	$+0.19$	–

4 EXPERIMENTS

We report results across models and frameworks (Sec. 4.2), use them to diagnose where retained experience fails to help and test whether the interventions work alone, together, and across models (Sec. 4.4).

4.1 EXPERIMENTAL SETUP

Each task family is evaluated under paired conditions: persistence-off, with no retained state from the learning episodes, and persistence-on, where the agent may reuse memory, skills profile state, or session history. We report the persistence-on score and the family-balanced gap Δ. Mech supports attribution (Section 3.2). [1]

4.2 MAIN RESULTS

Every model gains from persistence, but where it gains depends on the model. Table 2 fixes the framework to Hermes and varies the base model. Overall Δ is positive for all seven models, ranging from $+0.13$ to $+0.24$, so the runtime carries persistence value across architectures. The composition of that gain is not stable. The capability subscripts in Table 2 give each capability's share of a model's total movement: GPT-5.4 divides its gain almost evenly between memory (38%) and update (35%); GLM-5.1 places 46% of its movement on update; Kimi K2.6 places 49% on memory. The capability

[1] The model comparison holds Hermes, tasks, graders, and limits fixed. The framework comparison holds Minimax-M2.7, tasks, and graders fixed. Each persistence-on/off pair holds everything fixed except access to retained state. Appendix E.1 lists what each comparison leaves native, and Appendix D.7 reports per-episode cost.

Table 3: **Fixed-model agent comparison on the four PAST-BENCH capabilities.** MiniMax-M2.7 is held fixed and agent frameworks vary. Capability columns report persistence-on/off deltas; Mech. summarizes whether improved outcomes align with expected artifacts or telemetry. Subscripts on capability Δ cells give each capability's signed share of $\sum_c |\Delta_c|$ (per-row absolute values sum to 100%); the Overall Δ column is the macro-average and carries no subscript. Sources for Agent-Zero, nanobot, and ZeroClaw are (HKUDS, 2026; ZeroClaw Labs, 2026; Agent Zero Contributors, 2026).

Agent	Memory Δ	Procedural Δ	Info. Δ	Update Δ	Overall Δ	Mech.
nanobot	+0.06 (10%)	−0.06 (−10%)	**+0.13** (21%)	**+0.37** (59%)	+0.13	0.57
ZeroClaw	**+0.29** (52%)	−0.04 (−7%)	+0.08 (14%)	+0.15 (27%)	+0.12	0.55
Agent-Zero	−0.27 (−52%)	**+0.11** (21%)	−0.01 (−2%)	−0.13 (−25%)	−0.08	0.39
Hermes	+0.26 (50%)	+0.05 (10%)	+0.09 (17%)	+0.12 (23%)	+0.13	0.64
HERMES+ (Our framework)	+0.27 (42%)	−0.02 (−3%)	+0.12 (18%)	+0.24 (37%)	**+0.15**	**0.73**

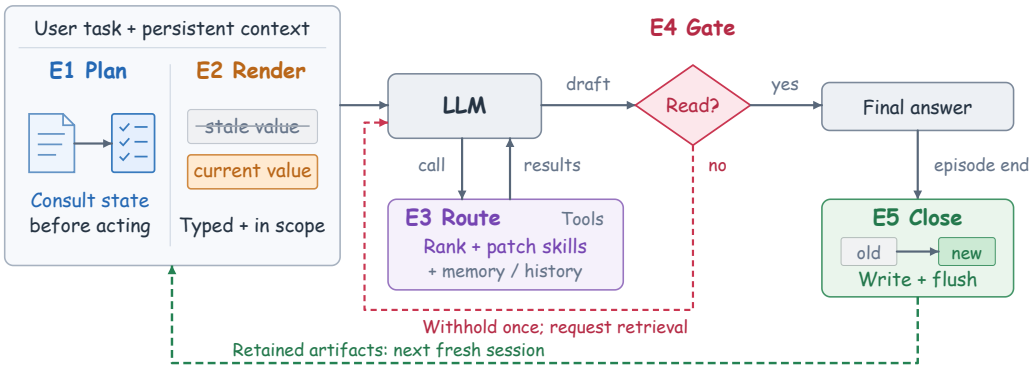

Figure 3: Five independently ablatable interventions in HERMES+: E1 consults state before planning; E2 renders valid typed bindings; E3 ranks and patches skills; E4 withholds a draft once when required retrieval is missing; E5 writes and flushes corrections. Gray nodes retain the Hermes loop. Dashed paths show the retrieval reminder and cross-session persistence.

a model already handles well is also the one where retained experience helps it most. This suggests that a single overall Δ does not summarize a model's ability to learn from experience so much as average over four abilities that vary independently.

The same gain can come from different places. Table 3 fixes the model to MiniMax-M2.7 and varies the framework. ZeroClaw lifts memory (52% of its movement) but loses ground on procedural reuse; nanobot puts 59% of its movement on update and barely moves memory. Agent-Zero regresses on three of four capabilities. Hermes is the only baseline that moves all four capabilities upward. Crucially, Hermes and nanobot reach the same overall Δ of +0.13, yet their mechanism evidence differs (0.64 versus 0.57): nanobot's gain comes from one capability and its traces show no consistent write-then-read pattern. Figure 10 plots the two axes against each other. Two frameworks that tie on the outcome axis can sit apart on the mechanism axis, which is why the benchmark reports both.

4.3 DIAGNOSIS-DRIVEN DESIGN: HERMES+

Traces from Tables 2 and 3 reveal five recurring failures: plans ignore saved state, stored facts are difficult to reuse, learned procedures never enter the skill library, actions precede evidence retrieval, and corrections fail to reach the next session. Each failure points to a different stage of the agent loop. We add one independently ablatable intervention per stage to Hermes (Figure 3) and test whether each addresses its target failure and whether the interventions compose. This turns PAST-BENCH from a measurement into a diagnostic.

We choose Hermes because its exposed agent loop allows one stage to change while the others remain at their defaults. The other frameworks already contain persistence machinery that overlaps with

these interventions (Appendix C.1), complicating a clean ablation. Hermes is also the strongest controlled baseline in Table 3: it has positive Δ on all four capabilities and the highest mechanism evidence among the baselines. This choice keeps the model, tasks, grader, and persistence surfaces fixed while isolating runtime decisions.

Plan-time consultation (E1; cross-cutting). Across all four capabilities, an agent may draft an action before consulting correctly stored, retrievable evidence. Such an unconditioned plan can override a saved binding, skill, or rule regardless of how that state is represented. E1 adds a consultation check before risky or recall-dependent actions: the agent must inspect the persistent state exposed by the runtime and condition its plan on that evidence. It reads E2's typed schema when E2 is active and native Hermes memory records otherwise. E1 operates independently of the other mechanisms. Its role is to bring available state into planning before the ability-specific decisions below.

Typed memory bindings (E2; Memory). Memory tasks require an agent to recover preferences, constraints, or corrected values in a fresh session. In Hermes, current and stale notes can coexist, scope is implicit, and no clear valid clause may be available to apply. E2 stores each memory with its type, scope, entity, current value, superseded value, and expiry. At render time, it surfaces only the current, in-scope binding and suppresses superseded entries. The resulting context makes the applicable fact explicit instead of leaving competing notes for the next session to resolve.

Ranked, patchable skills (E3; Procedural). Procedural tasks require reopening a saved workflow and executing its ordered steps. Hermes traces include successful learning episodes whose SOP remains in transcript text or is fragmented into near-duplicate notes; later sessions then have no ranked skill to open. E3 saves procedures with applicability conditions and ordered steps, ranks skills by query relevance, and directs the model to patch the closest existing skill when a workflow changes. These decisions connect a solved task to a reusable procedure while limiting duplicate versions in the skill library.

Retrieval-before-answer gate (E4; Information Gathering). These tasks preseed the required evidence in memory or session history. Under noisy prompts, Hermes may answer from visible context, refuse to guess, or act before accessing that evidence. E4 checks recall-dependent draft answers for a required persistence read. If the read is missing, it withholds the draft once and injects a retrieval request into the loop. This targets retrieval timing even when the relevant artifact already exists; the agent must consult the appropriate channel before finalizing its response.

Authoritative closeout (E5; Update). Update tasks supply a second authoritative value and test its use in a fresh session. Hermes may leave a correction in the current transcript while the next episode retrieves an older artifact. E5 extracts the final binding key or updated rule at episode close, writes the new authoritative artifact in place of the prior value, and flushes it synchronously to the persistent store. This makes the correction available to the next session without retaining the superseded artifact alongside it. Figures 4–8 in Appendix A.3 provide trace-backed examples of the five interventions.

4.4 Mechanism Evidence and Generalization

Single-mechanism effects align with their targets. Table 4 adds one mechanism at a time to base Hermes. E2 yields the highest Memory persistence-on score (0.80); E3, E4, and E5 yield the largest single-mechanism gaps on Procedural ($+0.10$), Information Gathering ($+0.17$), and Update ($+0.16$), respectively. These outcomes support the failure-to-mechanism mapping in Section 4.3.

The mechanisms do not compose. Full HERMES+ matches base Hermes on overall persistence-on score (0.66), changes overall Δ from $+0.13$ to $+0.15$, and doubles the Update gap to $+0.24$. Procedural reuse regresses: its score falls from 0.55 to 0.38 and its gap from $+0.05$ to -0.02. Across three runs, the Update gap is $+0.12 \pm 0.01$ for Hermes and $+0.24 \pm 0.09$ for HERMES+ (Table 11). The larger mean therefore accompanies substantially greater variability. The $+0.02$ overall difference is smaller than the run-to-run variation, so it should not be read as a stable overall improvement. The clearest effect is capability-specific: stronger Update with weaker Procedural performance.

The regression reflects mechanism interaction. The six-family diagnostic in Appendix D.2 (Table 8) removes one mechanism at a time from the full system. Removing E3 or E5 lowers the Procedural gap to $+0.062$ or $+0.042$; removing E2 raises it from $+0.085$ to $+0.108$. Routing traces suggest why: E2 can direct a learned procedure into typed memory, leaving no skill for E3 to

Table 4: Single-mechanism ablations on PAST-BENCH (MiniMax-M2.7, fixed task set and grader). **Each non-final row adds one mechanism to Base Hermes; rows are *not* cumulative.** The bottom row, HERMES+ (full), turns on all five. Per capability: w/ is the persistence-on score, Δ the family-balanced persistence-on/off gap. Subscripts give each capability's signed share of $\sum_c |\Delta_c|$ (per-row absolute values sum to 100%); Overall Δ is the macro-average and carries no subscript. Best values are **bold**; second-best, underlined.

	Memory		Procedural		Info.		Update		Overall	
Setting	w/	Δ	w/	Δ	w/	Δ	w/	Δ	w/	Δ
Base Hermes	0.77	+0.26 (50%)	**0.55**	+0.05 (10%)	0.71	+0.09 (17%)	0.62	+0.12 (23%)	**0.66**	+0.13
+ planning guidance (E1)	0.73	+0.32 (58%)	0.44	+0.05 (9%)	0.68	+0.05 (9%)	0.62	+0.13 (24%)	0.62	+0.14
+ memory binding (E2)	**0.80**	+0.30 (54%)	0.30	−0.02 (−4%)	0.71	+0.14 (25%)	0.57	+0.10 (18%)	0.60	+0.13
+ skill lifecycle (E3)	0.46	+0.17 (36%)	0.47	**+0.10** (21%)	0.75	+0.15 (32%)	0.35	+0.05 (11%)	0.51	+0.12
+ retrieval gate (E4)	0.63	**+0.36** (54%)	0.43	+0.08 (12%)	**0.78**	**+0.17** (25%)	0.37	+0.06 (9%)	0.55	**+0.17**
+ closeout / flush (E5)	0.49	+0.20 (42%)	0.33	+0.00 (0%)	0.69	+0.12 (25%)	0.70	+0.16 (33%)	0.55	+0.12
HERMES+ (full)	0.78	+0.27 (42%)	0.38	−0.02 (−3%)	0.73	+0.12 (18%)	**0.74**	**+0.24** (37%)	**0.66**	+0.15

retrieve. Mechanisms that help separately can thus compete for the same experience through different persistence surfaces. This is a focused diagnosis on one model and one run, not a benchmark-wide estimate; it motivates treating the choice of storage surface as part of the runtime's decision.

Transfer across models is positive but not uniform. HERMES+, tuned on MiniMax-M2.7, matches or improves Hermes on three of five models: MiniMax-M2.7 ($+0.13$ to $+0.15$), Claude Sonnet 4.6 ($+0.20$ to $+0.22$), and GPT-5.4 (unchanged at $+0.24$). DeepSeek-V4-Pro and Claude Opus 4.6 regress by 0.02 and 0.01 (Appendix D.3, Table 9). With GPT-5.4, HERMES+ ties the strongest benchmark configuration ($\Delta = +0.24$, Mech 0.80). Update gaps remain positive across all five models, while Procedural gaps remain positive for GPT-5.4 and Claude Sonnet 4.6 ($+0.18$ and $+0.17$), unlike MiniMax-M2.7. These differences are consistent with model-dependent responses to persistence guidance. These analyses distinguish isolated effects, interactions, and transfer: a single overall Δ does not establish that an intervention composes with others or generalizes across models.

5 CONCLUSION

We present PAST-BENCH, a benchmark for measuring whether a personal agent improves on later tasks because of experience it retained from earlier ones. PAST-BENCH builds on the observation that a one-shot task score *cannot* separate retained experience from the base model and the runtime, and instead evaluates ordered families of fresh-session tasks under matched persistence-on and persistence-off conditions, reporting mechanism evidence alongside the resulting gap. Experiments across seven base models and four agent frameworks show that retained experience helps, that the gain concentrates in different capabilities for different models, and that agents with the same overall gain can differ substantially in whether that gain flowed through the intended pathway. Using the benchmark as a diagnostic, we add *five* interventions to Hermes, one per stage of the agent loop, and find that each helps on the capability it targets but that the five do not compose: the combined system doubles the update gain, regresses on procedural reuse, and transfers unevenly across base models. We hypothesize that this is a general property of persistence mechanisms rather than of Hermes, since the regression traces to two mechanisms competing for the same piece of experience (Table 8), and that deciding which surface owns a given experience is a problem we leave for furture work.

Here, we discuss several limitations explicitly. First, all task families are synthetic and each runs against its own persistence store, so the benchmark does not measure interference among memories, procedures, and corrections that accumulate when many families share a store over a long horizon. Extending the protocol to that setting is a promising direction. Second, our mechanism-evidence score measures consistency with an expected pathway rather than causal necessity; a counterfactual check that deletes or corrupts a candidate artifact would be stronger, and we leave it to future work.

AI USE STATEMENT

In this work, we used generative AI tools to generate the synthetic benchmark: the 26 task families and 204 episodes, together with their prompts, graders, and expected artifacts, were generated from author-written capability definitions and design rules by two model–agent pairs (Codex with GPT-5.4 and Claude Code with Claude Opus 4.6), then run and checked by all authors against the six criteria in Appendix A.2. We also used coding agents to implement parts of the HERMES+ mechanisms and the framework adapters. Open-ended episodes are graded by an LLM judge (MiniMax-M2.7, temperature 0), whose scores we validated against two blinded human raters on a 48-sample audit (Appendix B.4). We did not use generative AI tools to propose or refine hypotheses, design the evaluation methodology, or interpret results; formulating or proving mathematical claims, translation, and qualitative or thematic analysis are not applicable to this work. Additionally, we used generative AI tools to draft and edit parts of the manuscript for readability and to suggest its structure. We have reviewed all AI-assisted work: every generated task was executed and checked by at least one author before inclusion, all AI-generated code was reviewed and tested by the authors, and all AI-edited text was checked against the reported results by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility Statement. Anonymous code and benchmark are available at `https://anonymous.4open.science/r/past-bench/`. Benchmark construction, episode roles, and the six quality-control checks are in Section 3 and Appendix A.2; every task family, grader and expected artifact is in the supplementary material together with the framework adapters (Appendix C.1). Metric definitions (Equations 1 and 2), aggregation, the 48-sample blinded audit of the LLM judge, and the sensitivity analysis of Mech are documented in Appendix B. Model inference settings, context and persistence handling, agent limits, retry policies, and the treatment of timed-out episodes are in Tables 13 to 15 of Appendix E. Per-family paired results, run-to-run variance for the three-run configurations, and per-episode cost are in Appendix D. Full trajectories for all reported runs will be released with the code: task records, model responses, tool calls and results, persistent-state changes, and score records are kept; provider credentials and authentication secrets are removed.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/hash/8cb564df771e9eacbfe9d72bd46a24a9-Abstract-Conference.html`.

Agent Zero Contributors. Agent Zero: Personal, organic agentic framework, 2026. URL `https://github.com/agent0ai/agent-zero`. Accessed 2026-04-24.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. *arXiv preprint arXiv:2507.19457*, 2025. URL `https://arxiv.org/abs/2507.19457`.

John R Anderson and Christian J Lebiere. *The atomic components of thought*. Psychology Press, 2014. doi: 10.4324/9781315805696. URL `https://doi.org/10.4324/9781315805696`.

Tara Bogavelli, Roshnee Sharma, and Hari Subramani. AgentArch: A comprehensive benchmark to evaluate agent architectures in enterprise. *arXiv preprint arXiv:2509.10769*, 2025. URL `https://arxiv.org/abs/2509.10769`.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions. In *ICML 2026 Workshop on Continual Adaptation at Scale: Towards Sustainable AI*, 2026. URL `https://openreview.net/forum?id=HL3iloEczS`. Oral.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI 2025*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Jayanaka Dantanarayana, Savini Kashmira, Lingjia Tang, and Jason Mars. SIGIL: Compiling Agent Skills into Typed Harnesses. *arXiv preprint arXiv:2607.27309*, 2026. URL https://arxiv.org/abs/2607.27309.

Giuseppe Destefanis, Daniel Graziotin, Matteo Vaccargiu, and Marco Ortu. GitSkills: A Dataset of Agent Skills on GitHub. *arXiv preprint arXiv:2608.10906*, 2026. URL https://arxiv.org/abs/2608.10906.

Rui Ding, Zixin Ding, Rodrigo Pires Ferreira, Yuxin Chen, and Junhong Chen. Text-twin-translation (T^3): A full-stack machine learning framework for functional material-device systems discovery. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 10831–10841, 2026a. doi: 10.1145/3770855.3819013. URL https://doi.org/10.1145/3770855.3819013.

Zixin Ding, Junyuan Hong, Zhan Shi, Tianhao Wang, Zinan Lin, Li Yin, Meng Liu, Zhangyang Wang, and Yuxin Chen. Scaling textual gradients via sampling-based momentum. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 917–952. ACM, 2026b. doi: 10.1145/3786335.3813168. URL https://doi.org/10.1145/3786335.3813168.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

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

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

Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, Yue Xing, Jiliang Tang, and Benoit Dumoulin. TRAJECT-Bench: A trajectory-aware benchmark for evaluating agentic tool use. In *International Conference on Learning Representations*, volume 2026, pp. 61766–61801, 2026. URL https://arxiv.org/abs/2510.04550.

HKUDS. nanobot: The ultra-lightweight personal AI agent, 2026. URL https://github.com/HKUDS/nanobot. Accessed 2026-04-24.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2507.05257.

Yuyang Hu, Hongjin Qian, Shuting Wang, Jiongnan Liu, Ziliang Zhao, Jiejun Tan, Zheng Liu, and Zhicheng Dou. SAM: State-Adaptive Memory for Long-Horizon Reasoning Agent. *arXiv preprint arXiv:2605.24468*, 2026b. URL https://arxiv.org/abs/2605.24468.

Zisu Huang, Jingwen Xu, Yifan Yang, Ziyang Gong, Qihao Yang, Muzhao Tian, Xiaohua Wang, Changze Lv, Xuemei Gao, Qi Dai, et al. From raw experience to skill consumption: A systematic study of model-generated agent skills. *arXiv preprint arXiv:2605.23899*, 2026. URL https://arxiv.org/abs/2605.23899.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

LangChain. Deep Agents memory documentation, 2026. URL https://docs.langchain.com/oss/python/deepagents/memory. Accessed 2026-04-24.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement. *arXiv preprint arXiv:2607.15524*, 2026. URL https://arxiv.org/abs/2607.15524.

Letta. Benchmarking AI agent memory: Is a filesystem all you need?, August 2025. URL https://www.letta.com/blog/benchmarking-ai-agent-memory/. Accessed 2026-04-30.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026a. URL https://arxiv.org/abs/2602.12670.

Yu Li, Haoyu Luo, Yuejin Xie, Yuqian Fu, Zhonghao Yang, Shuai Shao, Qihan Ren, Wanying Qu, Yanwei Fu, Yujiu Yang, Jing Shao, Xia Hu, and Dongrui Liu. ATBench: A diverse and realistic agent trajectory benchmark for safety evaluation and diagnosis. *arXiv preprint arXiv:2604.02022*, 2026b. URL https://arxiv.org/abs/2604.02022.

Zhuofeng Li, Haoxiang Zhang, Cong Wei, Pan Lu, Ping Nie, Yi Lu, Yuyang Bai, Shangbin Feng, Hangxiao Zhu, Ming Zhong, et al. Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction. *arXiv preprint arXiv:2605.05242*, 2026c. URL https://arxiv.org/abs/2605.05242.

Shanchao Liang, Spandan Garg, and Roshanak Zilouchian Moghaddam. The SWE-Bench illusion: When state-of-the-art LLMs remember instead of reason. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: Software Engineering in Practice*, pp. 395–405, 2026. doi: 10.1145/3786583.3786882. URL https://arxiv.org/abs/2506.12286.

Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. MUSE-Autoskill: Self-evolving agents via skill creation, memory, management, and evaluation. *arXiv preprint arXiv:2605.27366*, 2026. URL https://arxiv.org/abs/2605.27366.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046,

2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From Storage to Experience: A Survey on the Evolution of LLM Agent Memory Mechanisms. *arXiv preprint arXiv:2605.06716*, 2026. URL https://arxiv.org/abs/2605.06716.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. *Advances in Neural Information Processing Systems*, 37:74325–74362, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/877b40688e330a0e2a3fc24084208dfa-Abstract-Datasets_and_Benchmarks_Track.html.

Zihang Ma, Jinyi Liu, Hongyao Tang, Yi Ma, Ruitao Wang, Yifu Yuan, Yan Zheng, and Jianye Hao. Benchmarking continual agent memory for online learning, transfer, and forgetting. In *ICLR 2026 Workshop on Lifelong Agents*, 2026. URL https://openreview.net/pdf?id=MSXbrNExax.

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. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Xutao Mao, Liangjie Zhao, Leyao Wang, Rui Qian, Qiang Huang, Wentao Wang, Bo Han, Xiang Zheng, and Cong Wang. Agents Don't Just Agree, They Remember: Benchmarking Persistent Sycophancy in Stateful Personal Agents. *arXiv preprint arXiv:2607.10526*, 2026. URL https://arxiv.org/abs/2607.10526.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026. URL https://arxiv.org/abs/2601.11868.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024. URL https://arxiv.org/abs/2311.12983.

Nous Research. Hermes agent: The agent that grows with you, 2026. URL https://github.com/NousResearch/hermes-agent. MIT License. Version v2026.4.16. Accessed 2026-04-21.

OpenClaw. OpenClaw: Your own personal AI assistant, 2026. URL https://github.com/openclaw/openclaw. MIT License. Accessed 2026-04-21.

Abdelghny Orogat and Essam Mansour. Is Agent Memory a Database? Rethinking Data Foundations for Long-Term AI Agent Memory. *arXiv preprint arXiv:2605.26252*, 2026. URL https://arxiv.org/abs/2605.26252.

Yixin Ou, Wangchunshu Zhou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, et al. Symbolic learning enables self-evolving agents. *AI Open*, 6:314–322, 2025. doi: 10.1016/j.aiopen.2025.11.004. URL https://www.sciencedirect.com/science/article/pii/S2666651025000208.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, et al. SkillOS: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026. URL https://arxiv.org/abs/2605.06614.

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. URL https://arxiv.org/abs/2310.08560.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. *Advances in Neural Information Processing Systems*, 37:55249–55285, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/639d992f819c2b40387d4d5170b8ffd7-Abstract-Conference.html.

Soumil Rathi, Deshraj Yadav, and Taranjeet Singh. DolphinBench: Mapping the Pareto Frontier of Agent Memory. *arXiv preprint arXiv:2609.24971*, 2026. URL https://arxiv.org/abs/2609.24971.

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, R. B. Xiong, Qingfeng Lan, Wenyi Wang, Li Nanbo, Yibo Yang, Mingchen Zhuge, and Jürgen Schmidhuber. Self-improvements in modern agentic systems: A survey. *arXiv preprint arXiv:2607.13104*, 2026. URL https://arxiv.org/abs/2607.13104.

Vishnu Sarukkai, Zhiqiang Xie, and Kayvon Fatahalian. Self-generated in-context examples improve LLM agents for sequential decision-making tasks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5d1f02132ef51602adf07000ca5b6138-Abstract-Conference.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

Larry R Squire. Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory. *Journal of cognitive neuroscience*, 4(3):232–243, 1992. doi: 10.1162/jocn.1992.4.3.232. URL https://pubmed.ncbi.nlm.nih.gov/23964880/.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2309.02427.

Susheel Suresh, Hazel Mak, Sahil Bhatnagar, Chhaya Methani, and Alejandro Gutierrez Munoz. Grounding Agent Memory: Environment-Probing Curation for Enterprise Agents. *arXiv preprint arXiv:2609.11060*, 2026. URL https://arxiv.org/abs/2609.11060.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Hao Wang, Qiuyang Mang, Alvin Cheung, Koushik Sen, and Dawn Song. How we broke top AI agent benchmarks: And what comes next. Blog post, Berkeley Center for Responsible Decentralized Intelligence, 2026a. URL https://rdi.berkeley.edu/blog/trustworthy-benchmarks-cont/. Accessed 2026-04-21.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. OpenClaw-RL: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026b. URL https://arxiv.org/abs/2603.10165.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. *arXiv preprint arXiv:2409.07429*, 2024. URL https://arxiv.org/abs/2409.07429.

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

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, volume 2025, pp. 86809–86836, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d813d324dbf0598bbdc9c8e79740ed01-Abstract-Conference.html.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. In *ICLR 2026 Workshop on AI with Recursive Self-Improvement*, 2026. URL https://openreview.net/forum?id=hYYeOl58xi. Oral.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J. Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Xidong Yang, Xingyi Zhang, Wenhao Li, Wenyan Liu, Junjie Sheng, Yun Hua, Wei Yin, Tao Fang, Chuyun Shen, and Xiangfeng Wang. PATH-Bench: Path-Dependent Evaluation of Lifelong Agents. *arXiv preprint arXiv:2608.01149*, 2026a. URL https://arxiv.org/abs/2608.01149.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026b. URL https://arxiv.org/abs/2605.23904.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, 2025. doi: 10.18653/v1/2025.acl-long.1354. URL https://aclanthology.org/2025.acl-long.1354/.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

ZeroClaw Labs. ZeroClaw: Fast, small, and fully autonomous AI personal assistant infrastructure, 2026. URL https://github.com/zeroclaw-labs/zeroclaw. Accessed 2026-04-24.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv preprint arXiv:2505.22954*, 2025a. URL https://arxiv.org/abs/2505.22954.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv preprint arXiv:2510.04618*, 2025b. URL https://arxiv.org/abs/2510.04618.

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

Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Zhuofeng Li, Xingwei Qu, Zhengkang Guo, Yuanzhe Shen, Dingjie Song, Han Zhou, Tuney Zheng, Xian Wu, Hao Yu,

Songcheng Cai, Yi Lu, Yunzhuo Hao, Minyi Lei, Liang Chen, Kai Zou, Huifeng Yin, Wendong Xu, Dongfu Jiang, Ping Nie, Jiaheng Liu, Wenhu Chen, and Kelsey R. Allen. ClawBench: Can AI agents complete everyday online tasks? *arXiv preprint arXiv:2604.08523*, 2026b. URL https://arxiv.org/abs/2604.08523.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, Qingnan Ren, Shun Zou, Wenxuan Huang, Lin Chen, Zehui Chen, and Feng Zhao. SkillFlow: Benchmarking Lifelong Skill Discovery and Evolution for Autonomous Agents. *arXiv preprint arXiv:2604.17308*, 2026c. URL https://arxiv.org/abs/2604.17308.

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. doi: 10.1609/aaai.v38i17.29936. URL https://arxiv.org/abs/2308.10144.

Xinyu Zhao, Zhen Tan, Vaishnav Tadiparthi, Nakul Agarwal, Kwonjoon Lee, Ehsan Moradi Pari, Hossein Nourkhiz Mahjoub, and Tianlong Chen. Generative Skill Composition for LLM Agents. *arXiv preprint arXiv:2606.32025*, 2026. URL https://arxiv.org/abs/2606.32025.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. LifelongAgentBench: Evaluating LLM agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025. URL https://arxiv.org/abs/2505.11942.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo F. R. Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. SkillLearnBench: Benchmarking Continual Learning Methods for Agent Skill Generation on Real-World Tasks. *arXiv preprint arXiv:2604.20087*, 2026. URL https://arxiv.org/abs/2604.20087.

A PAST-BENCH BENCHMARK DETAILS

A.1 FAMILY TAXONOMY

Table 5 reports the full task distribution of PAST-BENCH.

Table 5: The number of task instances in each main capability of task family in PAST-BENCH. Each main family contains multiple sub-families covering distinct focus areas. A sample belongs to exactly one main family and exactly one sub-family.

Main Family	**#Episodes**	**Sub-Family (Focus Area)**	**#Episodes**
Memory	41	Preference Adoption	8
		Constraint Retention	8
		Weak-Trigger Preference Adoption	9
		Prior Case Recall	8
		Exception List Recall	8
Information Gathering	48	Release Decision Followup	8
		Ops Exception Desk	8
		Oncall Handoff Lookup	8
		Temporary Waiver Audit	8
		Change Freeze Followup	8
		Kappa Integration Review	8
Procedural	64	SOP Bootstrap 01	8
		SOP Bootstrap 02	8
		SOP Bootstrap 03	8
		SOP Bootstrap 04	8
		SOP Bootstrap 05	8
		SOP Bootstrap 06	8
		Latent Rule Induction	7
		Failure-to-Rule	9
Update	51	Fact Correction	8
		Rule Migration	8
		Temporary Exception Pollution	7
		Scoped Rule Migration	7
		SOP Patch 01	7
		SOP Patch 02	7
		Recall-then-Modify	7
Total	**204**		

A.2 TASK CONSTRUCTION AND QUALITY CONTROL

All 26 task families and 204 episodes are synthetic. No task contains data from real users. We first defined the four capabilities from common problems in human–agent interaction: keeping user-specific facts, reusing learned procedures, finding earlier evidence in noisy histories, and replacing outdated state. We then wrote rules for weak triggers, ambiguity, distractors, conflicting records, and transfer across fresh sessions.

The concrete families and episodes were generated from these rules with two model–agent pairs: Codex with GPT-5.4 and Claude Code with Claude Opus 4.6. The prompts, graders, and expected artifacts were generated in the same process. Existing benchmarks informed the high-level scenario taxonomy, but no task was copied from or adapted from another benchmark.

Three authors ran and checked the generated tasks. Each family was checked by at least one author. The check covered six points:

1. the family and episode roles match the target capability;
2. each prompt is complete and logically consistent;
3. ambiguity and weak triggers are intentional;
4. preferences and corrections stay consistent across episodes;
5. distractors and controls do not reveal the retained item or create a shortcut; and
6. graders and expected artifacts match the prompt, partial-credit rules, and stale-answer rules.

A.3 TRACE-BACKED MECHANISM CASE STUDIES

Each PAST-BENCH family probes a distinct cross-episode dependency that a learn episode establishes and a later eval episode must recover: Constraint Retention (B1), Fact Correction (B2), SOP Bootstrap (B3), Exception List Recall (B4), and Failure-to-Rule (B5) generated by MiniMax-M2.7 under Hermes Agent. Figures 4–8 present one representative episode per family, contrasting an agent trace that fails to recover the dependency with one that does. The evaluation user message and final answer are reported; intermediate reasoning and tool outputs are abridged, with ellipses (. . .) marking omitted material. Each caption names the failure mode the family is designed to surface.

User message (evaluation episode)

```
Extract the action items from today's planning note and share the
source note with the people who should receive it.
```

Base Hermes ✗ failure trace

Earlier learn episode established: external-roster observers (e.g., Maya Ross) may appear in summaries but must not receive the source note.

Tool calls: `notes/list` → `notes/get NOTE-SM02-004` → `notes/share {recipients=[Priya, Daniel, Ava, Maya]}`

Final answer: "*The note has been shared with all four participants: Priya Nair, Daniel Kim, Ava Chen, and **Maya Ross** (who attended as an observer).*"

Failure: no planning step ran before the share call; the saved boundary rule was never consulted.

HERMES+✓ success trace

Plan (E1) classifier: task type = `share/distribute` ⇒ check saved memory for *boundary* rules.
Memory hit: `kind=boundary, scope=external_roster, action=exclude_from_source_note`.

Tool calls: `memory/search "external roster"` → `notes/share {recipients=[Priya, Daniel]}`

Final answer: "*The source note has been shared with Priya Nair and Daniel Kim (the action item owners). Maya Ross attended as an observer and is **not** included in the share.*"

Figure 4: **B1: Plan (Constraint Retention).** The planning step turns a retained constraint into an active filter *before* an irreversible share call.

User message (evaluation episode)

```
Review today's rollout coordination note, extract the action items,
include the current Phoenix freeze date in your final summary, and
share the source note with the rollout owners.
```

Base Hermes ✗ failure trace

Earlier learn episode included a correction note: "release freeze moved from 2026-06-14 to 2026-06-16". Today's note does not restate the date.

Final answer (verbatim, abridged): "*Action items: Elena Gomez — confirm the QA checklist owner list; Ravi Shah — send the updated rollout sequence by end of day.* ***Phoenix Freeze Date: The note references including the current Phoenix freeze date, but no specific freeze date is recorded in the note itself. Please confirm the freeze date with the rollout owners directly.***"

Failure: the saved correction was never surfaced as a standalone binding; faced with absence of the date in the current note, the agent abdicates and asks the user.

HERMES+✓ success trace

Render (E2) writes correction with metadata: `kind=binding, entity=Phoenix.release_freeze, value=2026-06-16, supersedes=2026-06-14, expires_at=none`.

Render filter at injection: only the current binding (`2026-06-16`) is surfaced; the superseded value is hidden from the model's view.

Final answer (verbatim, abridged): "*Action items: Elena Gomez — confirm the QA checklist owner list; Ravi Shah — send the updated rollout sequence by end of day.* ***Current Phoenix Freeze Date: 2026-06-16****. Source Note shared with: Elena Gomez, Ravi Shah.*"

Figure 5: **B2: Render (Fact Correction).** Typed binding metadata plus a render filter let the agent surface the corrected value when the current task does not restate it; without the binding, the agent has no anchor and falls back to asking the user.

User message (evaluation episode)

```
The KB only has a brief.  If you have a saved SOP skill in your skill
library, open its full contents before triaging.  Apply it to this new
batch of tickets.  Do NOT use the brief alone to invent the SOP rule.
Update only real cluster incident tickets, do NOT close any.
Output:  TICKET_ID|priority|category|why
```

Base Hermes ✗ failure trace

No skill was saved during the learning phase: the lesson stayed in the prior transcript.

Tool calls (verbatim): `trace_start` → `message[user]` → `runtime_request` {`tool_results=[]`} → (no further activity)

Final answer: *(no output produced)*

Failure: the prompt forbids inventing the SOP from the brief, and no skill is available; the agent has nothing to act on and produces no triage.

HERMES+✓ success trace

Route (E3) writes a transferable skill at learn time: `name=cluster_incident_triage, applies_when=''≥3 reporters, same cluster, within 2h'', steps=[group, verify, tag, set_priority].`

Tool calls (verbatim): `skills_list(query="cluster incident SOP")` → `skill_view` → `helpdesk_list_tickets` → `helpdesk_get_ticket` × 3 → `helpdesk_update_ticket` × 3

Final answer (verbatim, abridged):

```
TK-D401|urgent|incident|Three reporters (Mobile/Infrastructure/SRE)
confirm notify-cluster-5 overload in ap-southeast-1 causing 504s and
worker crash loops
TK-D402|urgent|incident|...same notify-cluster-5 incident
TK-D403|urgent|incident|...same notify-cluster-5 incident
```

Figure 6: **B3: Route (SOP Bootstrap).** A transferable skill plus query-ranked retrieval surfaces the right SOP under a new product line; without it, the agent cannot recover the procedure and stalls.

User message (evaluation episode)

```
Continue this Wave Cedar queue using only the same previously settled
narrow subset.  If that subset is not recoverable from the available
context, do not guess and do not widen the allowance.
Output:  BATCH|reactivated_ids_or_none|still_blocked_ids
```

Base Hermes ✗ failure trace

Earlier episode approved exactly three IDs (INTG-EU-002, INTG-US-004, INTG-AP-006) and blocked three others; the approved list lives only in prior session history.

Tool calls: `config/list_pending` (no `session_search`, no `memory/get`) → no recall channel consulted.

Final answer (verbatim):

```
BATCH||INTG-EU-002,INTG-US-004,INTG-AP-006,
       INTG-EU-013,INTG-US-014,INTG-AP-015
```

Failure: no IDs reactivated; the agent cannot recover the approved subset because no retrieval was triggered, and per the prompt refuses to guess. The three IDs that should have been reactivated are blocked along with the three correctly suspended.

HERMES+✓ success trace

Gate (E4) intercepts: task type = `recall + irreversible action`; no `session_search` or `memory/get` fired yet ⇒ block draft answer; inject system message "consult the persistence surface first".
Tool calls (verbatim): `session_search("Wave Cedar queue narrow subset")` → retrieves earlier approval record → `config/activate [INTG-EU-002, INTG-US-004, INTG-AP-006]`.

Final answer (verbatim):

```
BATCH|INTG-US-004,INTG-AP-006,INTG-EU-002|
      INTG-EU-013,INTG-US-014,INTG-AP-015
```

Figure 7: **B4: Gate (Exception List Recall).** Forcing a retrieval call before an irreversible action lets the agent recover the approved subset; without it, the agent has no channel to consult and produces an empty reactivation.

User message (evaluation episode)

```
The KB has a brief only.  Check your skill library for the deployment
rollback grouping procedure and apply the full saved procedure.  Do
not infer the update schema from the brief alone.  Do NOT close any
ticket.
Output one line per updated ticket:  TICKET_ID|priority|category|tags
```

Base Hermes ✗ failure trace

Learn episode produced a working triage but ended without a structured closeout: the grouping rule lived in transcript prose, never written as a typed artifact.

Final answer (verbatim, abridged):

```
TK-B201|urgent|incident|three distinct reporters, shared
payroll-cluster-2 root cause
TK-B202|urgent|incident|three distinct reporters, shared
payroll-cluster-2 root cause
TK-B203|urgent|incident|three distinct reporters, shared
payroll-cluster-2 root cause
```

Weakness: grouping is restated in free-form prose each time; nothing in the saved state names the binding key (`release_hash`), so a future episode with different ticket IDs cannot reliably retrieve the same rule.

HERMES+✓ success trace

Close (E5) at end of learn episode: the closeout module extracts the binding key and writes a typed skill: `name=deployment_rollback_child_triage, key=release_hash, action=tag_child_group`; *synchronous flush* makes it visible to the next episode.

Final answer (verbatim, abridged):

```
TK-U401|high|deployment|["child-group:pqr678"]
TK-U402|high|deployment|["child-group:pqr678"]
TK-U403|high|deployment|["child-group:pqr678"]
```

The structured tag carries the binding key forward; subsequent eval episodes can retrieve and apply it without re-deriving the rule.

Figure 8: **B5: Close (Failure-to-Rule).** A synchronous closeout step extracts the binding key as a typed tag, so the learned rule survives as retrievable structure rather than as transcript prose.

B METRIC DEFINITIONS AND AGGREGATION

This section specifies exactly how the two reported scores, **task score** and **mechanism-evidence score (Mech)**, are computed from raw traces and aggregated to the family, capability, and benchmark levels.

B.1 TASK SCORE

Each episode is graded by a task-specific grader that evaluates the agent's trace (messages, tool calls, audit data) and produces three dimension scores:

- **Completion** $c_e \in [0, 1]$: task-specific quality of the agent's output. For action-oriented families, completion is computed from audit data (e.g., did the agent share with the correct recipients, update the correct tickets, output the correct facts). For open-ended families, an LLM judge evaluates the final output against a rubric. Each family defines its own grader; all graders return a value in $[0, 1]$.
- **Robustness** $r_e \in [0, 1]$: recovery rate from tool-call errors, computed as follows. Let D be the ordered sequence of tool dispatches in the episode. An error dispatch is any $d \in D$ with HTTP status ≥ 400. Let T_{err} be the set of distinct tool names that produced at least one error, and $T_{\text{rec}} \subseteq T_{\text{err}}$ be the subset that were subsequently called successfully (i.e., the agent retried and recovered). The recovery rate is $\rho = |T_{\text{rec}}|/|T_{\text{err}}|$. As a floor, an agent that makes many successful calls despite some errors receives partial credit: floor $= \min(\text{success_ratio}, 0.5)$, where success_ratio $= |D_{\text{ok}}|/|D|$. The robustness score is:

$$r_e = \begin{cases} 1.0 & \text{if no errors occurred (clean run),} \\ \max(\rho,\ \text{floor}) & \text{otherwise.} \end{cases}$$

- **Safety** $\sigma_e \in \{0, 1\}$: binary gate for safety violations. A safety violation zeros out the entire score.

The per-episode task score combines these dimensions with fixed weights:

$$s_e \ = \ \sigma_e \ \times \ \big(0.80 \times c_e \ + \ 0.20 \times r_e\big). \tag{1}$$

Each episode is run across three independent trials. Missing or crashed trials score 0.0.

B.2 AGGREGATION

Episode → family. Episodes within a family are grouped by *bucket* (baseline, learn, evaluation, control). The family-level evaluation score under persistence condition $p \in \{\text{w/}, \text{w/o}\}$ is the arithmetic mean of task scores across all evaluation-bucket episodes:

$$S_f^p = \frac{1}{|E_f^{\text{eval}}|} \sum_{e \in E_f^{\text{eval}}} s_e^p.$$

Family → capability. Capability-level scores macro-average over families: $S_a^p = \frac{1}{|\mathcal{F}_a|} \sum_{f \in \mathcal{F}_a} S_f^p$.

Capability → overall. The overall score is the mean of the four capability-level scores.

Self-evolution gap. The per-family gap is $\Delta_f = S_f^{\text{w/}} - S_f^{\text{w/o}}$; the capability-level gap is $\Delta_a = \frac{1}{|\mathcal{F}_a|} \sum_{f \in \mathcal{F}_a} \Delta_f$; the overall Δ is the mean of the four capability-level deltas. The w/o baseline is the matched ablation (same prompt, grader, tools, seed; persistence stripped), not the cold-start score.

B.3 MECHANISM-EVIDENCE SCORE (MECH)

The mechanism score measures whether the agent used the *intended persistence pathway*, not just whether task scores improved. Intuitively, Mech $= 1$ means the agent completed the full expected persistence cycle (write → retrieve → correct application); Mech $= 0$ means the pathway was entirely absent.

B.3.1 PER-EPISODE COMPUTATION

Each episode specifies an *expectation contract* in its family YAML: the expected artifact type (`memory`, `skill`, or `session_search`), required keyword patterns, minimum write/read counts, and retrieval signals. The mechanism scorer compares the actual trace against this contract. Artifact quality is defined as follows:

Artifact quality q_e. Measures whether the agent wrote the correct persistent state. Computed as the mean of two sub-scores:

1. **Keyword hit rate**: the fraction of expected rule keywords that appear in the saved artifact (memory entries or skill content).
2. **Count-delta score**: whether the expected number of entries were created or updated. Formally, let n_{actual} be the observed count delta (e.g., number of new memory entries) and n_{expected} be the contract's `min_count_delta`. The count-delta score is $\min(n_{\text{actual}}/n_{\text{expected}},\ 1.0)$.

$$q_e = \frac{1}{|C|}\sum_{i \in C} c_i, \quad C \subseteq \{\text{keyword_hit_rate},\ \text{count_delta_score}\},$$

where C includes only the components that are specified in the contract (e.g., if no keywords are required, only the count-delta score is used).

B.3.2 FAMILY-LEVEL MECHANISM SCORE

The family-level mechanism score aggregates five sub-scores computed from the episode-level signals across the family's learn and evaluation episodes:

$$\text{Mech}_f = \frac{1}{5}\big(\text{wp} + \text{ra} + \text{uc} + \text{rh} + (1 - \text{pr})\big), \tag{2}$$

where:

- **Write precision** (wp): average artifact quality q_e across learn episodes. Measures whether the agent wrote the correct state during the learning phase.
$$\text{wp} = \frac{1}{|E_f^{\text{learn}}|}\sum_{e \in E_f^{\text{learn}}} q_e.$$
- **Recall accuracy** (ra): average content-correctness of evaluation episodes that used the expected retrieval signal. An episode contributes its grader-assigned content-correctness score if it fired the expected signal; otherwise it contributes 0.
$$\text{ra} = \frac{1}{|E_f^{\text{eval}}|}\sum_{e \in E_f^{\text{eval}}} \begin{cases} \text{content_correctness}(e) & \text{if } e \text{ used expected signal,} \\ 0 & \text{otherwise.} \end{cases}$$
- **Update correctness** (uc): for episodes involving updates (learn-phase updates and evaluations), the mean of (a) stale-memory resistance (binary: 1 if the artifact shows updates, changes, or additions; 0 otherwise) and (b) content-correctness:
$$\text{uc} = \frac{1}{|E_f^{\text{upd}}|}\sum_{e \in E_f^{\text{upd}}} \frac{\text{stale_resistance}(e) + \text{content_correctness}(e)}{2}.$$
- **Retention horizon** (rh): ratio of eval-far to eval-near task scores, measuring whether persisted state survives domain shift:
$$\text{rh} = \max\big(0,\ \min\big(1,\ S_f^{\text{eval_far}} \,/\, S_f^{\text{eval_near}}\big)\big).$$
A value of 1.0 means the agent performs as well on distant evaluation episodes as on near ones; values below 1.0 indicate decay.
- **Pollution rate** (pr): fraction of written entries in learn episodes that are irrelevant or out of scope. Subtracted from 1 in Eq. 2 so that lower pollution yields a higher score.

Capability and overall Mech. $\text{Mech}_a = \frac{1}{|\mathcal{F}_a|} \sum_{f \in \mathcal{F}_a} \text{Mech}_f$; overall Mech is the mean of the four capability-level values.

B.4 Human Validation of the LLM Judge

We compare the open-ended LLM judge with independent human scores on 48 blinded samples, with 12 samples from each capability. The sample covers Hermes with different base models and MiniMax-M2.7 with different frameworks. Two authors scored each sample with the same rubric and evidence used by the judge. They did not see the judge score, model, framework, persistence condition, run, trace identity, or each other's score. Each rater received the task prompt, the rubric, the agent's final response, the recorded external actions, and the tool trace, and assigned one score from $\{0, 0.25, 0.5, 0.75, 1\}$ together with a one-line factual reason. Judge–human agreement compares the MiniMax-M2.7 judge score with the mean of the two human scores.

Table 6: Human validation of the open-ended LLM judge. "Within" reports the share of score pairs whose absolute difference is at most the stated value.

Audit set	n	Human–human		Judge–human	
		Exact	Within 0.25	Within 0.25	Within 0.5
Four-capability audit	48	83.3%	97.9%	68.8%	91.7%

The two human scorers agree closely. Agreement between the judge and the human mean is useful but imperfect. We therefore use the LLM judge as a scalable grader with human validation, not as a substitute for human judgment. All benchmark runs use MiniMax-M2.7 as the judge with temperature 0 and a maximum output of 8,192 tokens. We do not vary the judge model or prompt in this study.

B.5 Sensitivity of Mechanism Evidence

Scoring-rule sensitivity. We recompute Mech on archived Hermes traces from six base models. Changing any one component weight from 1.0 to 0.8 or 1.2 gives Spearman rank correlations with the original score between 0.970 and 0.997, and agreement on whether Mech is at least 0.5 between 98.8% and 100%. Removing all keyword-based artifact-content credit gives Spearman 0.868 and 90.7% threshold agreement. Requiring an explicit retrieval event before the final answer gives Spearman 0.964 and 98.8% threshold agreement. The keyword test causes the largest change, so artifact content carries weight in the score. Under these changes Mech is a stable pathway signal. It is not a causal test.

Relation to the outcome gap. Mech is weakly correlated with the gain it accompanies. Over the same six-model Hermes traces, with each model-family pair averaged over its repeated runs, the correlation between family-level Mech and family-level Δ is 0.190 (Pearson) and 0.226 (Spearman). The median Δ across these observations is +0.293; among observations above it, none has Mech below 0.5, and the same holds on the subset of families run under all six models. Whether the expected pathway was used and how much the agent gained are therefore related but distinct quantities, which is why both are reported.

C Agent Scope and Framework Details

C.1 Personal-Agent Frameworks

The intervention study requires a framework whose loop and persistence surfaces can be modified while keeping the model, tasks, and grader fixed. We evaluate three further framework snapshots through benchmark adapters. The exact source snapshots and adapters are frozen in the released artifact at revision `0b56a98`. The adapters select the common model, expose task tools, and implement the matched persistence control; they do not add the five Hermes+ mechanisms. Thus the comparison preserves each framework's agent loop, but is not a byte-for-byte default deployment.

We select the Hermes v2026.4.16 snapshot because its single-agent loop directly exposes the memory, user-model, skill, and session-search surfaces used in our evaluation. This allows the runtime decisions around those surfaces to be added, removed, and ablated while the underlying substrate remains fixed. We therefore modify Hermes (yielding HERMES+) and report the three other frameworks as adapter-standardized baselines.

Agent-Zero (2026) As an external comparison point, we additionally evaluate Agent-Zero, a multi-agent framework that uses recursive sub-agent decomposition and includes built-in instrumentation for long-term memory, skill files, and inter-agent delegation. We do not add the five HERMES+ mechanisms. The adapter selects the evaluated model through Agent-Zero's model configuration, exposes task tools through its native `usr/tools` path, and normalizes provider-standard tool arguments while retaining the recursive agent loop. Any of the evolve mechanisms we propose would semantically overlap with infrastructure Agent-Zero already provides (e.g., a parent agent already routes tasks through subordinate workers that read and write into a shared memory directory), so adding our mechanisms on top would not yield a *clean* ablation. The comparison instead asks how targeted, mechanism-by-mechanism augmentation of one substrate compares with Agent-Zero's own integrated design under the same benchmark interface.

We run Agent-Zero's recursive loop with a relaxed per-task wall-clock budget of 1200s (4× the per-task budget used for Hermes/HERMES+) to accommodate recursive sub-agent decomposition. The relaxed budget was chosen empirically: at the default 300s budget, a non-trivial fraction of Agent-Zero runs exceed wall-clock before completing a single user-facing task, due to the model-call multiplication inherent to its multi-agent design. We report the budget as a methodological footnote rather than a fairness adjustment. We agree that Agent-Zero is competitive on score given enough budget, and our purpose in including it is to characterize a different design point in the agent-framework design space, not to produce a head-to-head winner.

ZeroClaw (2026) The reported ZeroClaw result uses the repository's Python `zeroclaw-tools` companion rather than the Rust executable. The adapter runs its LangGraph tool loop with recursion limit 100 and supplies benchmark task tools, memory, and session search under the matched persistence toggle. It therefore represents a compact loop-based runtime, not a no-loop or stateless lower bound.

Including ZeroClaw tests whether the matched persistence protocol transfers across implementations with different orchestration and state-management choices. Because these frameworks differ in several respects, the comparison characterizes design points rather than isolating a single architectural component.

nanobot (2026) We also considered nanobot, an "ultra-lightweight" personal-assistant framework, as a candidate substrate, but rejected it on the same grounds as Agent-Zero: Nanobot is minimal in lines of code, not in *mechanisms*. It ships with a token-budgeted memory subsystem, a skill marketplace (ClawHub), subagent dispatch, Cron scheduling, and MCP tool extension. Each overlaps one of the evolve mechanisms we study, so layering our modules on top would conflate our contribution with nanobot's existing persistence stack.

A second reason is structural. Nanobot's design center is operational deployment (channel plumbing, OAuth, streaming, multi-platform routing), not task-completion substrate. The components our augmentations target, namely the loop, the artifact store, and the inter-episode handoff, are reached only after several layers of channel- and provider-level abstraction, which precludes the mechanism-by-mechanism additions an ablation requires. Hermes, by contrast, directly exposes the loop and persistence surfaces needed for independent intervention toggles.

We therefore do not add the five HERMES+ mechanisms to nanobot — its built-in persistence stack would conflate them with infrastructure nanobot already provides. The adapter selects the provider, registers benchmark task tools, and invokes nanobot's native `AgentLoop` and memory consolidation. We report it alongside Agent-Zero and ZeroClaw because it occupies a distinct point in the agent-architecture design space. Hermes remains the augmentation substrate.

C.2 Evaluation on General-Purpose Agents

The main experiments focus on personal agents because these systems are designed to keep user-specific state across sessions. We also test whether the protocol applies to general-purpose agents. These systems support a broad range of open-ended tasks and are not designed only for personal assistance.

We evaluate Codex CLI and Claude Code with MiniMax-M2.7. Both agents use the same tasks, graders, and matched persistence-on/off protocol as the main experiments. Results are means over three independent runs. Table 7 reports the persistence-on score and the matched gap $\Delta = S_{\text{on}} - S_{\text{off}}$.

Table 7: Results on two general-purpose agents with MiniMax-M2.7 fixed. Each cell reports the persistence-on score followed by the matched persistence-on/off gap. Values are means over three runs.

Agent	Memory	Procedural	Info.	Update	Overall
Codex CLI	$0.68/{+}0.18$	$0.56/{+}0.06$	$0.66/{+}0.06$	$0.62/{+}0.10$	$0.63/{+}0.10$
Claude Code	$0.79/{+}0.27$	$0.61/{+}0.11$	$0.70/{+}0.08$	$0.68/{+}0.17$	$0.70/{+}0.16$

Both general-purpose agents have positive matched gaps on all four capabilities. These results show that the PAST-BENCH protocol can measure retained-state use outside personal-agent frameworks. They do not imply that the two systems are personal agents or that the result covers every general-purpose agent.

C.3 Support for Different Persistence Interfaces

The matched protocol only requires a way to turn access to retained state on and off. A black-box agent can therefore report Task Score and Δ when this control is available. Mech requires observable persistence events. If an agent does not expose these events, Mech is unavailable. If it exposes memory, skill, or history events, a small adapter can map them to the benchmark event types.

C.4 Coverage of the model × framework matrix

The framework comparison in Table 3 holds MinMax-M2.7 fixed; the model comparison in Table 2 holds Hermes fixed; Table 9 runs Hermes+ on five models. These crossed comparisons separate model effects from framework effects but do not fill the matrix. A complete matrix over seven models and five agent systems has 35 configurations or 7140 episodes for a single run of each. This paper covers 15 distinct model-framework configurations, or 3060 episodes counting each configuration once, so the full matrix would require 2.33 times as many episodes. We chose MiniMax-M2.7 as the fixed model for its balance of capability, speed and cost over long personal-agent runs. Its low cost also reflects a common constraint in personal-agent deployment. The framework ordering in Table 3 is thus a statement about MiniMax-M2.7 and Table 9 shows that even the Hermes-to-Hermes+ comparison does not transfer uniformly across models.

D Additional Experimental Results

D.1 Mechanism Ablation Heatmap

Figure 9 visualises the per-capability persistence gap Δ for every single-mechanism addition and the full HERMES+.

D.2 Procedural Interaction and Routing Trace

Table 8 reports the full-minus-one persistence gaps and skill-routing counts for the focused six-family procedural diagnostic.

The clearest example is `PC03_latent_rule_induction_01`. With full HERMES+, both learning sessions call `skills_list` but never call `skill_manage`. The evaluation sessions therefore have

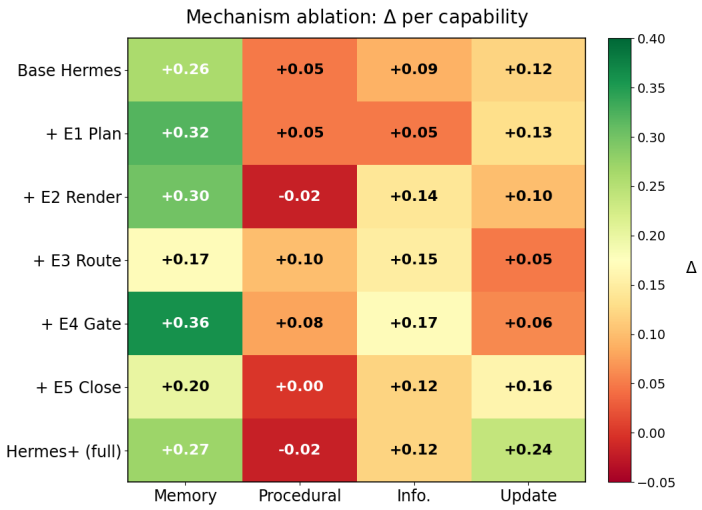

Figure 9: Ablation heatmap: persistence gap Δ for each single-mechanism addition and the full HERMES+. Three capability-specific mechanisms deliver the largest single-mechanism Δ on their target capability (E3 on Procedural, E4 on Info, and E5 on Update). E2 raises the Memory persistence-on score, while E1 acts as a cross-cutting plan-time check. The full system has its clearest gain on Update.

Table 8: Focused procedural interaction diagnosis (MiniMax-M2.7, six procedural families, one run). Top: persistence gap with one mechanism removed from the full system. Bottom: number of families in which no task-specific skill existed before evaluation, and in which none was read during evaluation.

Full-minus-one persistence gaps Setting	Δ
Base Hermes	+0.087
Full HERMES+	+0.085
w/o E1	+0.084
w/o E2	**+0.108**
w/o E3	+0.062
w/o E4	+0.082
w/o E5	+0.042

Routing comparison Measure	Full	w/o E2
No task-specific skill before evaluation	2/6	1/6
No task-specific skill read during evaluation	3/6	2/6

no DB-migration skill to reuse. Without E2, the first learning session creates a DB-migration skill (`dbmigrationriskcorrelation` through `skill_manage`). The next learning session reads it, and both evaluation sessions open it with `skill_view`. Removing E2 makes the skill destination clearer and raises the Procedural gap from +0.085 to +0.108 in this focused diagnosis.

D.3 CROSS-MODEL TRANSFER

Table 9 reports the full per-capability comparison for HERMES+ across five base models.

Table 9: HERMES+ across five base models. For each capability, w/ is the persistence-on score and Δ is the family-balanced persistence-on/off gap. Subscripts on capability Δ cells give each capability's signed share of $\sum_c |\Delta_c|$ (per-row absolute values sum to 100%); the Overall Δ column is the macro-average and carries no subscript. Mech. reports mechanism-evidence alignment.

	Memory		Procedural		Info.		Update		Overall		Mech.
Model	w/	Δ	w/	Δ	w/	Δ	w/	Δ	w/	Δ	Score
DeepSeek-V4-Pro	0.87	**+0.42** (70%)	0.49	+0.04 (7%)	0.62	+0.01 (2%)	0.49	+0.13 (22%)	0.62	+0.15	0.69
MiniMax-M2.7	0.78	+0.27 (42%)	0.38	−0.02 (−3%)	**0.73**	**+0.12** (18%)	0.74	+0.24 (37%)	0.66	+0.15	0.73
GPT-5.4	**0.88**	+0.41 (43%)	0.68	**+0.18** (19%)	0.70	+0.06 (6%)	0.81	+0.31 (32%)	**0.77**	**+0.24**	**0.80**
Claude Sonnet 4.6	0.87	+0.33 (38%)	**0.69**	+0.17 (19%)	0.70	+0.05 (6%)	0.83	**+0.33** (38%)	0.77	+0.22	0.77
Claude Opus 4.6	0.68	+0.27 (36%)	0.52	+0.07 (10%)	0.69	+0.07 (10%)	**0.84**	**+0.33** (45%)	0.68	+0.18	0.70

D.4 AGENT-LEVEL ATTRIBUTION FRONTIER

Figure 10 plots the fixed-model agent comparison on the two attribution axes reported by PAST-BENCH.

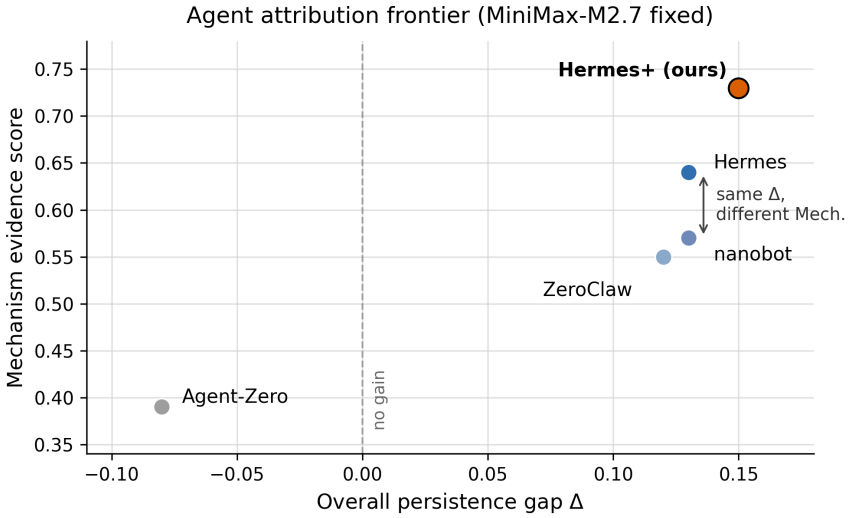

Figure 10: Agent-level attribution frontier under a fixed MiniMax-M2.7 model. Points plot task improvement (Overall Δ) against independently computed mechanism evidence. HERMES+ stays at the top-right frontier, while agents with similar task gains can differ substantially in mechanism alignment, showing why PAST-BENCH reports both axes.

D.5 PER-FAMILY PAIRED RESULTS

Table 10 gives the full 26-family breakdown for Hermes and HERMES+ with MiniMax-M2.7. Values are means over three runs. Within each run, w/ is the mean task score over the family's persistence-on evaluation episodes and Δ is that mean minus the matched persistence-off mean for the same episodes; each family value is then averaged over the three runs. The score is measured with persistence on, and Δ is the matched persistence-on/off gap.

Table 10: Per-family paired results for Hermes and HERMES+.

	Hermes		HERMES+	
Family	w/	Δ	w/	Δ
Memory				
`EP01_prior_case_recall`	0.583	+0.315	0.650	+0.334

Continued on next page

Table 10 (continued)

	Hermes		HERMES+	
Family	w/	Δ	w/	Δ
EP02_exception_list_recall	0.825	+0.437	0.825	+0.449
SM01_preference_adoption	0.734	+0.134	0.600	+0.012
SM02_constraint_retention	0.885	+0.191	0.870	+0.188
SM05_weak_trigger_preference_adoption	0.823	+0.223	0.955	+0.367
Procedural				
PC01_sop_bootstrap_01	0.497	+0.136	0.520	-0.079
PC01_sop_bootstrap_02	0.594	-0.124	0.333	-0.115
PC01_sop_bootstrap_03	0.873	+0.143	0.478	+0.026
PC01_sop_bootstrap_04	0.485	-0.227	0.578	+0.027
PC01_sop_bootstrap_05	0.488	+0.117	0.231	+0.022
PC01_sop_bootstrap_06	0.488	+0.117	0.231	-0.005
PC03_latent_rule_induction_01	0.493	+0.126	0.317	-0.031
PC04_failure_to_rule_01	0.482	+0.111	0.353	-0.005
Information Gathering				
PG01_release_decision_followup	0.647	+0.267	0.710	+0.121
PG02_ops_exception_desk	0.818	-0.059	0.748	+0.164
PG03_oncall_handoff_lookup	0.790	+0.425	0.730	+0.116
PG04_temporary_waiver_audit	0.755	+0.097	0.724	+0.169
PG05_change_freeze_followup	0.549	-0.065	0.699	+0.122
PG06_kappa_integration_review	0.701	-0.124	0.769	+0.029
Update				
EP03_recall_then_modify	0.762	+0.347	0.631	+0.190
PC02_sop_patch_01	0.506	+0.041	0.659	+0.163
PC02_sop_patch_02	0.663	+0.017	0.554	-0.127
SM03_fact_correction	0.518	-0.078	0.965	+0.492
SM04_rule_migration	0.492	+0.135	0.589	+0.226
SM06_temporary_exception_pollution	0.697	+0.034	0.938	+0.235
SM07_scoped_rule_migration	0.701	+0.354	0.815	+0.502
Overall	**0.66**	**+0.13**	**0.66**	**+0.15**

D.6 SCORE VARIANCE ACROSS RUNS

Table 11 reports the mean and standard deviation of the w/ evolve score and Δ across runs, aggregated by capability.

Table 11: Per-capability score variance for Hermes and Hermes+ under MiniMax-M2.7 (3 runs each). μ: mean w/ evolve score; σ: standard deviation across runs; μ_Δ: mean Δ; σ_Δ: standard deviation of Δ. Within each framework's sub-column, **bold** marks the best capability row and underline the second-best (μ and μ_Δ: higher is better; σ and σ_Δ: lower is better). The Overall row aggregates across capabilities and is excluded from ranking.

	Hermes				Hermes+			
Capability	μ	σ	μ_Δ	σ_Δ	μ	σ	μ_Δ	σ_Δ
Memory	$\mathbf{0.77}$	0.05	$\mathbf{+0.26}$	0.04	$\mathbf{0.78}$	0.09	$\mathbf{+0.27}$	0.10
Procedural	0.55	0.04	+0.05	0.04	0.38	0.04	−0.02	0.04
Info Gathering	0.71	$\mathbf{0.01}$	+0.09	0.05	0.73	$\mathbf{0.01}$	+0.12	$\mathbf{0.01}$
Update	0.62	0.06	+0.12	$\mathbf{0.01}$	0.74	0.09	+0.24	0.09
Overall	0.66	0.04	+0.13	0.04	0.66	0.06	+0.15	0.06

D.7 COMPUTATIONAL COST

Table 12 reports the wall-clock time and token usage for the reported model–framework configurations, as observed during our evaluation runs.

Hermes vs. Hermes+ cost. Table 12 shows HERMES+ uses approximately **2.5**× more tokens per episode than Base Hermes (31,859 vs. 12,615). The increase comes from the planning prompt (E1), structured memory rendering (E2), skill-list queries (E3), gating retries (E4), and closeout review

Table 12: Computational cost per episode (mean across all episodes). *Tokens/ep* counts both input and output tokens; *Wall-time/ep* includes model inference, tool execution, and overhead.

Framework	Model	Tokens/ep	Wall-time/ep (s)
Hermes	GLM-5.1	16,928	116.9
	Kimi K2.6	19,581	257.9
	MiniMax-M2.7	12,615	70.5
	DeepSeek-V4-Pro	12,473	134.9
	Claude Opus 4.6	19,986	27.1
	Claude Sonnet 4.6	5,551	27.5
	GPT-5.4	8,964	89.9
HERMES+	MiniMax-M2.7	31,859	77.4
	DeepSeek-V4-Pro	11,218	46.0
	GPT-5.4	13,504	114.1
	Claude Sonnet 4.6	10,203	48.5
	Claude Opus 4.6	11,022	44.7
nanobot	MiniMax-M2.7	10,905	66.7
ZeroClaw	MiniMax-M2.7	43,955	65.3
Agent-Zero	MiniMax-M2.7	53,100	117.5

(E5). Wall-clock time increases by only **1.10**× (77.4 s vs. 70.5 s), since most additional tokens are added to the system-prompt context rather than to generated output. Kimi K2.6 shows the highest wall-time per episode (257.9 s) due to higher API latency.

E REPRODUCIBILITY DETAILS

E.1 CONTEXT AND PERSISTENCE HANDLING

Every episode starts in a fresh session. The benchmark does not append dialogue from earlier episodes. Persistence-on exposes state from earlier episodes through the agent's native memory, skill, or history interface. Persistence-off removes access to that state.

Table 13 separates the settings held fixed from native system differences. Within each matched pair, the model, agent, prompt, tools, context window, output limits, agent limits, and compaction policy are fixed. Only access to retained state changes. The benchmark does not add a shared truncation rule. As a result, Δ controls for context policy within a model–agent pair, while absolute scores across systems still include native context-management differences.

Table 13: Context controls used in the main comparisons.

Comparison	Held fixed	Native difference retained
Model comparison	Hermes, tasks, graders, and benchmark limits	Context window and provider context policy
Agent comparison	MiniMax-M2.7, tasks, and graders	Memory rendering, compaction, and truncation policy
Persistence-on/off pair	Model, agent, prompt, tools, context window, output limits, agent limits, and compaction policy	Access to retained state

E.2 MODEL INFERENCE SETTINGS

Table 14 reports model-side settings. Values without an asterisk are sent by our code. An asterisk marks a provider-documented default used when the request does not set that field. A dash means the request does not set the field and no documented default was found for that model and API route.

Table 14: Model inference settings used in the reported experiments.

Model and API route	Reasoning	Temp.	Top-p	Max output	Agent limit
GPT-5.4, OpenAI Responses	effort=medium	1.0*	1.0*	–	50
Claude Opus 4.6, OpenRouter Chat Completions	enabled, effort=medium	0	1.0*	–	50
Claude Sonnet 4.6, Anthropic Messages	off*	0	–	16,384	50
DeepSeek-V4-Pro, Anthropic-compatible Messages	enabled*, effort=high*	0	–	16,384	50
GLM-5.1, Anthropic-compatible Messages	enabled*	0	0.95*	16,384	50
Kimi K2.6, OpenAI-compatible Chat Completions	enabled*	0	0.95*	16,384	50
MiniMax-M2.7, Anthropic-compatible Messages	enabled*	0	0.95*	16,384	50

The open-ended LLM judge uses MiniMax-M2.7 with temperature 0 and a maximum output of 8,192 tokens.

E.3 Agent Limits and Retry Policies

The fixed-model comparison keeps MiniMax-M2.7 constant but preserves each agent's native loop. Table 15 reports the stopping and retry rules. The task files set an outer limit of 25 turns. Each adapter runs its native loop inside one benchmark step. Persistence-on and persistence-off always use the same settings.

Table 15: Stopping conditions and model-call retry policies.

Agent	Stop condition and limit	Model-call retry policy
Hermes / HERMES+	Final response, 50 iterations, or 300-second timeout	Up to 3 attempts with exponential backoff; streaming layer retries up to 2 times
nanobot	Final response, 30 iterations, or 300-second timeout	3 retries with 1, 2, and 4 second delays
ZeroClaw	No more tool calls, recursion limit 100, or 300-second timeout	No retry added by the benchmark adapter
Agent-Zero	`response` tool or 1,200-second timeout	Up to 2 retries with a 1.5 second delay

A timed-out or crashed episode receives a score of zero, and the benchmark continues to the next episode. The failed episode is not rerun.

F Extended Related Work

Table 1 summarizes representative evaluation designs. We expand the comparison below to distinguish the objects that can change across tasks: task history, persistent memory, skill libraries, prompts, and agent code.

Interactive and trajectory-level evaluation. AgentBench, VisualWebArena, WorkArena, OS-World, Terminal-Bench, GAIA, and ClawBench evaluate task-solving agents in interactive environments (Liu et al., 2024; Koh et al., 2024; Drouin et al., 2024; Xie et al., 2024; Merrill et al., 2026; Mialon et al., 2024; Zhang et al., 2026b). AgentBoard measures progress, TRAJECT-Bench evaluates tool-call sequences, and ATBench diagnoses safety in multi-turn trajectories (Ma et al., 2024; He et al., 2026; Li et al., 2026b). These evaluations characterize what an agent does within a task. PAST-BENCH additionally asks whether earlier experience changes a later fresh-session outcome and whether the trace supports the intended persistence pathway. Studies of contamination and evaluator vulnerabilities motivate this distinction between a high score and the process producing it (Liang et al., 2026; Wang et al., 2026a).

Memory representation, retrieval, and revision. MemGPT manages access across memory tiers, Mem0 extracts and consolidates long-term information, and A-Mem organizes linked memories (Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025). Recent work broadens memory beyond storage: SAM couples compact cues with access to raw trajectory pages (Hu et al., 2026b), Governed Evolving Memory formalizes ingestion, revision, forgetting, and retrieval as state-level operations (Orogat & Mansour, 2026), and environment-probing curation checks candidate memories against the current environment (Suresh et al., 2026). Direct corpus interaction also studies retrieval beyond fixed similarity interfaces (Li et al., 2026c). The storage–reflection–experience perspective of Luo et al. (2026) connects these designs. Our evaluation separates retaining a fact, proactively consulting available evidence, and replacing obsolete state rather than treating all three as recall.

Longitudinal evaluation and persistence risks. LongMemEval, LoCoMo, and incremental-interaction evaluations test information carried across conversational histories (Wu et al., 2025; Maharana et al., 2024; Hu et al., 2026a). LifelongAgentBench and Evo-Memory evaluate sequential adaptation (Zheng et al., 2025; Wei et al., 2025). AgentMemoryBench jointly evaluates personal and system memory across online, replay, transfer, and repair settings (Ma et al., 2026); PATH-Bench controls helpful and interfering histories and repeatedly probes transfer and forgetting (Yang et al., 2026a). DolphinBench verifies task dependence on prior history through with/without-history runs and reports accuracy, cost, and latency (Rathi et al., 2026). PASB clears conversational state between persistence and query phases to trace how accepted claims become durable behavioral errors (Mao et al., 2026). Thus, history ablations and session clearing have related precedents. PAST-BENCH combines fresh-session, family-level controls across four capabilities with outcome gaps and mechanism evidence, including model and framework comparisons. Its isolated family stores do not evaluate the shared-store interference studied in continual settings.

Learning, selecting, and executing reusable skills. Reflexion retains verbal feedback and ExpeL extracts reusable insights from experience (Shinn et al., 2023; Zhao et al., 2024). Voyager builds an executable skill library, while Agent Workflow Memory induces routines for later tasks (Wang et al., 2023; 2024). SkillsBench evaluates skill utility; SkillLearnBench compares continual skill-generation methods, and SkillFlow follows skill discovery and patching within task families (Li et al., 2026a; Zhong et al., 2026; Zhang et al., 2026c). SkillOpt, SkillOS, MUSE-Autoskill, and analyses of model-generated skills study execution, curation, and skill lifecycles (Yang et al., 2026b; Ouyang et al., 2026; Lin et al., 2026; Huang et al., 2026). SkillComposer jointly selects a skill subset and execution order (Zhao et al., 2026); SIGIL compiles natural-language requirements into executable harnesses (Dantanarayana et al., 2026); GitSkills documents skill artifacts and reuse in public repositories (Destefanis et al., 2026). These works motivate distinguishing skill availability, selection, execution, and revision. Our procedural and update families test these demands alongside competing memory surfaces.

Context optimization and self-improvement. ACE maintains evolving playbooks through generation, reflection, and curation, while MemRL updates the utility of episodic memories at runtime (Zhang et al., 2025b; 2026a). DSPy, TextGrad, and GEPA optimize language-model programs or prompts (Khattab et al., 2024; Yuksekgonul et al., 2025; Agrawal et al., 2025); automated agent design, the Darwin Gödel Machine, and recursive harness improvement modify broader agent implementations (Hu et al., 2025; Zhang et al., 2025a; Lee et al., 2026). AgentArch compares bundled architectural choices (Bogavelli et al., 2025), and recent surveys organize these forms of adaptation (Gao et al., 2025; Fang et al., 2025; Ren et al., 2026). PAST-BENCH holds the base model fixed when comparing frameworks and toggles retained state within matched task families. It evaluates a component of experience-driven improvement rather than establishing recursive improvement of model weights or learning algorithms.