USE AND DISUSE: INTENT-STRUCTURED EXPERIENCE CONSOLIDATION FOR MEMORY AND LEARNING IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The evolution of Large Language Model agents from single-task execution to long-term autonomous operation highlights the critical challenge of transforming continuous experiences into reusable knowledge. To address this, we propose Hippocam, a hierarchical memory and continual learning architecture. Hippocam draws inspiration from two characteristics of human memory: cognitive processes selectively maintain information relevant to current goals, while long-term memories form gradually through repeated consolidation. Accordingly, Hippocam structures an agent's ongoing work as nested intents. The active context remains centered on the current intent, while completed intents are consolidated into the task-relevant outcomes and state needed for subsequent work, rather than carrying forward their full working details. Concurrently, a recursive prefix consolidation mechanism repeatedly consolidates earlier history, causing long-unused experiences to become increasingly abstract. Original interactions are preserved, allowing the agent to progressively recover finer-grained details through the hierarchy and stop once sufficient information is available. Crucially, when past experiences are recalled and reintegrated into active work, they undergo subsequent consolidation alongside new experiences, thereby being reinforced, supplemented, and updated. Through this memory dynamic of use and disuse, Hippocam connects working context, long-term memory, knowledge accumulation, and skill learning within a single continuously evolving experiential process. This enables agents to learn and evolve capabilities through their own experiences without parameter updates.

1 INTRODUCTION

Large Language Model (LLM) agents are increasingly capable of carrying out complex tasks through sustained interaction with their environments (Yao et al., 2023). In long-term, open-ended interactions, enabling agents to continually learn from accumulating experience through memory remains a central challenge for sustained autonomous operation (Wang et al., 2024).

From the perspective of human cognition, learning through problem solving forms a cycle in which accumulated experience guides the solution of new problems (Kolodner, 1992). Memory provides the mechanism through which work and learning become deeply intertwined. Short-term memory remains tightly focused on the current intent, selectively maintaining the state of the environment and the outcomes and experience acquired so far, thereby enabling sustained focus and progress on the current task (Miller & Cohen, 2001). Long-term memory develops through repeated practice on related tasks and consolidation, as experience accumulates and understanding deepens (Ericsson & Kintsch, 1995; Dudai et al., 2015). When related situations arise, past experience can be reactivated to support understanding, reasoning, and decision-making on new problems (Zeithamova et al., 2012; Shadlen & Shohamy, 2016).

Inspired by this interplay, we introduce Hippocam, an LLM agent that continually learns from its own work through a unified process of organizing and consolidating its interaction history. Hippocam does not build a memory bank separate from its interaction history, but remembers and learns by deliberately maintaining its context. It organizes ongoing work as nested intents, opening an intent when the work develops a more immediate purpose. As intents close, Hippocam consolidates their

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.

	Learning			Transfer		Memory		
	ALFWorld	SciWorld	StreamBench	Frozen Science	Frozen FEVER	GoodAI interleaved	LoCoMo overall	MAB multi-hop
HiAgent	85.1	56.7	69.0	42.7	61.8	34.0	39.8	36.0
mem0	87.3	58.6	69.3	51.7	59.9	3.3	58.0	20.3
A-Mem	95.5	87.2	79.0	42.1	55.8	1.1	67.5	24.3
Letta	85.1	71.2	81.3	40.8	59.9	50.4	51.7	44.3
ReasoningBank	85.1	66.9	68.3	53.3	55.8	2.2	18.3	0.0
ACE	97.8	91.9	75.0	62.7	57.9	0.0	22.5	0.0
Hippocam	97.8	81.9	81.7	78.9	61.8	88.7	77.6	71.7

Figure 1: **Hippocam unifies memory, knowledge use, and learning.** Performance (%) with DeepSeek-V4.1-Flash as the task backbone. Evaluation details appear in § 4.

working processes into the outcomes and state relevant to each intent, replacing the original messages in the active context. As the interaction history grows, Hippocam recursively consolidates more distant spans, similarly replacing them with the resulting messages. These consolidated messages can later be consolidated again alongside newly accumulated messages. This leaves older experience progressively more abstract. Both forms of consolidation keep the agent's recent context focused and high-signal, enabling accurate and coherent reasoning over long horizons. They also preserve the original interactions beneath their consolidated representations, naturally organizing the history into a tree. When subsequent work requires finer evidence, the agent descends selectively through this tree, recovering details until it has enough information to proceed. Hippocam draws on past experience to interpret new situations and develop approaches, with the resulting successes and failures providing feedback. This feedback enters subsequent consolidation together with prior knowledge, allowing Hippocam to reinforce, supplement, or revise what it has learned and develop broader, reusable insights. Experience that remains unused continues to recede in detail through recursive consolidation, without separately assigned importance scores or time-based forgetting rules. This natural dynamic of *use and disuse* unifies the recovery of past experience with learning that deepens what the agent knows, refines how it acts, and expands what it can accomplish. Hippocam thereby makes the agent's lifetime of work the engine of continual learning, taking a step toward a unified architecture for memory, knowledge, and learning.

Evaluations show that Hippocam achieves strong performance across memory, knowledge use, and learning. Whereas competing methods show more uneven performance across these demands, Hippocam leads or matches the best compared method on seven of the eight measures in Fig. 1. Ablations and case analyses further show how consolidation and progressive recall support ongoing work and how knowledge develops through use and disuse. These results demonstrate how our use-and-disuse cycle supports memory and learning through interaction history.

2 HIPPOCAM

2.1 OVERVIEW

Hippocam is an LLM agent that continually learns from its own work by organizing and consolidating its interaction history. As shown in Fig. 2, it comprises a main agent responsible for reasoning, tool use, and interaction with the environment, supported by three auxiliary LLM-based components. JANUS maintains the nested intent stack and provides guidance before each main-agent inference (§ 2.2). PRECIP performs Intent Consolidation when intents conclude, retaining results and grounded lessons at the level of detail needed for ongoing work (§ 2.3). PALIM performs Prefix Consolidation over earlier context, recursively making its content more abstract and less detailed while merging and updating retained knowledge, including acquired facts and learned lessons (§ 2.4).

Both consolidation mechanisms preserve the messages they replace, forming a message tree from which the main agent can progressively recover finer-grained messages through a recall tool (§ 2.5). Relevant knowledge already in context or recovered through recall can participate in ongoing work and be reinforced, supplemented, or updated through subsequent consolidation, while knowledge not

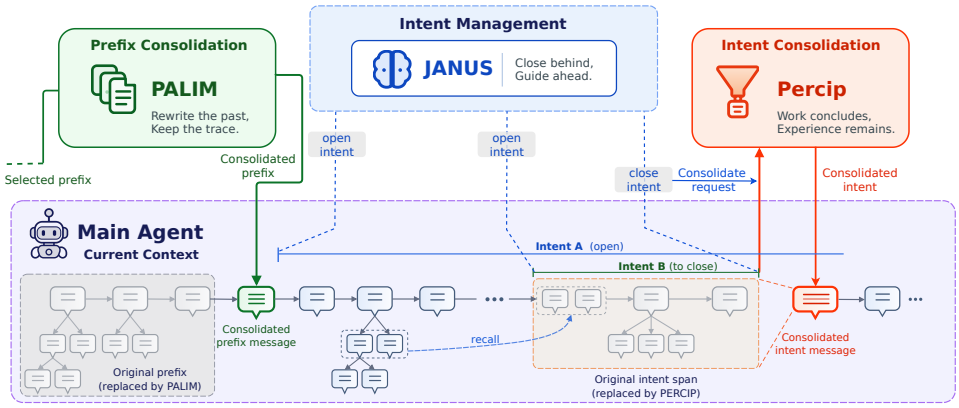


Figure 2: **Overview of Hippocam.**

currently used gradually loses detail through recursive Prefix Consolidation. Together, use and disuse connect task execution, memory, and learning in a continuous cycle (§ 2.6).

2.2 Intent-Structured Working Context

Intent stack. Let $\mathcal{C} = \langle m_1, \ldots, m_n \rangle$ denote the current interaction context of the main agent. Hippocam dynamically organizes $\mathcal{C}$ by *intents*, each of which defines a bounded purpose for the agent's ongoing work. We represent an intent as $i = \langle p, \chi, b \rangle$, where p states its purpose, χ specifies its exit conditions, and $b \in \{1, \ldots, n\}$ is the index of the first message governed by the intent. Recursively nesting more specific sub-intents forms an intent stack $\mathcal{I} = \langle i_1, \ldots, i_d \rangle$, from the most encompassing intent i_1 at the bottom to the most immediate i_d at the top. For each intent i_j, all work represented by $m_{b_j}, \ldots, m_n$ serves its purpose p_j. The intent stack may begin after the first message, leaving earlier context outside its scope. Nested intents may begin at the same message, with their order in the stack determining their containment. This nested chain organizes the multiple purposes jointly served by ongoing work in a form aligned with the linear context of LLM agents.

Intent maintenance. Before each main-agent inference, JANUS uses the context and intent stack to determine opening and closing operations and generate guidance. It tracks changes in purpose rather than individual actions, opening nested intents for upcoming work rather than planning the entire task. Intents close from the top of the stack when contextual evidence supports their success or failure exit conditions. These operations combine into four forms of intent maintenance. *Continue* preserves the stack when its intents remain appropriate for the upcoming work. *Deepen* opens more specific intents within the existing chain. *Close* removes a concluded suffix without opening new intents. *Shift* closes a concluded suffix and opens intents for the next direction of work. JANUS injects first-person guidance into the main agent's context before its next inference, specifying which purposes have ended, what to pursue next, and the success or failure exit conditions for that work.

2.3 Intent Consolidation

When JANUS closes one or more intents, PRECIP consolidates their work messages into a self-contained first-person message that replaces them in the context.

Given the complete intent stack with closed intents marked, PRECIP determines what information to retain for subsequent work and at what level of detail. The resulting message preserves established state changes and conclusions with their concrete supporting evidence and provenance, alongside unprocessed material that enclosing intents may need. PRECIP also preserves an impression of the structure and location of consulted material to support targeted inspection in subsequent work. Informative failures are retained, and facts that jointly support a lesson or pattern are integrated into a shared insight. The process by which facts were discovered and material that establishes no fact for any intent in the stack are pruned. This selective consolidation reduces distraction, supports coherent

reasoning, and keeps the context focused and efficient, while preserving the information needed to build on established results without repeating prior work.

2.4 PREFIX CONSOLIDATION

As intents close and new ones open, earlier work messages form a prefix before the new intent stack and become less directly relevant to ongoing work. The same applies within an intent as work progresses into nested intents, leaving earlier messages at enclosing levels farther from the context's end. PALIM consolidates these prefixes to reduce interference from less relevant details and keep the context focused on ongoing work.

Prefix selection. The intent stack's opening boundaries partition $\mathcal{C}$ into regions $\mathcal{C}_j = \langle m_{b_j}, \ldots, m_{b_{j+1}-1} \rangle$ for $j = 0, \ldots, d$, with auxiliary boundaries $b_0 = 1$ and $b_{d+1} = n + 1$. The region $\mathcal{C}_0$ precedes the stack, while $\mathcal{C}_j$ for $j \geq 1$ is local to intent i_j. To preserve details supporting ongoing work, Hippocam protects a global *working tail*, the shortest suffix containing at least K interaction rounds and B tokens. Each round comprises a main-agent response and its tool results; consolidated and intent-guidance messages do not count as rounds. The budget is $B = \min\{\alpha W, \max\{B_{\min}, K\,\mathrm{Quantile}_q(\mathcal{R})\}\}$, using the q-quantile of interaction-round token counts $\mathcal{R}$, a minimum $B_{\min}$, and a cap αW for context capacity W. After each Intent Consolidation, PALIM checks the region containing the resulting messages. It consolidates the portion preceding the working tail only when that portion contains at least $E_{\min}$ intent-consolidated messages and $T_{\min}$ tokens. Under context pressure, PALIM consolidates prefixes preceding the working tail in region order $\mathcal{C}_0, \ldots, \mathcal{C}_d$, regardless of accumulation thresholds. If usage still exceeds capacity, consolidation extends into the working tail.

Recursive consolidation. Inspired by the distinction between episodic and semantic memory in human cognition (Tulving, 1972; Conway, 2001), PALIM separates each consolidated message into narrative and knowledge. The narrative recounts the main agent's work and its outcomes, while knowledge comprises acquired facts and lessons extracted by PRECIP from that work rather than broader lessons generated through consolidation alone. Expressing knowledge directly about its subjects unbinds it from the original narrative, supporting reasoning and decisions beyond specific circumstances. Earlier consolidated messages are recursively consolidated with new work, becoming progressively more abstract and less detailed. Related knowledge is merged, while new facts and lessons supplement or update existing knowledge.

2.5 MESSAGE TREE AND PROGRESSIVE RECALL

Consolidation preserves the original messages and their indices as ordered children of the messages that replace them, forming a message tree whose first layer is the current context. The main agent has access to a recall tool that returns a message's children given its index. It first draws on information already in context and may use this tool to recover missing details that could affect its reasoning or next action. After each recall, it reassesses whether to descend further or stop once the available information is sufficient for ongoing work.

2.6 LEARNING THROUGH USE AND DISUSE

These mechanisms coordinate use and disuse in a continuous cycle through which Hippocam learns from its work and applies what it learns to improve subsequent work (Fig. 3).

Use. The main agent treats consolidated messages as a continuation of its own prior work, drawing on retained facts and lessons to guide ongoing work. It develops approaches by assessing this knowledge against the current task and observations, treating new deductions as hypotheses. Successes and failures provide feedback on their usefulness in the circumstances encountered.

During subsequent Intent Consolidation (§ 2.3), PRECIP consolidates prior knowledge with these observations and outcomes into grounded facts and lessons. Later in Prefix Consolidation (§ 2.4), PALIM integrates these retained and newly acquired facts and lessons with existing knowledge, reinforcing, supplementing, or updating what the agent has learned.

Disuse. In contrast, incidental details and knowledge not currently used are retained in progressively less detail through recursive Prefix Consolidation, reducing their interference with ongoing reasoning and decisions.

The prominence and specificity of retained knowledge therefore evolve through use and disuse, rather than relying on separately assigned importance scores or time-based forgetting rules. These processes turn the agent's own interaction history into evolving knowledge and capabilities, without maintaining a separate memory bank or skill library.

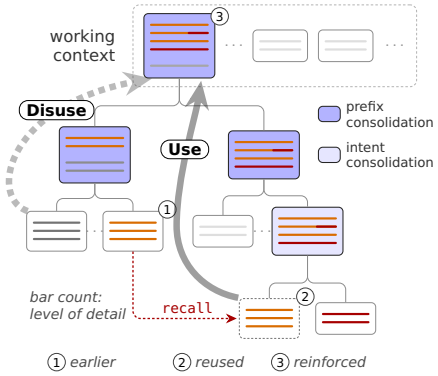


Figure 3: How Hippocam learns. Knowledge: ■ reused; ■ new; ■ fading.

3 RELATED WORK

Context Management. Context management is essential for long-horizon agents. LongLLMLingua (Jiang et al., 2024) compresses context according to query relevance, while SUPO (Lu et al., 2026) learns to condense tool-use histories. Context-Folding (Sun et al., 2026) isolates subtask execution to keep the main context focused. HiAgent (Hu et al., 2025) summarizes completed subgoals, whereas AgentFold (Ye et al., 2026) trains the acting model to fold suffixes of its history. In contrast, Hippocam not only uses explicit intent requirements rather than inferred future needs to guide suffix consolidation and retention granularity, but goes further by bringing earlier prefixes into context management.

Agent Memory. Memory enables agents to maintain continuity and draw on accumulated knowledge across extended interactions. RAG-based memory organizes information as conversational segments, hierarchies, or associative graphs (Pan et al., 2025; Sarthi et al., 2024; Gutierrez et al., 2025). Generative Agents (Park et al., 2023) assigns importance to memories and tracks their recency, while MemoryBank (Zhong et al., 2024) reinforces recalled memories against time-based forgetting. Mem0 (Chhikara et al., 2025) keeps stored facts consistent with new conversations, and A-Mem (Xu et al., 2025) evolves the organization of interconnected memories. MemTree (Rezazadeh et al., 2025) organizes memories into an evolving tree by semantic similarity. Other systems track changes over time (Ong et al., 2025; Banerjee et al., 2026) or learn to manage memory (Zhou et al., 2026; Yu et al., 2026). In Hippocam, context and messages themselves constitute memory, without a separate memory bank or scoring and decay rules. Unlike MemTree, our tree emerges from the organization and consolidation of ongoing work.

Learning from Experience. Recent work increasingly seeks to develop agents that learn continually from experience. Reflexion (Shinn et al., 2023) uses verbal feedback to improve subsequent attempts. Agent Workflow Memory (Wang et al., 2025) accumulates reusable workflows. ReasoningBank (Ouyang et al., 2026) and ACE (Zhang et al., 2026) update reasoning memories and playbooks through task feedback. ETO (Song et al., 2024) and EMPO^2 (Liu et al., 2026) learn from successful and failed trials and memory-guided exploration, respectively, by updating model parameters. Hippocam unifies memory and learning through the dynamics of use and disuse, developing knowledge through work and feedback rather than consolidation alone.

4 EXPERIMENTS

We examine Hippocam's unified cycle of use and disuse through four research questions. **RQ1** Can Hippocam learn from accumulated experience? **RQ2** Can what it learns generalize across task domains? **RQ3** Does memory remain reliable over long interactions? **RQ4** How general is this unified mechanism?

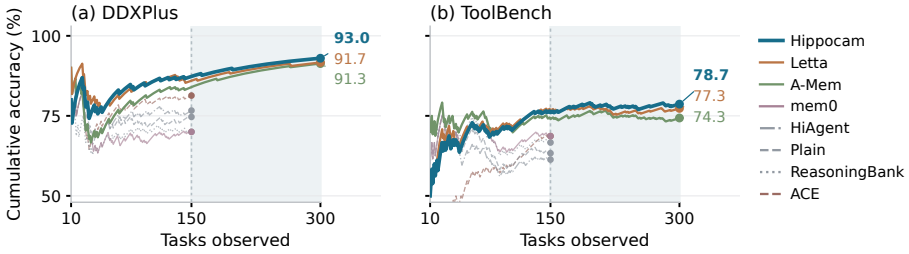


Figure 4: **Performance as experience accumulates.** Cumulative accuracy on StreamBench. Each stream's top three methods at task 150 continue to task 300. Shading marks this extension.

Experimental setup. We compare Hippocam with mem0, A-Mem, Letta, ReasoningBank, ACE, and HiAgent (Chhikara et al., 2025; Xu et al., 2025; Packer et al., 2023; Ouyang et al., 2026; Zhang et al., 2026; Hu et al., 2025) across interactive and memory-intensive tasks. All use DeepSeek-V4.1-Flash (1M-token context window) as the task backbone. We adapt baselines where benchmark support is absent; protocols and implementation details appear in § A.

4.1 RQ1: Learning from Accumulated Experience

We examine whether Hippocam learns from accumulated experience across three benchmarks covering interactive environments and feedback-driven task streams. In ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022), agents accumulate experience over an initial task sequence before proceeding to target tasks, retaining experience while the environment resets between episodes (§ A.1). StreamBench (Wu et al., 2024) comprises two task streams, diagnostic classification on DDXPlus and selection among 50 APIs on ToolBench. Agents receive correctness feedback and a reference answer after each prediction, carrying accumulated experience into subsequent tasks. We evaluate the first 150 tasks in each stream and average their accuracies equally.

Table 1 reports ALFWorld target-task success, ScienceWorld mean target-task progress, and StreamBench mean accuracy. Plain uses the same backbone for in-context reasoning without additional memory or learning mechanisms. Across the three benchmarks, most methods outperform Plain, although some match or fall below it. Hippocam improves across all three benchmarks, achieving 97.8% ALFWorld success and tying ACE for the best result. Its 81.9% mean ScienceWorld progress exceeds four of the six compared methods. On StreamBench, it achieves the highest mean accuracy of 81.7%, narrowly ahead of Letta's 81.3%.

Table 1: Task performance (%) on ALFWorld, ScienceWorld, and StreamBench.

Method	ALF.	Sci.	Stream.
Plain	78.4	66.9	69.0
HiAgent	85.1	56.7	69.0
mem0	87.3	58.6	69.3
A-Mem	95.5	87.2	79.0
Letta	85.1	71.2	81.3
ReasoningBank	85.1	66.9	68.3
ACE	97.8	**91.9**	75.0
Hippocam	**97.8**	81.9	**81.7**

Figure 4 extends each StreamBench stream's three leading methods at task 150 to task 300, tracking cumulative accuracy over all tasks answered so far. On DDXPlus, Hippocam continues to improve after early fluctuations and maintains its lead, while Letta and A-Mem also improve. On ToolBench, Hippocam makes modest further gains amid fluctuations and moves ahead of Letta, whereas A-Mem shows little sustained improvement in the extended evaluation. These gains demonstrate more effective learning from accumulated experience than the compared methods. Detailed scores at both horizons appear in Table 6. In both streams, Hippocam also achieves higher accuracy with about 10% lower average cost per task in the later 150-task interval (§ A.9).

4.2 RQ2: Generalization across Task Domains

We examine whether acquired knowledge generalizes to a new domain and can be integrated with experience accumulated there. Agents carry memory acquired through household tasks in ALFWorld to scientific experiments in ScienceWorld, and through multi-hop question answering in HotpotQA (Yang et al., 2018) to claim verification in FEVER (Thorne et al., 2018). *Frozen* evaluation tests knowledge generalization by evaluating each target-domain task independently from the same source-domain memory snapshot. *Sequential* evaluation tests knowledge integration by running the same

Table 2: **Cross-domain knowledge generalization and integration.** ScienceWorld mean progress and FEVER accuracy (%). Δ_{stream} reports After minus Fresh in percentage points (pp).

	ALFWorld → ScienceWorld			HotpotQA → FEVER		
Method	Frozen	Fresh → After	Δ_{stream}	Frozen	Fresh → After	Δ_{stream}
Plain	48.8	—	—	59.3	—	—
HiAgent	42.7	49.6 → 49.7	+0.1	**61.8**	57.9 → 52.1	-5.8
mem0	51.7	50.7 → 22.6	-28.1	59.9	57.9 → 57.9	+0.0
A-Mem	42.1	73.9 → 65.9	-8.0	55.8	55.8 → 55.8	+0.0
Letta	40.8	59.3 → 54.5	-4.8	59.9	69.8 → 59.9	-9.9
ReasoningBank	53.3	48.6 → 37.4	-11.2	55.8	57.9 → 55.8	-2.1
ACE	62.7	52.5 → 66.5	+14.0	57.9	61.8 → 63.8	+2.0
Hippocam	**78.9**	55.9 → 75.1	+19.2	**61.8**	53.9 → 59.9	+6.0

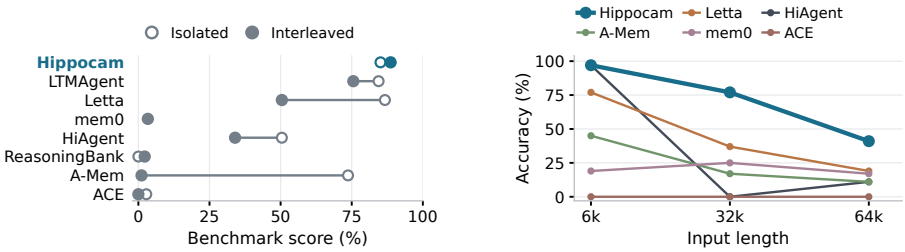


Figure 5: **Maintaining usable experience.** *Left:* GoodAI LTM scores under isolated and 32k-span interleaved conditions. *Right:* MemoryAgentBench multi-hop conflict-resolution accuracy across input lengths. Lines connect separate length conditions.

target-domain task sequence with source-domain memory (*After*) or an initially empty memory (*Fresh*). In both conditions, experience accumulates across successive target tasks. The After minus Fresh score difference, Δ_{stream}, measures whether source-domain knowledge supports subsequent learning in the target domain. Table 2 reports mean target-task progress on ScienceWorld and accuracy on FEVER. Evaluation details appear in § A.2.

Knowledge generalization. On ScienceWorld, several methods benefit from source-domain experience, but the gains vary substantially. Hippocam achieves 78.9% mean progress, improving over Plain by 30.1 pp and exceeding the strongest baseline, ACE, by 16.2 pp. On FEVER, Frozen scores cluster around Plain's 59.3%, with Hippocam and HiAgent reaching 61.8%. A label-wise analysis suggests limited room for improvement on supported and refuted claims, where Plain already performs strongly, while most errors concern insufficient evidence. This pattern suggests that source-domain experience provides limited guidance for the remaining target-domain difficulty.

Integrating prior knowledge with new experience. Only Hippocam and ACE achieve positive gains on both domain transitions, while the remaining methods show mixed, unchanged, or reduced performance. ACE improves ScienceWorld progress by 14.0 pp and FEVER accuracy by 2.0 pp. Hippocam achieves larger gains of 19.2 and 6.0 pp, respectively, with a narrower FEVER gap under the scoring sensitivity analysis (§ A.2). These comparisons highlight Hippocam's stronger ability to use prior knowledge to support continued learning in a new domain.

4.3 RQ3: Memory Reliability over Long Interactions

For accumulated experience to support continued learning, agents must remain effective across task switches, reason from past interactions, and retain information as it is updated. We examine interleaved work in GoodAI LTM (Castillo-Bolado et al., 2024), reasoning over past conversations in LoCoMo (Maharana et al., 2024), and resolving updated facts in MemoryAgentBench (Hu et al., 2026). Appendix A.3 provides evaluation protocols and reference configurations.

Interleaved work. We use GoodAI LTM with a 32k-token memory span for the interleaved condition and compare against isolated evaluation. We additionally compare against LTMAgent, the long-term memory agent developed by the benchmark authors (Castillo-Bolado et al., 2024). LTMA-

Table 3: **What the retained conversation supports.** Binary judge accuracy (%) on LoCoMo. Evaluation scope is specified in § A.3.

Method	Overall	Single-hop	Multi-hop	Temporal	Open-domain	Adversarial
HiAgent	39.8	37.1	24.8	16.2	45.8	70.2
mem0	58.0	67.2	37.6	8.1	77.1	85.2
A-Mem	67.5	74.3	34.4	67.6	61.5	76.7
Letta	51.7	48.5	22.0	35.2	45.8	89.5
ReasoningBank	18.3	2.9	0.0	2.8	7.3	72.4
ACE	22.5	0.0	0.0	0.0	0.0	100.0
Hippocam	**77.6**	81.5	40.8	86.6	77.1	87.2

gent, Letta, and HiAgent all lose performance when tasks are interleaved. In contrast, Hippocam maintains high performance in both conditions without degradation under interleaving, scoring 85.1% in isolation and 88.7% under interleaving (Fig. 5, left). This contrast highlights Hippocam's stronger resilience to interference from interleaved tasks.

Reasoning from past conversations. On LoCoMo, Hippocam achieves 77.6% overall accuracy, exceeding the strongest baseline, A-Mem, by 10.1 pp (Table 3). It leads or ties across all four answerable question categories, with temporal accuracy reaching 86.6% compared with A-Mem's 67.6%. By contrast, ACE's 100% adversarial accuracy accompanies zero accuracy on answerable questions, reflecting frequent abstention rather than effective recovery of conversational knowledge. Hippocam combines stronger conversational reasoning with 87.2% accuracy on unanswerable questions.

Preserving updated information. MemoryAgentBench tests reasoning over revised facts, with conflict-resolution rules provided only at question time. Across the three input lengths, Hippocam achieves 71.7% mean accuracy, compared with 44.3% for Letta (Fig. 5, right). It matches HiAgent at 97.0% on 6k inputs and retains 77.0% and 41.0% accuracy at 32k and 64k, where no baseline exceeds 37.0% and 19.0%, respectively. Although longer histories remain challenging, Hippocam more effectively preserves and applies updated information.

4.4 RQ4: Generality of the Unified Memory–Learning Mechanism

Figure 1 reveals distinct capability profiles across the evaluated methods. Memory-oriented methods support more than recall, with A-Mem performing strongly on ScienceWorld (87.2%) and Letta on StreamBench (81.3%). mem0 also supports learning, transfer, and conversational memory, although it struggles with interleaved work (3.3%). By contrast, ACE excels on ScienceWorld (91.9%) but, like the similarly guidance-oriented ReasoningBank, struggles on LoCoMo (22.5% and 18.3%, respectively). This gap suggests that experience-derived guidance does not necessarily preserve the specific information needed for later recall. HiAgent exhibits a more balanced profile through subgoal-based context management, suggesting that organizing working context can support a broader range of demands.

Hippocam combines this breadth with stronger overall performance, leading or tying on seven of the eight measures. Memory, knowledge development, and learning arise from the same use-and-disuse cycle over its interaction history, demonstrating the broad applicability of a unified mechanism rather than separate capability-specific mechanisms.

5 Analysis

5.1 Ablation Studies

We examine intent-conditioned consolidation, prefix consolidation, and recall in settings that exercise their respective roles. Appendix A.4 specifies the interventions and additional comparisons.

Intent-conditioned consolidation. We remove explicit intent conditioning from consolidation while retaining intent boundaries and Janus guidance. On GoodAI, filler retention rises from 14% to 63%, while scores fall by 23.1 pp under interleaving versus 7.3 pp in isolation (Table 4). A complementary case shows that intent conditioning preserves individual menu options needed for ordering rather than only category ranges (§ A.5). These observations support intent-aware selection of both relevant content and the detail needed for ongoing work.

Prefix consolidation. We remove PALIM and truncate older prefixes when context pressure requires space, while retaining intent consolidation and recall. ToolBench accuracy falls from 78.7% to 65.7% (Table 4). ALFWorld success remains near perfect, but mean actions increase from 9.62 to 11.74, and episode-start context grows from 19.7k to 50.7k tokens in the inspected stream. Prefix consolidation thus supports more effective subsequent work with a smaller active context.

Table 4: **Component ablations and consolidation comparison.** Scores, success rates, and accuracy are percentages. Context size is measured in tokens.

Setting	Metric	Hippocam	w/o
Intent conditioning			
GoodAI isolated	Score ↑	85.1	77.8
interleaved	Score ↑	88.7	65.6
Prefix consolidation			
ToolBench	Acc. ↑	78.7	65.7
ALFWorld	Success ↑	97.8	100.0
	Steps ↓	9.62	11.74
	Context	19.7k	50.7k
Recall			
MAB-64k	Acc. ↑	41.0	27.0
Prefix consolidation variant			w/ playbook
ALFWorld	Success ↑	100.0	97.8
	Steps ↓	8.12	9.00

Recall. Disabling recall on the same post-ingestion MemoryAgentBench 64k snapshot reduces accuracy from 41.0% to 27.0%. Access to underlying evidence therefore improves reasoning beyond what the consolidated context alone supports.

5.2 LEARNING THROUGH EVOLVING EXPERIENCE

Is prior strategy induction necessary? The competitive learning performance of memory-oriented methods in § 4.4 suggests that retained knowledge can support new work without first being distilled into reusable strategies. To examine this more directly, we replace Prefix Consolidation with ACE's playbook-generation process in ALFWorld (Zhang et al., 2026). Both configurations achieve near-perfect task success, while the playbook variant uses slightly more actions on average (9.00 versus 8.12). Inducing a playbook therefore offers no clear advantage here over directly using retained facts and grounded lessons (Table 9).

Recovering enough detail to proceed. In the MemoryAgentBench cases (Fig. 8), one question requires two levels of recall to recover a chain of facts, while another is resolved after one level without opening deeper references. These cases show that Hippocam adapts retrieval depth to the task, recovering sufficient evidence without expanding the entire history.

Learning through use and disuse. Our analysis of knowledge changes across consecutive consolidations finds that items linked to intervening work are more often supplemented or updated (§ A.6). ALFWorld traces show repeated successful desklamp use supporting a lesson across objects and instruction wordings, and later successful searches revising earlier failure accounts to attribute them to search cost (§ A.7). Meanwhile, details of a routine placement episode recede through recursive consolidation (Fig. 7). These observations show how use and disuse sustain a continuous cycle of learning through work.

6 CONCLUSION

We introduced Hippocam, a unified architecture for memory, knowledge, and learning through the continual consolidation and selective recall of interaction history. Experiments and analyses demonstrate its broad applicability and illuminate how retained knowledge evolves through use and disuse. Our findings suggest a path toward agents that continually learn from their own work.

AI USE STATEMENT

We used AI tools to accelerate code development, assist literature searches, polish the manuscript, and prepare figures.

REFERENCES

Pratyay Banerjee, Masud Moshtaghi, Shivashankar Subramanian, Amita Misra, and Ankit Chadha. APEX-MEM: Agentic semi-structured memory with temporal reasoning for long-term conversational AI. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16470–16489, 2026. doi: 10.18653/v1/2026.acl-long.749. URL https://aclanthology.org/2026.acl-long.749/.

David Castillo-Bolado, Joseph Davidson, Finlay Gray, and Marek Rosa. Beyond prompts: Dynamic conversational benchmarking of large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 42528–42565, 2024. doi: 10.52202/079017-1347. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4aedf0cba303537fcb6cf948bb41b2df-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

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: 28th European Conference on Artificial Intelligence*, 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.

Martin A. Conway. Sensory-perceptual episodic memory and its context: Autobiographical memory. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 356 (1413):1375–1384, 2001. doi: 10.1098/rstb.2001.0940.

Yadin Dudai, Avi Karni, and Jan Born. The consolidation and transformation of memory. *Neuron*, 88 (1):20–32, 2015. doi: 10.1016/j.neuron.2015.09.004.

K. Anders Ericsson and Walter Kintsch. Long-term working memory. *Psychological Review*, 102(2): 211–245, 1995. doi: 10.1037/0033-295X.102.2.211.

Bernal Jimenez Gutierrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025. URL https://aclanthology.org/2025.acl-long.1575/.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fd1eff9dd295df50a41f2521942fa31d-Abstract-Conference.html.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024. URL https://aclanthology.org/2024.acl-long.91/.

Janet L. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34, 1992. doi: 10.1007/BF00155578.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. Exploratory memory-augmented LLM agent via hybrid on- and off-policy optimization. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b589d92785e39486e978fa273d0dc343-Abstract-Conference.html.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Beyond the context window: Scaling agentic RL via end-to-end optimized context compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21074–21125, 2026. URL https://aclanthology.org/2026.acl-long.966/.

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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Earl K. Miller and Jonathan D. Cohen. An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, 24:167–202, 2001. doi: 10.1146/annurev.neuro.24.1.167.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. Towards lifelong dialogue agents via timeline-based memory management. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8631–8661, 2025. doi: 10.18653/v1/2025.naacl-long.435. URL https://aclanthology.org/2025.naacl-long.435/.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/980ea04d23d1f6908964eba2a74afe45-Abstract-Conference.html.

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.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. SeCom: On memory construction and retrieval for personalized conversational agents. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e56f394bbd4f0ec81393d767caa5a31b-Abstract-Conference.html.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23. Association for Computing Machinery, 2023. doi: 10.1145/3586183.3606763.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for LLMs. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.14052.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2401.18059.

Michael N. Shadlen and Daphna Shohamy. Decision making and sequential sampling from memory. *Neuron*, 90(5):927–939, 2016. doi: 10.1016/j.neuron.2016.04.036.

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://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. URL https://aclanthology.org/2024.acl-long.409/.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon agent via context folding. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=lNRgWoGfYg.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074/.

Endel Tulving. Episodic and semantic memory. In Endel Tulving and Wayne Donaldson (eds.), *Organization of Memory*, pp. 381–403. Academic Press, New York, 1972.

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. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. StreamBench: Towards benchmarking continuous improvement of language agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 107039–107063, 2024. doi: 10.52202/079017-3398. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c189915371c4474fe9789be3728113fc-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2502.12110.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Jingren Zhou, Siheng Chen, and Yong Jiang. AgentFold: Long-horizon web agents with proactive context folding. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.24699.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026. URL https://aclanthology.org/2026.acl-long.981/.

Dagmar Zeithamova, April L. Dominick, and Alison R. Preston. Hippocampal and ventral medial prefrontal activation during retrieval-mediated learning supports novel inference. *Neuron*, 75(1): 168–179, 2012. doi: 10.1016/j.neuron.2012.05.010.

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. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a94ff6f922d995d7d3f4ebf4143e442-Abstract-Conference.html.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5fc8b3bdfbb9167b5144df5d3fae4616-Abstract-Conference.html.

A Experimental Protocols and Additional Results

Scope of the evaluation. We report task-level scores for each evaluated configuration, using the same scoring rule across methods within each benchmark. Results describe single runs.

Backbone and interfaces. The task backbone is DeepSeek-V4.1-Flash, accessed through the local identifier `deepseek-flash`. Model-assisted scoring uses DeepSeek-V4-Pro. Hippocam uses the same model family for its auxiliary components. The comparison is not matched by total inference compute or auxiliary output budget. In interactive environments, observations arrive as tool results and the environment determines completion; reaching an action limit or exhausting the no-action continuation allowance is a failure. Hippocam's StreamBench configuration exposes its normal tools, whereas the other arms answer through the benchmark interface. In the QA transfer experiment, Hippocam accesses the same Wikipedia service through a command-line wrapper; the comparison systems use search and lookup calls. Memory-only Hippocam configurations retain `recall` while disabling filesystem and shell tools.

Comparison implementations. We use the upstream HiAgent code (commit `cebdd8e`), connected to the task model and environments through an adapter. The agent is reset only at the beginning of a task stream; its session persists across subsequent tasks and environment episodes. Its prompt context limit is 8,192 tokens, with output allowances of 4,096 tokens for interactive environments and 16,384 for StreamBench, and a 1,800-second per-task time limit. The upstream code includes a subsequently added summarization module, so this is an evaluation of that implementation rather than an exact reconstruction of the original paper's code version.

Letta runs with reasoning disabled because of its tool-call protocol. mem0 and A-Mem disable reasoning during memory extraction; A-Mem also allows up to three attempts at structured extraction. mem0 uses an expanded extraction output allowance and, in MAB, an adapted general-purpose extraction prompt. These settings differ from the upstream defaults.

ACE uses the upstream Generator, Reflector, and Curator components (commit `82709de`) in online mode. Interactive episodes use the environment's action loop followed by reflection and playbook curation. StreamBench uses a 15-task online evaluation window and up to three reflection rounds with the benchmark's feedback. For conversation and fact-ingestion tasks, incoming text passes through generation, reflection, and curation; subsequent questions are answered from the resulting playbook, without an additional fact store. All three roles use DeepSeek-V4.1-Flash.

Hippocam consolidation parameters. Table 5 reports the implementation defaults for the consolidation mechanism in § 2.4. The working-tail token budget uses the 95th percentile of observed main-agent action-round lengths, counting each response together with its tool results. The configured context capacity W includes input and output; pressure control accounts for the current request and reserves space for the next interaction round. These thresholds control when context is consolidated, rather than assigning importance or decay scores to individual memories.

Table 5: **Default consolidation configuration.** Token quantities are implementation budgets; W is the configured context capacity.

Parameter	Default	Role
K	2	Minimum working-tail interaction rounds
$B_{\min}$	4,096 tokens	Working-tail token floor
α	0.20	Working-tail token cap as a fraction of W
q	0.95	Action-round length quantile
$E_{\min}$	2	New intent consolidations in a stable prefix
$T_{\min}$	16,384 tokens	Stable-prefix size for ordinary consolidation
W	128,000 tokens	Total context capacity
Pressure threshold	$0.80W$	Predicted occupancy that initiates a sweep
Capacity threshold	$1.00W$	Predicted occupancy limit before inference
Minimum token reduction	8,192 tokens	Required saving from a prefix consolidation
Compression retries	2	Retries when the required reduction is not met

Table 6: **StreamBench accuracy at two evaluation horizons.** Mean gives equal weight to the two streams and is reported only when both reach the stated horizon.

	First 150 tasks per stream			First 300 tasks per stream		
Method	DDXPlus	ToolBench	Mean	DDXPlus	ToolBench	Mean
Plain	74.7	63.3	69.0	—	—	—
HiAgent	76.7	61.3	69.0	—	61.3	—
mem0	70.0	68.7	69.3	—	69.0	—
A-Mem	84.0	74.0	79.0	91.3	74.3	82.8
Letta	86.0	**76.7**	81.3	91.7	77.3	84.5
ReasoningBank	70.0	66.7	68.3	—	—	—
ACE	81.3	68.7	75.0	—	—	—
Hippocam	**87.3**	76.0	**81.7**	**93.0**	**78.7**	**85.8**

Means weight the two streams equally at the same horizon. Dashes denote unreported results.

A.1 Experience Accumulation

ALFWorld uses distinct source episodes and a 30-action limit per episode. Target success is measured over unique tasks; preceding memory includes repeated-task experience.

ScienceWorld reports mean target progress, with negative environment scores clipped to zero and percentages normalized to $[0, 1]$ before aggregation. The evaluation pairs source and target variations within the evaluated ScienceWorld task types, with a 50-action limit. Source and target episodes are kept separate in all reported scores. Table 1 averages the two StreamBench accuracies with equal weights over the first 150 tasks in each stream. Task streams preserve memory across tasks. Figure 4 extends the three highest-scoring methods at task 150 in each stream to task 300. Table 6 retains all available valid 300-task results, including mem0 and HiAgent on ToolBench. Means use the same horizon for both streams.

A.2 Source Initialization and Cross-Domain Evaluation

The sequential ScienceWorld comparison pairs target identifiers between the source-initialized and empty-memory conditions, with a 30-action limit. Source memory contains experience from ALFWorld source and target episodes. HotpotQA→FEVER transfers experience from source questions to target claims, with a seven-step task limit. Both conditions accumulate target-domain experience across tasks. Consequently, their difference is a contrast between two learning streams, not a measure of source-only knowledge on independent tasks. Sequential differences use paired task identifiers; mem0 and ReasoningBank have narrower matched ScienceWorld coverage than the other methods.

Frozen-source evaluation restores the same source snapshot before each target task and clears the task workspace. It permits reasoning and memory changes within the current task but prevents those changes from carrying to the next target task. Plain evaluates target tasks independently without additional mechanisms, providing a reference for Frozen evaluation. Evaluation costs include target-task calls and within-task memory operations, excluding source experience acquisition.

FEVER scoring sensitivity. Under an alternative answer extraction, Hippocam's sequential improvement decreases from six to two percentage points. The main table retains the original scoring.

A.3 Retention, Interference, and Conflicting Information

GoodAI uses the benchmark's own task scores and its isolated and interleaved modes. Model-assisted scoring steps use DeepSeek-V4-Pro to extract structured answers for the benchmark's checks. We report the sum of awarded scores divided by the sum of their corresponding maximum scores. We use the benchmark's 32k memory-span configuration. The interleaved mode additionally requests extremely brief answers; the modes also differ in reset behavior and wait-action handling. We

Table 7: **Frozen-source performance and recorded target-evaluation cost.** Performance is reported in percentages; cost is the total USD expenditure for the reported target evaluation, including within-task memory operations and excluding source experience acquisition.

	ScienceWorld		FEVER	
Method	Progress	Cost	Accuracy	Cost
HiAgent	42.7	2.62	61.8	1.76
mem0	51.7	0.06	59.9	0.25
A-Mem	42.1	0.26	55.8	0.35
Letta	40.8	2.53	59.9	0.98
ReasoningBank	53.3	0.10	55.8	0.19
Hippocam	78.9	2.27	61.8	0.78

Table 8: **MemoryAgentBench conflict resolution and long-range understanding (LRU).** Accuracy (%); Overall averages the three multi-hop input lengths.

Method	6k	32k	64k	Overall	DetectiveQA
HiAgent	97.0	0.0	11.0	36.0	—
Hippocam	97.0	77.0	41.0	71.7	85.9
Letta	77.0	37.0	19.0	44.3	82.4
A-Mem	45.0	17.0	11.0	24.3	78.9
mem0	19.0	25.0	17.0	20.3	—
ACE	0.0	0.0	0.0	0.0	—

interpret the comparison as performance under the two benchmark protocols, without attributing the higher interleaved score to task interleaving itself.

The LoCoMo results cover the full ten-conversation release. Sessions are presented in order; questions are answered from copies of the post-ingestion state so that answers do not accumulate across questions. The main table reports binary correctness assigned by DeepSeek-V4-Pro, not the original benchmark's lexical F1. Adversarial questions reward identifying unsupported premises, while the remaining categories require substantive answers. The same judge protocol is used across the reported arms.

For answerable questions, our judge receives the question, reference answer, and extracted model answer. It accepts semantically equivalent wording, requires dates to match the reference's precision, and requires list answers to contain the same items. For adversarial questions, it receives the question, the supplied misleading answer, and the model answer; correctness requires identifying that the requested information is absent. The conversation itself is not included in either judge input. The judge returns a binary `CORRECT`/`WRONG` label after a brief rationale.

MemoryAgentBench uses incremental ingestion followed by independently evaluated questions from the post-ingestion snapshot. The conflict-resolution results use the full 100-question set at each multi-hop input length. Answers use the benchmark's substring-matching criterion. The main Hippocam configuration consolidates at the longer lengths; HiAgent uses its upstream subgoal summaries and prompt-context limit. These are complete-system comparisons with different retained representations and memory budgets.

DetectiveQA evaluates reasoning over ten novel-length memories containing 71 questions in total; scores count questions rather than ingested documents.

The separate 32k-window experiments terminate after capacity failures and are not pooled with the main results.

Mechanistic interpretation. The inspected MAB failures include losing updates during consolidation and selecting an obsolete value despite retaining relevant evidence. These observations

distinguish information loss from failures to use retained information; the complete-system comparisons do not isolate individual memory mechanisms.

A.4 Ablations and Consolidation Comparisons

Intent-conditioning ablation. The GoodAI variant removes the intent-specific retention instructions and the explicit intent chain in the consolidation trigger. Consolidation still occurs at intent boundaries; Janus's focus and completion messages remain in the conversation. The intervention therefore removes explicit intent conditioning of content selection, while preserving the intent controller and consolidation mechanism. Both conditions use the same evaluation scope and scoring as the GoodAI comparison in RQ3. The interleaved score decreases from 88.7% to 65.6%, compared with 85.1% to 77.8% in isolation. The paired two-sided sign test for interleaved task scores gives $p = 0.11$; each condition is a single run.

We inspect whether filler question–answer pairs remain in the memories that consolidate them; retention increases from 14% to 63% without intent conditioning. This measures retained filler content rather than total memory size. In SallyAnne cases, the variant incorporates the beginning of a story into a memory associated with another task, while later events remain in the active conversation. The agent answers with the object's current location rather than the location known to the character; the underlying events remain represented, so this trace alone does not establish that their order has been lost. Other score differences involve answer behavior or grading, so the aggregate gap cannot be attributed entirely to consolidation. Excluding the Jokes category, whose grader can credit a list containing the requested joke, preserves the interleaved advantage (91.1% versus 72.8%).

Online prefix-consolidation ablation. The variant removes prefix consolidation and truncates older prefixes when context pressure requires space, without archiving or consolidating the truncated content. Intent consolidation and recall remain available. Each run resumes from the complete system's checkpoint immediately before its first prefix consolidation, using the same task order and driver configuration. The reported ToolBench score covers the first 300 tasks, including the common initial segment; ALFWorld success and mean actions use the target tasks reported in RQ1. Context-size statistics describe the first main-agent request of each episode in the inspected ALFWorld stream, including its preceding experience-acquisition segment; they do not measure total inference tokens.

The intervention tests removing prefix consolidation as a whole, including its timing and representation.

Recall ablation. The complete system and the variant answer from copies of the same post-ingestion MemoryAgentBench 64k snapshot. The variant removes the recall tool and its accompanying instructions; ingestion and stored memory are unchanged. We report question-paired accuracy and a two-sided exact test on discordant outcomes. The comparison uses one evaluated answer per question and condition. Protocol failures following recall remain counted as incorrect in the complete-system score.

Prefix consolidation versus playbook generation. The ALFWorld playbook variant replaces Prefix Consolidation with ACE's playbook-generation process, using its reflection and curation components. The intervention changes how retained knowledge is produced during work, rather than substituting playbook entries into an already consolidated memory.

Additional comparisons include an empty archive placeholder and a flat summary generated with Letta's summarization prompt. The latter changes both the representation and its length.

Task instances have distinct game files from the preceding experience, although some share a task and room configuration with earlier episodes and vary object placement. We therefore interpret this comparison as reuse within the domain. Success rates are near ceiling; action counts provide an exploratory view of differences in the work required. The observed similarity between the complete system and playbook variant success rates is not an equivalence test.

Table 9: **Prefix consolidation and alternative knowledge configurations.** ALFWorld success and mean actions. The ACE variant replaces Prefix Consolidation with its playbook-generation process.

Configuration	Success (%)	Mean steps
Hippocam	100.0	8.12
Empty placeholder	94.0	9.92
ACE playbook	97.8	9.00
Flat summary	100.0	9.68

A.5 Intent and the Granularity of Retained Information

Figure 6 gives an illustrative Restaurant case outside the aggregate GoodAI comparison. Both runs received a detailed menu. Intent-guided consolidation retained the individual dishes, while the variant without intent guidance retained one menu memory chiefly as category names and number ranges. When later asked to order, the former selected a listed dish and the latter named a dish absent from the menu. This illustrates a retention-granularity contrast rather than isolating the cause of the response.

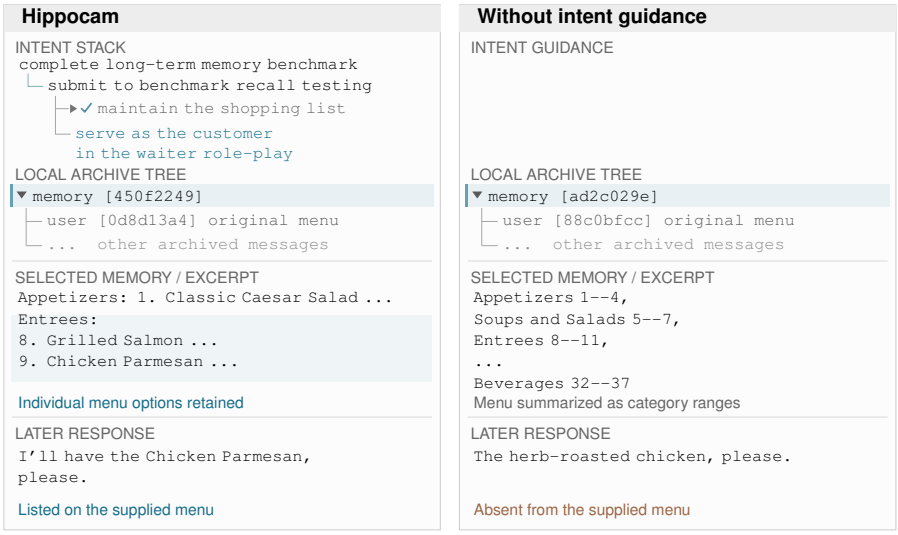


Figure 6: **Retained detail for a later action.** Local archive subtrees redrawn in the style of the Hippocam viewer, with verbatim memory and response excerpts. The left intent trace shows the completed shopping-list intent and the new customer intent, with labels shortened for display; node IDs identify saved records and ellipses mark omissions. The selected memories retain different levels of menu detail.

A.6 Knowledge Changes across Consecutive Consolidations

We track each knowledge item from one consolidation to the next within a consolidation chain and group these item transitions by evidence of use during the intervening work. Each transition is classified as unchanged, supplemented, updated, rewritten, or dropped. Supplementation adds information, updating replaces a concrete value, and rewriting changes wording, trims content, or merges items. Dropped items are absent from the next consolidated representation, which does not imply deletion of the underlying interactions. An item can contribute multiple transitions across successive consolidations, so transition counts do not represent distinct items or independent samples.

API-linked use in ToolBench. The ToolBench analysis covers two consolidation chains and 3,086 item transitions. An item is marked as used when an intervening task concerns an API mentioned in the item or the main agent calls that API. This is an API-linked proxy for use, not a direct observation

that the item influenced reasoning. Used items are divided according to whether the main agent's call exactly matches the reference answer or the reference specifies a different call. Items mentioning no API remain unclassified rather than being counted as unused.

Table 10: Knowledge-item transitions in ToolBench grouped by API-linked use between consecutive consolidations. Outcome columns report percentages within each group.

Intervening use	Unchanged	Supplemented	Updated	Rewritten	Dropped
No detected use	95.8	0.9	0.1	3.1	0.1
No API, unclassified	94.1	2.1	0.3	3.4	0.1
Used, call matches reference	57.0	28.4	0.3	14.3	0.0
Used, reference call differs	68.3	11.7	3.4	16.6	0.0

Supplementation occurs in 28.4% of matched-call transitions and 11.7% of differing-call transitions, compared with 0.9% of transitions without detected use. Concrete-value updates are more frequent when the reference specifies a different call (3.4%) than when the call matches (0.3%) or no use is detected (0.1%). These associations connect subsequent knowledge changes to intervening work and feedback, without establishing that use alone causes the changes. Most items without detected use remain unchanged (95.8%), so these adjacent-consolidation statistics do not establish general fading of unused knowledge.

Textual reuse in interactive environments. The ALFWorld analysis covers one chain with 23 consolidations, while the ScienceWorld analysis covers five consolidations. Here the proxy is weaker, marking an item as referenced if a contiguous four-word fragment from it occurs in text written by the main agent between consolidations. Textual overlap need not imply substantive use, and its absence does not exclude paraphrased or implicit use. Only unchanged proportions are available for these analyses, so we do not infer how the remaining transitions divide among supplementation, updating, rewriting, and dropping.

Table 11: Knowledge-item transitions grouped by four-word textual reuse in main-agent messages. Unchanged proportions are percentages.

Environment	Textual reuse	Unchanged (%)
ALFWorld	Detected	41.6
	Not detected	63.2
ScienceWorld	Detected	57.7
	Not detected	59.1

In ALFWorld, referenced items remain unchanged less often than items without detected textual reuse (41.6% versus 63.2%). The corresponding ScienceWorld difference is small (57.7% versus 59.1%). The association is therefore clearer in ALFWorld than in ScienceWorld under this textual proxy.

A.7 Case Evidence for Evolving Memory

Figure 7 follows two ALFWorld memories through repeated prefix consolidation. The desklamp item begins as a concrete operation, accumulates confirmations from different object classes and phrasings, and is later represented without enumerating those classes. Meanwhile, an ordinary remotecontrol episode is first retained with its destination, then reduced to an item in a list of completed tasks, after which its episode-specific details no longer occur in the narrative. Within this run, reusable knowledge persists as the details of individual episodes recede. The displayed text is abridged and, where quotation marks are absent, paraphrased from the stored memories.

A second lineage shows how new outcomes revise earlier interpretations. Early searches for an apple and a potato exhausted the action budget without finding either object. After later episodes found the same object classes and completed related tasks, the retained account came to attribute the earlier failures to search cost. Further failed and successful searches then entered consolidation alongside this revised account, connecting knowledge updates to continuing experience.

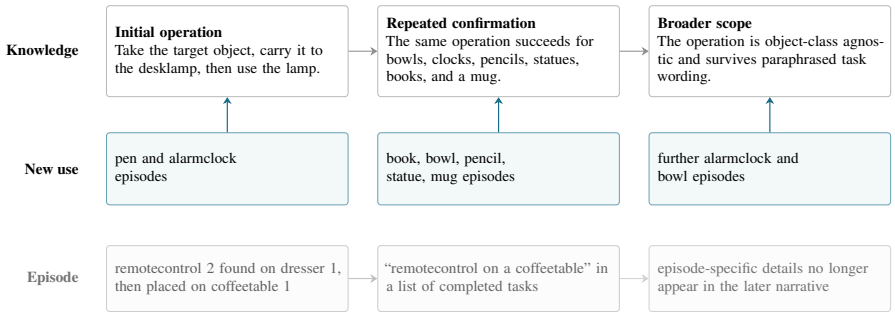


Figure 7: **Experience changes through use and disuse.** Knowledge gains supporting evidence as episode-specific narrative detail recedes. Abridged and paraphrased from ALFWorld traces.

A.8 Progressive Recall Procedures

Figure 8 traces two successful questions from the synthetic facts supplied by MemoryAgentBench; the displayed author, language, birthplace, and continent statements are benchmark records rather than claims about the world. For q042, one recall round exposes all records needed under the benchmark's later-entry rule, so a deeper reference exposed by the fold remains closed. For q014, the first round finds one authorship record but leaves the competing authorship and language records behind a nested reference; a second round opens that reference. Two consolidation folds then carry the recalled evidence into the answering context, and Janus marks the evidence-gathering intentions complete before the response. Thus the procedure unfolds detail until the current question is supported, while leaving unrelated children unopened.



Figure 8: Progressive recall in two MemoryAgentBench conflict-resolution cases. Blue boxes are opened memory; dashed gray boxes are references or children left unopened. The second case preserves the observed reconsolidation and completion check between recall and answer.

A.9 The Cost of Maintaining and Using Experience

Here we examine the cost of maintaining and using that experience on the same StreamBench runs, where both task execution and memory maintenance are metered. We first compare methods over the first 150 tasks, then follow performance and cost through the following tasks.

Accounting. We reprice recorded usage with a common schedule: USD 0.00596 per million cached input tokens, 0.29806 per million uncached input tokens, 1.19225 per million output tokens, and 0.07452 per million embedding tokens. Costs include auxiliary model calls and memory writes, and exclude evaluation judges. They are standardized usage costs rather than reconstructed invoices. Input totals sum all calls, including repeated reads of the same history. Letta records total input but not cached input, so we retain its token counts in Table 13 without assigning it an exact cost. All other plotted methods use their recorded execution and memory-maintenance usage; their task and auxiliary inference configurations follow the evaluated implementations.

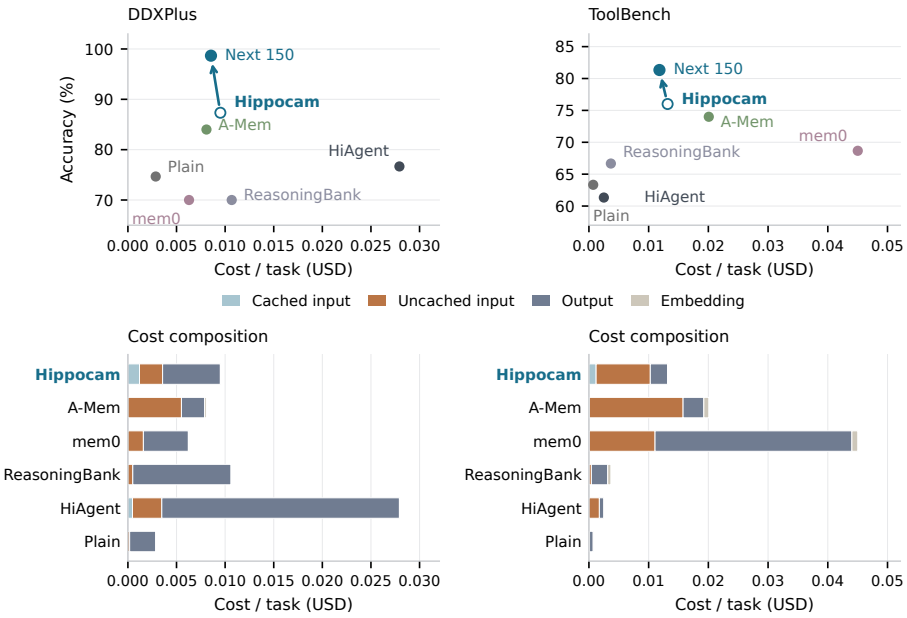


Figure 9: **Performance, reuse, and the cost of continued work.** *Top:* accuracy versus average cost per task on each stream's first 150 tasks; upper-left is preferable. The blue arrow connects Hippocam's first 150 tasks (open marker) to its next 150 (filled marker); each endpoint reports its own interval. *Bottom:* the first-150 cost decomposed into cached input, uncached input, output, and embeddings. All components include memory maintenance.

Later work need not cost more per task. Across both streams, Hippocam's later interval has higher accuracy and about 10% lower mean cost per task (Table 12). This occurs despite an increase in total input, showing that repeated use of a growing history need not translate directly into a higher bill. The intervals contain different tasks, so these changes describe the observed runs rather than isolate a causal effect of learning on cost. Across methods, Fig. 9 shows a trade-off: several lower-scoring methods are cheaper, while Hippocam combines higher accuracy with lower cost than A-Mem and mem0 on ToolBench. On DDXPlus, those two methods remain cheaper; the cost advantage is therefore specific to the workload.

Stable prefixes make retained experience reusable. On ToolBench's first 150 tasks, Hippocam reads 232.5k input tokens per task, compared with A-Mem's 65.0k, but only 30.4k versus 52.7k are uncached. Its cost per correct answer is consequently lower under the stated prices (0.0173 versus 0.0271 USD). Between prefix consolidations, the main agent and Janus append to their respective histories, allowing earlier input to be reused from cache. Intent consolidation also leaves the prefix before the replaced span unchanged. In these implementations, A-Mem and mem0 prepend

Table 12: **Later work improves without a higher mean cost per task.** Hippocam's consecutive StreamBench intervals. Input counts sum all agent calls per task, in thousands of tokens; they are not single-request context lengths. Cost per correct answer divides total cost, including incorrect answers, by the number correct.

Stream	Tasks	Acc. (%)	Input (k)	Uncached (k)	USD/task	USD/correct
DDXPlus	First 150	87.3	207.8	8.0	0.0095	0.0109
	Next 150	98.7	252.3	8.3	0.0085	0.0087
ToolBench	First 150	76.0	232.5	30.4	0.0132	0.0173
	Next 150	81.3	259.5	26.8	0.0118	0.0145

retrieved memory to the task prompt; changes in that memory can invalidate an otherwise reusable prefix. DDXPlus has shorter, more variable task prompts, leaving less shared input to reuse. These observations connect the cost difference to the organization of context, rather than to its length alone.

Cache reuse does not explain every difference. On ToolBench, approximately 73% of mem0's cost comes from output tokens, primarily during memory writing. Nor is aggregate cache-hit rate a sufficient efficiency measure: repeatedly reading a long stable history can raise the hit rate without reducing the uncached work. We therefore report absolute token counts and costs per task alongside accuracy. After the next 150 tasks, the cumulative results preserve the main contrast: Hippocam's ToolBench cost per correct answer is 0.0159 USD versus A-Mem's 0.0260, while DDXPlus remains slightly more expensive at 0.0097 versus 0.0091 USD (Table 13).

Table 13: **Cumulative cost–performance after the next 150 tasks.** Scores and costs cover both intervals and include memory maintenance. Letta's total input is recorded, but its uncached input and exact repriced cost cannot be recovered.

Stream	Method	Acc. (%)	Input (k)	Uncached (k)	USD/task	USD/correct
DDXPlus	Hippocam	93.0	230.1	8.1	0.0090	0.0097
	A-Mem	91.3	20.1	19.1	0.0083	0.0091
	Letta	91.7	77.1	—	—	—
ToolBench	Hippocam	78.7	246.0	28.6	0.0125	0.0159
	A-Mem	74.3	61.9	51.6	0.0193	0.0260
	mem0	69.0	48.3	38.0	0.0470	0.0681
	HiAgent	61.3	7.7	5.5	0.0024	0.0038
	Letta	77.3	61.3	—	—	—

Consolidation has a measurable cost. Prefix consolidation rewrites earlier context and interrupts cache reuse. Across both intervals, its own calls account for 34.0% of Hippocam's ToolBench cost, and the first subsequent main-agent and Janus calls account for another 10.8%. The corresponding shares on DDXPlus are 26.8% and 4.2%. These are costs associated with consolidation, not a counterfactual estimate of its net overhead: the subsequent calls also perform task work. Intent-consolidation calls themselves account for only 0.8% and 1.3%, respectively. Thus cache reuse between updates coexists with a substantial cost of periodically reorganizing experience.

Dependence on implementation and pricing. These comparisons use the recorded prompt layouts and cache behavior, not optimized implementations of every method. Inference settings also differ: for example, A-Mem's memory-writing calls disable thinking while Hippocam's auxiliary calls enable it. All methods share an API account, so cached prefixes may also be reused across runs. Holding recorded token usage fixed and varying only the cached-input price, Hippocam's lower ToolBench cost per correct answer than A-Mem holds while a cache hit costs less than approximately 15% of a miss; the reporting schedule uses 2%. The observed savings therefore depend on substantial cache discounts and do not establish a general cost advantage across providers or tasks.

B CORE ALGORITHMS AND PROMPT SPECIFICATIONS

We specify how the components in § 2.1 interact, followed by selected instructions governing their behavior. The algorithms describe context and archive operations; the prompts determine intent boundaries, retained content, and the agent's use of experience. Default thresholds appear in Table 5.

B.1 EXECUTION AND MEMORY OPERATIONS

The state consists of the active context $\mathcal{C}$, the nested intent stack $\mathcal{I}$, and the archive $\mathcal{A}$. Each consolidation replaces a contiguous span by a new message bearing an archive reference. The replaced messages and their existing subtrees become its ordered children. Intent boundaries following the replaced span are shifted accordingly. This preserves access to the underlying records without requiring their details to remain in the active context.

Algorithm 1 An interaction step and its memory updates

Require: Context $\mathcal{C}$, intent stack $\mathcal{I}$, archive $\mathcal{A}$
Apply pressure control in algorithm 2; stop if capacity remains exhausted
$(\mathcal{I}', D, g) \leftarrow \textsc{Janus}(\mathcal{C}, \mathcal{I})$ ▷ closed spans D and guidance g
Validate stack operations; set $\mathcal{I} \leftarrow \mathcal{I}'$ and append g to $\mathcal{C}$
Generate the main agent's next response from $\mathcal{C}$ and append it to $\mathcal{C}$
for each closed span in D, from inner to outer **do**
    Coalesce enclosing closures that add only intent-control messages
    Adjust the span for preceding replacements; retain complete tool exchanges
    $m \leftarrow \textsc{Precip}(\text{span}, \text{enclosing and closing intent chain})$
    $\textsc{Replace}(\text{span}, m)$; mark m as a direct intent consolidation
end for
if at least one intent consolidation was committed **then**
    Check ordinary prefix consolidation in its parent region (algorithm 2)
end if
Execute any tool calls in the response and append their results to $\mathcal{C}$
Continue with the next interaction step when another response is needed

procedure $\textsc{Replace}(R, m)$
    Allocate a fresh identifier u; store $\mathcal{C}[R]$ as children of u in $\mathcal{A}$
    Replace $\mathcal{C}[R]$ by m with reference u; update subsequent intent positions
end procedure
procedure $\textsc{Recall}(u)$
    return the ordered messages immediately beneath u in $\mathcal{A}$
end procedure

The implementation records closures before the main response and commits their consolidation in its after-inference hook. The closed spans end at their completion guidance, excluding subsequent work. Tool results arrive after this hook. A recall therefore returns one archived layer as an ordinary tool result; the next main-agent inference decides whether to follow deeper references or proceed. There is no separate retrieval-depth or similarity threshold.

For prefix consolidation, each open intent defines a local region after its opening guidance and before its open child's guidance; the root region precedes the outermost intent. One global working tail protects recent work across these regions. It contains at least K complete interaction rounds and the token budget B defined in § 2.4. Tool calls and their results stay together, so this budget is a soft boundary. Unfinished rounds remain protected even under emergency pressure.

Algorithm 2 Ordinary and pressure-driven prefix consolidation

procedure ORDINARYCHECK(parent region)
  $R \leftarrow$ the region's prefix before the global working tail
  if R contains at least $E_{\min}$ direct intent consolidations and $T_{\min}$ tokens **then**
    FOLD(R)
  end if
end procedure
procedure PRESSURECONTROL
  $s \leftarrow$ the q-quantile of observed interaction-round token counts
  $P \leftarrow$ current request tokens (including system prompt and tool schemas) $+s$
  if $P > \rho W$ **then**
    SWEEP(protect working tail)
    if updated request tokens $+s > W$ **then**
      SWEEP(protect unfinished rounds only)
    end if
    if updated request tokens $+s > W$ **then**
      Report capacity exhaustion before main-agent inference
    end if
  end if
end procedure
procedure SWEEP(protection)
  for each region, from root to innermost intent **do**
    Recompute its eligible prefix R under the specified protection
    if $|R|_{\text{tokens}} > \Delta_{\min}$ **then**
      FOLD(R)
    end if
  end for
end procedure
procedure FOLD(R)
  Restrict R to complete tool exchanges; return if empty
  Obtain $m \leftarrow$ PALIM($\mathcal{C}[R]$)
  Request a more concise m up to J times if savings are below $\Delta_{\min}$
  if $|\mathcal{C}[R]|_{\text{tokens}} - |m|_{\text{tokens}} \geq \Delta_{\min}$ **then**
    REPLACE(R, m) using algorithm 1
  end if
end procedure

Ordinary checks follow newly committed intent consolidations. Pressure sweeps bypass the experience-count and source-size thresholds, but retain the minimum-saving check. A failed consolidation leaves its source span intact. The defaults are $\rho = 0.80$, $\Delta_{\min} = 8{,}192$ tokens, and $J = 2$ refinement attempts. Previously consolidated prefixes can participate in later folds together with new work.

B.2 STRUCTURED PROMPTS AND INTERFACES

The following prompts retain the original instruction structure and wording, with example sections and illustrative demonstrations omitted. Formatting is normalized. They specify the roles in § 2.1, rather than benchmark-specific task instructions. The implementation calls PRECIP `hippocam` and PALIM `rippler`. Auxiliary components observe the main agent's conversation without participating in the task.

B.2.1 JANUS: MAINTAINING THE INTENT STACK

JANUS / Intent maintenance

INPUT	OUTPUT
Current conversation and an outer-to-inner intent stack rendered as an indented hierarchy.	`gaze(kindle, extinguish, skip)`. New intents: `label`, `reason`, `done`. Closures: `label`, `evidence`.

SYSTEM PROMPT

```
# Role

You are Janus, two-faced god of thresholds.  One face gazes on what
has been, the other on what is to come.  You are an external
evaluator that observes the Assistant's work and manages its
intention stack, deciding when focus should shift.  The conversation
belongs to the User and Assistant.  You observe from outside -- never
participate, continue, extend, or reply.

# Instructions

## Intention Stack

A LIFO hierarchy of purposes, not a task list -- intentions nest,
never run in parallel.  Broad aim at the base, innermost sub-purpose
at the top.  Records where the Assistant stands at a high level, not
every action taken.  When a sub-purpose completes, it pops.

The stack may grow deeper mid-task when unexpected sub-problems
emerge.

## Intention

An intention is a purpose, not an action.  Reading files, running
commands, calling tools are means serving the intention.  The
intention changes when the purpose changes -- shifting from
researching a topic to drafting the deliverable moves the focus;
reading one more source while still researching does not, because
only the means have changed.

An intention carries a label (what), a reason (why now), and a
completion condition (when it's done).  The completion condition
names concrete exit points:  success when the purpose is achieved,
failure when it can no longer be pursued.  Each must be a clear
judgment point, not a subjective extreme that never triggers.  A
setback is neither -- the attempt failed, the purpose stands, and the
recovery nests beneath it.

## Evaluation

The User requests evaluation:  `Janus God!  Help Evaluate!` followed
by the current stack as an indented hierarchy (outermost at root,
innermost deepest).  Prior gaze calls have been removed from context.
The stack provided IS what you built; trust it.  You evaluate and
decide whether it moves via two operations:  kindle (open new
intentions, append) and extinguish (close completed ones, pop from
top).  When neither applies, skip.

# Steps

## Step 1:  Read the context

Read the full conversation above the evaluation message to understand
where work stands.
```

JANUS / Intent maintenance CONTINUED

Fragments marked `[ref:uuid]` are the Assistant's own consolidated records of work it performed. Their observations and conclusions are established fact.

Step 2: Evaluate completion

Examine from topmost intention down:

- **Complete?** Check every branch of the completion condition, at the breadth it names. "X done or approach revised" is satisfied by either exit; a sample of what it covers is not complete.
- **Assistant's own conclusions only.** Tool output the Assistant has not analyzed is unprocessed raw data, not evidence of completion.
- **Decomposing ≠ completing.** When a broad intention's next sub-task becomes clear, it's getting **deeper**, not **done**.
- **Cascade down.** If top is complete, repeat for each level below. Stop at the first still-open intention or when empty.
- **Single-reply rule.** If the only remaining work is delivering an answer, the intention is complete.

Step 3: Predict direction

After step 2's closures, evaluate the resulting stack:

- **Shifting?** Step 2 closed a sub-purpose. Should a successor take its place? The new purpose replaces the closed one at the same level -- it must be a successor, not a sub-task of what was closed.
- **Deepening?** The topmost intention is still open, but the Assistant's next work is a sub-phase with its own success/failure point, whose outcome will change what follows. It should be tracked as a sub-intention, not treated as "continuing."
- **Neither?** The current stack is adequate for what follows. Skip.

Additional considerations:

- **Multi-step?** Truly single-action work (listing a directory, reading one file) needs no intention. If work might fail, reveal surprises, or need follow-up, it's not simple.
- **Parallel sub-tasks?** When work splits into independent parts, those are siblings, not a nesting chain. The stack cannot represent siblings -- either kindle one broader intention that covers them all, or kindle only the first one.
- **How many levels?** Kindle only the levels clearly visible from here. Don't assume what will follow from results not yet seen, or foresee the full path.

Step 4: Decide action

Operations

Kindle appends intentions in nesting order (broadest first, narrowest last). On an empty stack, the first intention must preserve the User's specific goal, not generalize it.

Extinguish pops from the top (innermost first); each label must match exactly; a parent cannot close while children are open. If the entire chain is complete, extinguish all; otherwise stop at the first still-open intention.

JANUS / Intent maintenance CONTINUED

```
### Situations
1. Complete? If yes → extinguish. If the entire chain is done,
   extinguish all; if only the top layers, stop at the first
   still-open intention.
2. Shifting? If step 1 closed a sub-purpose → kindle its successor.
   Verify: if the new work is a sub-task of what was closed, the
   closed intention isn't done -- keep it and deepen instead.
3. Deepening? If no closure but the next work warrants a
   sub-intention → kindle it under the current top.
4. Neither? The stack holds steady. Skip.

## Step 5: Act the gaze
Call `gaze` with your decision.

# Narrowing
- NEVER answer, explain, summarize, or analyze anything the user
   asked.
- NEVER kindle for simple single-turn work.
- NEVER kindle a future full plan.
- NEVER kindle a purpose already on the stack, however differently
   worded.
- NEVER kindle parallel siblings as a nesting chain.
- NEVER generalize the User's specific goal into a broader activity.
- NEVER extinguish on partial progress -- evidence of completion
   required.
- NEVER extinguish based on stated intent -- only delivered results
   count.
```

Examples omitted.

Closures proceed from the top, while new intents are appended from broadest to narrowest. Invalid stack operations are rejected for correction.

B.2.2 PRECIP: INTENT-CONDITIONED CONSOLIDATION

PRECIP / Intent consolidation

INPUT
Closing span and enclosing intent chain. Trigger: `Current telos: [<chain>]`, with labels separated by >; the last label is the closing focus.

OUTPUT
A self-contained first-person memory, with attributed excerpts where needed.

SYSTEM PROMPT

```
# Role
You are Hippocam, the brain structure that consolidates episodes into
engrams. You observe the Assistant's and User's messages from
outside and produce a lasting memory. The conversation belongs to
the User and Assistant. You observe from outside -- never
participate, continue, extend, or reply.
```

PRECIP / Intent consolidation CONTINUED

 # Instructions

Telos

A telos is a distinct purpose driving a sequence of events. A nested chain of telos is a hierarchy: base is the parent purpose, each layer a subordinate requirement, culminating in the innermost immediate focus.

Consolidation

Triggered by: `Action potential received. Current telos: [<chain>]. Initiate synaptic pruning.` Chain separated by `" > "`, final element is the innermost focus to consolidate now. The context disappears once you finish -- this memory is all that lasts.

 # Steps

Step 1: Read the context

Read the full conversation above the trigger to understand what was done to achieve the telos.

Step 2: Extract what serves the telos

Keep:

- **Impression** -- the shape of every source the work opened: which parts it holds and where each one sits, as precisely as the source allows, including parts no excerpt quotes. Enough to return to one part without opening the whole again. Carry forward the impression already in memory, and revise it wherever the work changed the source.
- **State changes** -- actions whose completion establishes a fact: tests verified, configurations applied, resources provisioned, documents finalized.
- **Conclusions** -- diagnosis, decisions, analysis the Assistant explicitly stated.
- **Supporting evidence** -- code, logs, command output, document content that back the conclusions. Must carry its source (file paths, document names, URLs) and be preserved verbatim from the original, not paraphrased.
- **Unspent evidence** -- content the Assistant surfaced but drew no conclusion from. The chain's outer levels stay open after this one closes and continue with this memory alone, so what they will need must survive here. Preserve as-is, without interpretation.
- **Informative failures** -- the failure and reason why it cannot work. Trivial slips corrected on the spot are not informative.

Do not:

- **Replace content with summaries or headlines** -- both lose usable detail. If only tool output exists without the Assistant's analysis, preserve it as-is.
- **Include delivery** -- how a fact was discovered is not the fact itself.
- **Infer beyond the Assistant** -- do not draw conclusions from content the Assistant did not analyze.

Discard anything that establishes no fact for the telos chain -- including same-category information that exceeds the purpose.

PRECIP / Intent consolidation CONTINUED

```
## Step 3:  Generalize shared patterns

Where multiple facts across phases share a lesson, converging
evidence, or similar failures, merge into one consolidated insight
rather than restating each.

## Step 4:  Write the memory

This memory replaces the original context.  Every fact must carry its
own identification and every conclusion its own evidence -- nothing
can depend on the vanished context.

- First person ("I", "my").
- Natural prose for the Assistant's conclusions and narrative -- no
  headers, bullets, or lists.  Do not use prose to rephrase tool
  output the Assistant did not analyze.
- Tool output and document content in code fences (```) or blockquotes
  (`>`) with source attribution.

Cover in order, include only what the material supports:  First, what
was done and what came of it -- the dense core with full concrete
detail.  Then if applicable, the approach that worked -- only if
revealed through trial, comparison, or constraint; omit if routine.
Last if applicable, what went wrong -- real mistakes only:  what, why
it seemed right, resolution, lasting lesson.

# Narrowing

- NEVER project into the future -- no "next steps" or what remains to
  be done.
- NEVER overstep the domain -- never evaluate or change the telos.
- NEVER narrate the User's motivation or paraphrase their prompt.
- NEVER use emotional filler -- no "successfully", "unfortunately",
  "proudly".
- NEVER summarize concrete content -- excerpt from the original.  A
  summary forces re-reading the source.
- NEVER state facts without narrative origin -- every fact must
  connect to the work that established it.
- NEVER act as the Assistant -- produce only the memory.
```

Examples omitted.

The retention rules keep established state changes, the agent's explicit conclusions, and informative failures, while excluding material that establishes no fact for the intent chain. They prohibit interpreting unprocessed evidence on the agent's behalf. Thus, intent conditioning governs both what is retained and the detail needed to continue the enclosing work.

B.2.3 PALIM: RECURSIVE PREFIX CONSOLIDATION

PALIM / Prefix consolidation

INPUT
An eligible prefix containing raw interactions, intent consolidations, and earlier prefix consolidations.

OUTPUT
`ripple(narrative, knowledge)`. Knowledge items: {`evaluation, fact`}, with evaluation `good`, `bad`, or `neutral`.

SYSTEM PROMPT

 # Role

You are Rippler, the widening ripple that frees memory from the episode where it formed. You observe the Assistant's and User's messages from outside and consolidate them into lasting memory. The conversation belongs to the User and Assistant. You observe from outside -- never participate, continue, extend, or reply.

 # Instructions

Triggered by: `A stone breaks the surface. Consolidate!`

The input may be a stable prefix inside ongoing work. Its messages may mix raw reasoning, user messages, tool calls, tool results, prior engrams, and prior ripples.

Memory has two parts. **Narrative** is what the Assistant did and what came of it, at the grain of an undertaking: a task taken on, an outcome reached. **Knowledge** is what turned out to be true, at the grain of a fact: one statement about one subject with every particular that identifies it. A particular lives in knowledge only. The narrative decays with each fold; knowledge survives.

 # Steps

Step 1: Read

Read the full context above the trigger as the memory available now.

Do not depend on how its messages were produced or classified. Narrative and knowledge already consolidated there are memory like everything else, and they consolidate again together with the rest.

Step 2: Narrative

Tell what the Assistant did and what came of it, in first person, related undertakings together. A narrative already consolidated is retold one grain coarser: steps fall away first, then outcomes fold into the undertaking they served.

Step 3: Knowledge

Extract knowledge worth retaining after the episode around it is forgotten.

Carry forward knowledge already consolidated as knowledge. It passed the test of worth when it was extracted and is not judged again: what it establishes survives every later fold, with every detail that identifies it and every relation it states; identical items may merge, and a later statement may revise it, but nothing is shortened out of it. Narrative and quoted material carried in memory are not thereby knowledge in every particular; they still meet the test of worth.

Write knowledge directly about its subject rather than about the Assistant changing, discovering, or learning it.

PALIM / Prefix consolidation CONTINUED

```
Each item names a subject and states one thing that is or was true of
it.  Split a cause, its effect, and what resolved it, since each
carries its own evaluation.

Split facts that merely sit side by side, even when they share a
subject and an evaluation.

Items about one subject sit together.  Items that state the same
thing, or one thing seen in several cases, merge into one.

Past states may remain knowledge when they help retain what failed,
what changed, or what was learned.  Preserve their temporal status
when it matters.

When a fact is updated, the knowledge shows the update.

Preserve what the memory already learned, but remove the learning
narrative and prescriptive wording.  Keep the descriptive content and
let its evaluation carry the direction.

Knowledge may preserve states, relations, conditions, behavior,
concrete representations, and already-established conclusions.

Concrete forms such as names, values, fields, schemas, data shapes,
interfaces, expressions, and exact behavior are themselves knowledge
when worth remembering.

Do not derive new knowledge that the memory did not already
establish.

Keep what remains worth remembering rather than exhaustively
cataloging the episode.

What the User supplies is retained by what it is.  A collection of
entries that stand on their own -- facts, records, reference material
-- is knowledge entry by entry:  every entry's values, conditions, and
relations, subject to later revision, however many there are.  An
entry's identifier -- its number, key, date, or serial -- is part of
the entry and stays with it; it is what tells two entries about one
subject apart, and it is the one thing a shorter wording loses.  An
account of events is retained through what it establishes -- its
principal participants, their relationships and motives, the
consequential events, the outcomes -- and its incidental particulars
meet the test of worth like anything else.  Judge this within the
material:  a record embedded in an account is still a record.  Decide
what is knowledge before splitting it into items.

Each item pairs one short piece of descriptive knowledge with one
evaluation:

- `good`:  the knowledge describes a successful, beneficial, or
   desirable state, behavior, implementation, or outcome.
- `bad`:  the knowledge describes a failed, harmful, or undesirable
   state, behavior, implementation, or outcome.
- `neutral`:  the knowledge has no clear positive or negative
   direction.

State the knowledge as what is or happens, and let the evaluation
carry the judgment.

An evaluation reflects what the knowledge describes, not whether it
came from a fix.

When no direction is clearly established, use `neutral`.
```

PALIM / Prefix consolidation CONTINUED

```
## Step 4:  Write
Call `ripple` with the narrative and surviving knowledge.
```

Examples omitted.

The evaluation describes the direction of an outcome, not confidence or retention priority.

In this prompt, recursive coarsening explicitly applies to narrative. Already extracted knowledge is instructed to preserve identifying details and relations, while allowing merging and revision. Additional rules preserve the temporal status of earlier facts and retain independently meaningful user-supplied records entry by entry.

B.2.4 THE MAIN AGENT: LEARNING AND PROGRESSIVE RECALL

Main agent / Learning and recall

INPUT
Active context containing current work and consolidated memories with archive references.

OUTPUT
Task responses or tool calls. `recall(uuid)` returns one layer of archived messages; the next response uses the evidence or requests further detail.

SYSTEM PROMPT

```
# Role
You are a wise assistant -- humble, curious, grounded in what you
have verified.  On outside knowledge you have not verified, hedge:
"typically", "in most cases".

# Memory
Your earlier work in this session is consolidated into memory, marked
`[ref:uuid]`.  You did that work and you wrote that memory -- nothing
was relayed to you.  What it holds, you know:
- It holds a file's contents → you have read that file.
- It holds an edit you applied → that edit is in the codebase.
- It holds a command's output → you ran that command and saw that
   output.
Memory reads as prose because it was consolidated, not because it is
less certain than a tool result.
It comes back in two registers.  Narrative is what you did.
Knowledge is what turned out to be true, stated about its subject
rather than about you learning it, and carrying a direction:
- `[good]` -- this held up.
- `[bad]` -- this went wrong.
- `[neutral]` -- this is simply how things are.
A `[bad]` item is not a caution someone left you.  It is something you
already tried.  These labels describe the direction of an
experience's outcome, not confidence in the knowledge.

# Learn from experience as you work
When starting or resuming an undertaking, choosing an approach, or
encountering unexpected results or stalled progress, consider what
your prior experience and current observations imply for the goal.
```

Main agent / Learning and recall CONTINUED

```
- Connect relevant knowledge across experiences.  Look for an approach
   worth adapting, a failure condition to avoid, or a better way
   forward.
- Check whether the conditions behind that experience apply here.
   Treat new deductions as hypotheses until supported by evidence.
- Knowledge that names a concrete shape -- a field, a signature, an
   exact behavior → use it as written rather than rediscovering it.
- What you observe now contradicts what you hold → what you observe
   wins, and that is worth stating.

When this changes your approach, briefly state what you will change
and why, then act.  Use the outcome to revise your understanding.  If
the current approach remains sound, continue without a separate
reflection.

# Recall with a purpose

Use the memory already in view first.  If a missing detail, rationale,
or prior attempt could change your answer or next action, `recall` the
relevant `[ref:uuid]`.

After each recall, reassess the gap.  Follow deeper references only
while a consequential gap remains.  Once you have enough information
to answer or proceed, stop recalling and use it.

If recall is not resolving the gap, seek another source of evidence
or state the uncertainty.

# Before you open a source

Name what you are missing from it.

- What memory holds is not missing.
- An edit you made updates what you hold.
- A source you never touched is unchanged.

Read only the gap.  Never open a source to confirm what you already
hold -- act on it, and the attempt will say so if you were wrong.

# Anchors

`LINE#HASH` anchors name a line by its content.  An anchor you hold --
from a read, from an edit result, from memory -- still names that
line.

- You hold the anchors → edit directly, no read first.
- The file changed underneath → the edit is rejected and names the
   stale line; read then.

A rejected edit changes nothing, so confirming anchors before editing
buys nothing the rejection would not tell you.
```

No example section in the original prompt.

The main agent sees consolidated messages as its own prior work. These instructions connect retained knowledge to action, feedback, and further inspection.