CASEMEM: CLOSING THE LOOP FROM AGENT OUTCOMES TO MEMORY REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory lets an agent retain knowledge from a growing interaction history. Most memory systems focus on what to store and how to retrieve it. However, a relevant memory is not necessarily one that should be applied to the current decision, nor one that actually helped when it was used. We introduce **CaseMem**, which separates reusable knowledge from task-specific application and evidence from later use. Each CaseMem record assigns a stable identity to a piece of knowledge, along with its retrieval address, conditions of use, and supporting sources. Before the agent acts, a grounding step binds reusable procedures to the observable task state. When feedback is enabled, a lightweight ledger records exposure and outcome against supplied records to guide later selection without rewriting memory or retraining the agent. On static memory benchmarks, CaseMem is competitive with recent systems without generative construction calls. On agentic benchmark MemoryArena, retrieved experience and task grounding make complementary contributions under two different model family. Their combination improves string reward over retrieval alone by 24% with Qwen3-32B and 12% with GPT-4.1-mini. Separate controlled experiments on a fixed store show that outcome feedback can recover an overlooked procedure and demote a harmful one; replacing current feedback with earlier outcomes reduces the gains. Together, these findings show that agent memory can improve through how knowledge is applied and selected, alongside how it is stored.

1 INTRODUCTION

LLM agents are now used for long tasks that run over multiple steps and sessions. Memory lets an agent carry what it learned earlier into making decisions in a much later stage. Recent work on agent memory focuses on building a better store, including extracting facts, summarizing histories, linking related events, or writing down reusable lessons so that the useful information can be found later (Chhikara et al., 2025; Xu et al., 2025; Fang et al., 2026; Xia et al., 2026; Gutiérrez et al., 2025). Many of these systems make multiple model calls while building the store for better retrieval afterwards. However, building a richer store is only one part of making memory more useful. The same stored knowledge can be more useful when it is applied carefully to a later decision. The outcome of that decision then provides evidence about whether the memory should be reused in future tasks. This leads to our central question: *How can agents turn stored experience into better decisions, and use the outcomes to guide future memory reuse?*

We investigate whether this connection requires elaborate memory construction, or whether inexpensive records can support both task-specific application and outcome-driven selection. Outcome-driven methods already show that feedback can improve memory use, by distilling new strategies, training memory managers, or changing retrieval preferences from task reward (Ouyang et al., 2026; Yan et al., 2026b; Zhang et al., 2026). These approaches target different parts of memory use, from the knowledge stored to the policy that selects it. When an agent performs better, did it acquire better knowledge, apply the same knowledge more effectively, or learn to select a different memory? We treat these as distinct responsibilities. Stored knowledge determines what can be reused, task grounding determines how it applies to the decision at hand, and evidence from later outcomes changes future reuse without requiring the knowledge itself to change.

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.

CaseMem is built around this separation. A CASE record keeps reusable content, a retrieval address, a scope, and source evidence under a stable identity; for static histories, records are constructed by segmentation and indexing without generative extraction. Reusable procedures carry typed slots for their task-specific values, and a deterministic binder fills the slots from observable pre-action state, withholding the procedure when the binding is ambiguous; we call this grounded configuration CaseMem+. When feedback is enabled, a reuse ledger attached to each supplied record accumulates exposure and outcome evidence, and later retrieval can rank by that evidence while the content stays fixed. Stable identities keep stored content and evidence from later use linked. Grounding determines how a procedure applies now; the ledger changes which records are selected later. Neither operation retrains the agent or rewrites stored knowledge.

We evaluate the three responsibilities separately. On LoCoMo, LongMemEval-S, and AMA-Bench, CaseMem reaches competitive static-memory accuracy with stores built without generative construction calls, and stays above a lexical–dense fusion baseline over the same chunks (§4.2). In agentic evaluation across two reader models, retrieved experience and explicit task grounding make complementary contributions: their combination outperforms retrieval alone and the grounded procedure alone, and a generic reminder of the same instructions does not recover the benefit of grounding, with or without retrieved records present (§4.3). How retrieved knowledge is applied therefore matters even when the knowledge is unchanged. Finally, with the store fixed and the initial retrieval error known, correctly associated outcomes surface a useful procedure that the prior misses or demote a harmful one that it repeatedly selects, and reassociating outcomes across rounds removes the advantage (§4.4). The same update does not reliably improve an already-correct prior, natural stores, or held-out categories, and the credit signal does not identify which of several co-supplied memories the agent followed; these controls separate outcome-associated reuse from changes in memory content and mark its tested limits.

Together, the experiments support a practical view of agent memory: better performance need not always come from constructing richer memories. It can also come from changing how existing knowledge is applied and reused, while keeping that knowledge stable. We use these insights to make the following contributions:

1. **Economical construction of reusable memory.** We introduce CASE, a lightweight record that preserves content, retrieval information, scope, and source evidence. Segmentation and indexing construct static stores without generative calls, while achieving competitive accuracy across three memory benchmarks.
2. **Task grounding and outcome-driven reuse over persistent knowledge.** CaseMem makes procedure application explicit and attaches a reuse ledger to supplied records. Grounding resolves task-specific variables, while outcome evidence adjusts future selection without rewriting stored content or retraining the agent.
3. **Evidence that application and feedback improve memory use.** Agentic evaluation across two reader models shows that retrieval and grounding together outperform either alone, including against a generic-reminder control. Separate fixed-store experiments show that correctly associated outcomes can correct a known retrieval error.

2 RELATED WORK

Agent-memory methods differ in how they organize experience, how they apply it to a new task, and how they update future reuse from feedback; Figure 1 places prior work along these stages. CaseMem carries three elements through all of them: the stored record, the task-bound instance, and the reuse ledger. The first two are the structure; the third is the loop that runs on it.

Constructing and retrieving memory. Mem0 extracts facts and updates them, A-MEM links notes, and LightMem compresses histories (Chhikara et al., 2025; Xu et al., 2025; Fang et al., 2026). Memora indexes concrete values through abstractions and cue anchors, HippoRAG 2 retrieves over a graph, PlugMem provides task-agnostic storage, and AgentRunbook-C stores trajectories as files and gathers evidence with a coding agent at query time (Xia et al., 2026; Gutiérrez et al., 2025; Yang et al., 2026a; Wu et al., 2026). These methods improve how experience is represented and retrieved, often through generative extraction or consolidation. Our static construction uses segmentation and

indexing alone, and each item keeps a scope field and a pointer to its source that the later stages read rather than rewrite.

Applying reusable experience. When memory holds experience rather than facts, the agent must fit it to the task in front of it. Synapse supplies past trajectories as exemplars, Memento selects prior episodes as cases, AWM induces workflows with abstracted arguments, and ReasoningBank distills strategies from successes and failures (Zheng et al., 2024; Zhou et al., 2025; Wang et al., 2025b; Ouyang et al., 2026). We make task-specific binding an explicit operation, as structured retrieval does for latent variables across reasoning steps (Sun et al., 2026): procedures carry typed slots that a deterministic binder resolves from observable state before supplying a decision card, or withholds when the state does not fix a binding. Because the card is computed rather than stored, it can be switched off, which lets our component comparisons separate what grounding contributes from what retrieved experience and a generic reminder contribute (§4.3).

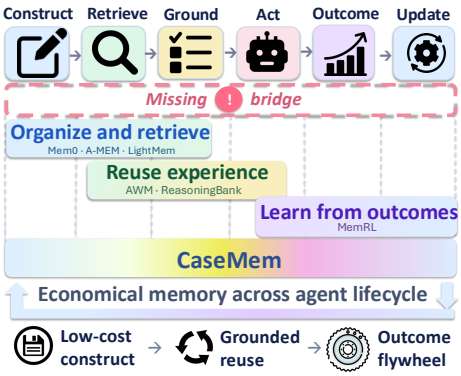


Figure 1: Agent memory landscape. CaseMem preserves reusable knowledge, grounds applicable procedures in the current task, and records subsequent outcomes to guide future selection.

Learning from outcomes. Feedback can change model parameters, memory text, or the preference used at retrieval. Memory-R1, Mem-α, Fine-Mem, and Memory-R2 train memory operations with reinforcement learning (Yan et al., 2026b; Wang et al., 2025a; Ma et al., 2026; Yan et al., 2026a); related work studies credit propagation over provenance graphs, graph-structured experience, outcome-trained retrieval, and reduced-order task-specific utility states (Liao et al., 2026; Feng et al., 2026; Tan et al., 2026; Yang et al., 2026b). MemRL is closest to our setting: it keeps content fixed and learns one utility per item, combined with similarity at retrieval (Zhang et al., 2026). Our ledger records exposure and credited outcome as separate per-family counts and does not seed them from the source episode's reward, so a procedure admitted from a failed episode is scored only by its later use. Feedback observed only after selection connects this setting to contextual bandits and learning from logged feedback (Beygelzimer et al., 2011; Swaminathan & Joachims, 2015), and a shared outcome for several supplied memories raises a credit-assignment problem related to slate selection and evaluation (Kale et al., 2010; Swaminathan et al., 2017); we test the link between exposure and outcome and treat credit within a set as approximate.

CaseMem connects these stages while keeping stored knowledge, task-specific bindings, and reuse statistics separate. The separation is what lets an outcome be written back to the memory that was actually supplied, and what lets us evaluate grounding and outcome-driven selection independently, without reconstructing the stored knowledge.

3 CaseMem: Stable Knowledge, Adaptive Reuse

CaseMem connects reusable knowledge with evidence from its use. A Case record keeps content, scope, and source evidence under a stable identity; grounding turns a reusable procedure into instructions for the task at hand; and once the agent has acted, the reuse ledger records the outcome against the records that were supplied, which shifts what is supplied next time. Because the identity stays fixed, the system can change how it uses knowledge without rebuilding the knowledge or retraining the agent, and neither grounding nor the ledger update adds a generative-model call. Figure 2 shows the two parts: the reusable unit (§3.1) is the record plus the binder that instantiates it, and the reuse flywheel (§3.2) is the loop that reads the unit, supplies it, and updates its ledger. CaseMem stores and retrieves Case records; CaseMem+ also grounds procedures. Ledger updates are a separate switch: the shopping comparisons test grounding with the ledger off, and the fixed-store comparisons test outcome-driven selection with grounding off.

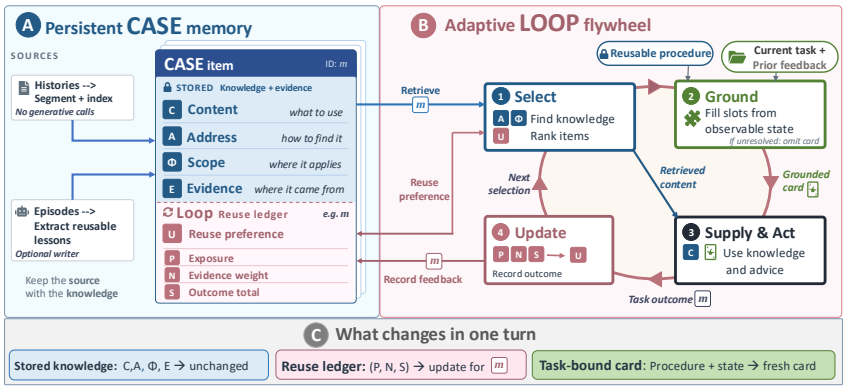


Figure 2: The CaseMem framework. Left: a CASE record keeps content, address, scope, and source evidence under one identity, with a reuse ledger attached. Right: on each task the system retrieves records, grounds any procedure into a decision card from the observable state, hands the result to the agent, and writes the outcome to the ledgers of the records it supplied. Bottom: stored knowledge stays fixed, ledger counts change after use, and the task-bound card is computed afresh.

3.1 THE REUSABLE UNIT

The record. A record has four fields under one identity, $m = (C, A, \Phi, E)$. **Content** C is what the agent reads: a passage, a fact, a procedure, or an exception. **Address** A is what retrieval reads, a summary or the full text with whatever dense, lexical, entity, or temporal cues are available. **Scope** Φ says where the record may be used, such as a product category or a decision slot. **Evidence** E points back to the source passage or episode and, when there was one, its outcome. Changing how a record is indexed changes A and nothing else, so retrieval settings never alter what the agent sees. For static histories we segment the text deterministically, fill A by text processing, and fill Φ and E from source metadata, using an embedding model and no generative calls. For agent episodes, a writer may distill a procedure from a success or from a failure. The episode's outcome goes into E as provenance and does not seed the record's reuse statistics, so a procedure taken from a failed episode starts on the same footing as one taken from a success.

Grounding. Part of a procedure recurs and part of it does not. In *choose an item compatible with the previous product*, the instruction is the same every time, while the previous product and the attribute that has to match differ on every task. We store the recurring part with typed slots for the rest. Before supply, a deterministic binder reads the observable pre-action request and the feedback from completed steps, resolves the slots (one earlier product, the current requirement, the avoid-list, the stated preference), and renders a short decision card; if any slot is unresolved or ambiguous, it supplies nothing. The card is computed fresh on each task and is not stored, so the procedure text stays fixed. The binder never sees the current target, the product catalog, the reward, or the post-action trace. In shopping, CaseMem+ supplies retrieved CASE records alongside the card from one fixed compatibility procedure, which receives no ledger updates in these experiments; Section 4.3 tests this combination against its components.

Four invariants hold throughout the reuse loop: outcome updates change reuse statistics and leave content alone; procedure variables are filled from observable state before use, or the procedure is withheld; source outcomes record how a memory was acquired, while later outcomes inform how it is reused; and feedback reaches only the identities of the records that were actually supplied.

3.2 THE REUSE FLYWHEEL

A frozen LLM agent works through tasks $t = 1, 2, \ldots$ with a memory store $\mathcal{M}$. Before it acts on task t, the system sees a query q_t and a context s_t, such as the current goal and product category, and supplies at most k records, written R_t; the environment then returns a reward $r_t \in [0, 1]$. Every task has a family label g_t, in our experiments the product category, and reuse statistics are kept per family because the same procedure can help in one category and hurt in another. Each record carries a validity flag $V(m)$ and, per family, three counts $(P_{m,g}, N_{m,g}, S_{m,g})$. A correction, contradiction,

supersession, or expiry can set $V(m) = 0$ through separate operations; task reward touches the counts and nothing else. This is why the shared-store experiments hold content and identity fixed: they ask whether reuse improves, not whether revision does. Appendix B gives the step-by-step loop (Table 4), the validity operations, and the notation (Table 5).

Exposure and credit. A task returns one reward, and up to k records were in the prompt when it did. The ledger keeps apart that a record was present and how much of the outcome it is credited with. Let $y_t = \mathbf{1}[\text{exact match } \vee\ r_t \geq \theta]$ for binary feedback, or $y_t = r_t$ for continuous feedback. For each $m \in R_t$ and $g = g_t$,

$$P_{m,g} \leftarrow P_{m,g} + 1, \qquad N_{m,g} \leftarrow N_{m,g} + w_{t,m}, \qquad S_{m,g} \leftarrow S_{m,g} + w_{t,m} y_t. \tag{1}$$

P counts exposures, N accumulates credit weight, and S the weighted outcome. Keeping P separate from N means that a record shown often but credited little is still known to have been shown often, which the exploration term below relies on. The reported configuration uses behavioral weights, which score each supplied record by how closely it agrees with the recorded actions and then normalize within R_t; a matched equal-credit control sets $w_{t,m} = 1/|R_t|$ (Appendix B). Records outside R_t are not updated. Within R_t, credit is shared rather than attributed: the ledger ties an outcome to the records that were exposed, and §4.4 looks at what follows from that.

Reference-centered preference. A family reference b_g centers the reuse preference, and shrinkage with strength κ keeps a handful of outcomes from taking over selection:

$$U(m, g) = \frac{\kappa b_g + S_{m,g}}{\kappa + N_{m,g}} - b_g = \frac{N_{m,g}}{\kappa + N_{m,g}} \big(\bar{y}_{m,g} - b_g\big), \tag{2}$$

with $\bar{y}_{m,g} = S_{m,g}/N_{m,g}$ when $N_{m,g} > 0$. A record with no evidence sits at $U = 0$; a few outcomes move it a little, and many move it toward its observed advantage over the reference. In the runs we report, b_g is the smoothed pooled outcome of the records used in family g, so U ranks exposed records against each other. With b_g held fixed, the binary update is a weighted Beta–Bernoulli posterior mean minus its prior mean; Appendix G compares this reference with a separately measured no-memory baseline.

Selection. Retrieval first drops records with $V(m) = 0$ or a detectable scope conflict. A base score $B(m \mid q_t, s_t)$, built from the configured relevance channels (Appendix B), ranks what remains, and the top candidates form a shortlist $\mathcal{C}_t$, which we rerank by

$$\text{score}_t(m) = B(m \mid q_t, s_t) + \lambda_U \left[U(m, g_t) + \lambda_E \sqrt{\frac{\ln(1 + T_{g_t})}{1 + P_{m,g_t}}} \right], \qquad T_g = \sum_m P_{m,g}, \tag{3}$$

taking the top k as R_t. Relevance keeps the choice among plausible candidates, utility pulls toward records whose recorded outcomes were better, and the bonus lets a record that has rarely been shown collect evidence. The bonus decays with exposures P rather than credit N; otherwise a record shown many times with a small weight would look novel forever. Exploration stays inside the shortlist. With $\lambda_U = 0$ the rule is static retrieval, and with $\lambda_E = 0$ it is utility-only. The declared configuration uses $\lambda_U = 4$ and $\lambda_E = 0.05$, an effective exploration weight of 0.2, and one turn updates at most k ledger entries.

4 Experiments

4.1 Evaluation design

We organize the evaluation around three research questions:

RQ1. Can inexpensive construction support accurate memory retrieval?
RQ2. Do retrieved experience and task grounding contribute complementary gains?
RQ3. Can outcomes improve future selection while stored knowledge stays fixed?

Static benchmarks (LoCoMo, LongMemEval-S, AMA-Bench) use Qwen3-32B and GPT-4.1-mini readers with the ledger disabled (Table 1); on LoCoMo and LongMemEval-S the GPT reader

Table 1: Static retrieval with a Qwen3-32B reader and with GPT-4.1-mini replayed on the same saved contexts (AMA-Bench: agent re-runs); *Tok.* is the median prompt tokens per question and *build calls* the generative construction calls on LoCoMo. LoCoMo uses the mem0/Memora judge and the other benchmarks the strict judge (Qwen for the Qwen column, GPT-4o-mini for the GPT column). Best per column in **bold**, second in ***bold italic***. Baselines run on selected benchmarks are in Table 7; strict LoCoMo scores in Table 12.

	AMA-Bench			LoCoMo			LongMemEval-S			Build calls
Method	Qwen	GPT	Tok.	Qwen	GPT	Tok.	Qwen	GPT	Tok.	(LoCoMo)
BM25	0.417	0.419	7.7k	0.671	0.677	0.8k	0.504	0.450	0.8k	0
Dense retrieval	0.444	0.444	5.9k	0.666	0.681	0.8k	0.522	0.454	0.7k	0
Hybrid dense+BM25 (RRF)	0.461	0.452	6.0k	0.704	0.723	0.8k	0.538	0.478	0.7k	0
mem0	0.276	0.285	0.6k	***0.819***	***0.875***	9.4k	0.526	0.516	555k	5,882
CaseMem	**0.492**	***0.457***	5.7k	0.749	0.783	1.2k	***0.630***	***0.600***	1.8k	0
CaseMem, larger read	***0.491***	**0.459**	6.1k	**0.842**	**0.889**	5.6k	**0.638**	**0.626**	2.2k	0

receives the same saved contexts as the Qwen reader, while AMA requires agent reruns. LoCoMo uses the mem0/Memora accuracy prompt with GPT-4o-mini for both columns; AMA and LongMemEval-S use the strict prompt, with Qwen judging the Qwen column and GPT-4o-mini the GPT column. Memora-P runs its own pipeline, baselines use released code with documented endpoint and context-limit patches, and generative build calls are reported separately from reader input tokens (Appendix D). In MemoryArena shopping, each bundle contains six interdependent purchases; the principal comparison covers all 150 bundles across five categories, with fresh agent state and empty memory at each bundle start. CaseMem retrieves earlier episodes from that bundle, CaseMem+ adds grounding, card-only removes retrieval, and the generic reminder repeats the task rules without binding them to the current state. Eight methods use Qwen3-32B and five component arms are repeated with GPT-4.1-mini. String reward measures product-type and attribute satisfaction, judged reward and exact-target success are reported separately, and paired intervals resample whole bundles. Grounding experiments disable ledger updates, and feedback experiments disable grounding: each feedback arm uses the same frozen store and agent, only the ledger changes across tasks, and settings are developed on baking, grocery, and beauty, then frozen for electronics and home. A controlled store of five procedures with predefined helpful, harmful, or irrelevant roles tests a known retrieval error before we study natural records; Appendix C gives stream lengths, repetition variance, store construction, query variants, credit rules, isolation checks, and adapter details.

4.2 ***RQ1:*** ACCURATE RETRIEVAL AT LOW CONSTRUCTION COST

CaseMem reaches competitive accuracy on all three benchmarks with stores built by segmentation and indexing alone (Table 1). On AMA-Bench it scores 0.492 with Qwen at $k = 15$, against 0.444 for dense retrieval at the same depth and 0.460 for HippoRAG 2 (Table 7); with GPT the larger read scores 0.459 against 0.444 and 0.451. On LongMemEval-S the $k = 8$ configuration scores 0.630 with Qwen and 0.600 with GPT, above dense retrieval and mem0 under both readers; the larger GPT read reaches 0.626, close to LightMem's 0.632, with 2.2k median reader tokens against LightMem's 70k, which include query-time compression.

LoCoMo: higher accuracy at lower reading cost. At $k = 8$, CaseMem scores 0.749 with Qwen against 0.666 for dense retrieval over the same chunks. At $k = 32$ with mem0's answer prompt it reaches 0.842 with Qwen and 0.889 with GPT, against 0.819 and 0.875 for mem0, with 5.6k median reader tokens against mem0's 9.4k and none of its 5,882 extraction calls. Memora-P's own GPT pipeline scores 0.856 with 15.3k reader tokens (Table 7). Under the strict judge, CaseMem scores 0.685, released Memora-P 0.679, and mem0 0.613 on the 1,518 questions they share (Table 12). Read depths and answer prompts differ across rows, so these rows compare accuracy and cost rather than representation alone.

The gain is not lexical–dense fusion. BM25 and dense retrieval also need no generative calls, and CaseMem improves on both and on their fusion over the same chunks: with GPT, fusion scores

0.723 against 0.783 on LoCoMo and 0.478 against 0.600 on LongMemEval-S, with a smaller gap on AMA-Bench (0.452 against 0.459). On LoCoMo the advantage over dense retrieval holds under a second embedding model (Table 15). More retrieval channels do not help on LongMemEval-S: dense-only retrieval scores 0.618 against 0.478 for the multichannel configuration (Table 14).

4.3 *RQ2:* COMPLEMENTARY GAINS FROM RETRIEVAL AND GROUNDING

CaseMem+ combines retrieved experience with a procedure grounded in the current task. To identify each contribution, we remove retrieval or grounding while keeping the ledger off. The binder reads the corrective Purchase History available to every method and resolves the earlier product, current requirement, and preference. It never reads the current target or action outcome.

Table 2: MemoryArena shopping with a Qwen3-32B agent: string reward, judged reward, and reader input tokens (thousands) under category-persistent memory (top, historical runs) and matched per-bundle reset (bottom, the principal comparison); rows are not comparable across blocks. The historical long-context run has no token accounting; *pre-repair MemRL adapter (Appendix C); full results in Tables 22 and 16.

Method	Mean	Judged	Tokens
Category-persistent memory; grounding off			
No memory	0.174	0.253	8.0
Long context	0.200	0.268	—
BM25 over episodes	0.167	0.230	37.4
Dense over episodes	0.150	0.197	37.4
HippoRAG 2 + our writer	0.190	0.273	8.7
LightMem	0.186	0.268	8.3
mem0	0.192	0.276	10.5
A-MEM	0.168	0.225	49.5
CaseMem, ledger on	0.194	**0.281**	**7.9**
CaseMem, ledger off	0.190	0.275	8.7
Matched per-bundle reset; ledger off			
No memory	0.199	0.283	17.7
Long context	0.158	0.213	60.3
Dense over episodes	0.195	0.267	17.3
mem0	0.178	0.253	17.3
Ungrounded procedure	0.202	0.283	16.0
Grounded procedure	0.224	0.317	16.2
CaseMem	0.201	0.278	**13.9**
CaseMem+	**0.249**	**0.344**	15.6

Higher reward with modest reading overhead. Across the full shopping evaluation, CaseMem+ has the highest mean string reward of the eight matched methods: 0.249, against 0.201 for CaseMem and 0.224 for card-only (Table 16); judged reward rises from 0.278 to 0.344. Grounding adds no model call and raises mean reader input from 13.9k to 15.6k tokens per subtask, below dense retrieval and mem0 (17.3k) and long context (60.3k).

On 50 fixed bundles that add ReasoningBank, A-MEM, and LightMem, CaseMem+ again has the highest mean (Table 19): $+0.066$ $[+0.035, +0.098]$ over ReasoningBank, with margins over A-MEM and LightMem whose intervals include zero.

Both components contribute across reader models. Removing grounding lowers Qwen reward by 0.048 $[+0.027, +0.069]$; removing retrieval lowers it by 0.025 $[+0.009, +0.042]$ (Figure 3a). GPT-4.1-mini shows the same pattern: 0.355 for CaseMem+ against 0.318 for retrieval alone and 0.317 for card-only, with paired gains of $+0.036$ and $+0.038$ and both intervals above zero (Figure 3b). The two contributions are close to additive (interaction $+0.023$ $[-0.001, +0.048]$). On 30 predeclared bundles executed four times, the grounding increment keeps its sign in every execution, while the retrieval-over-card increment ranges from -0.031 to $+0.051$ (Tables 20 and 21).

Binding helps beyond restating the rule. A matched control keeps CASE retrieval and replaces the grounded card with the same procedure left unresolved, reserving the same space for it. On 30 predeclared bundles, grounding raises mean reward from 0.212 to 0.275, $+0.063$ $[+0.016, +0.111]$, with 19 bundles improved (Figure 3c). Without retrieval, card-only exceeds the generic reminder by 0.021 with Qwen and 0.034 with GPT, both intervals above zero. The benefit comes from resolving task-specific values, not from restating the rule. Under identical grounding, CASE and dense retrieval score 0.275 and 0.227, with a paired interval that includes zero (Table 21).

What improves during execution. These gains are in product-type and attribute satisfaction; exact-target success does not follow: with GPT, CaseMem+ hits 130 exact targets and retrieval alone

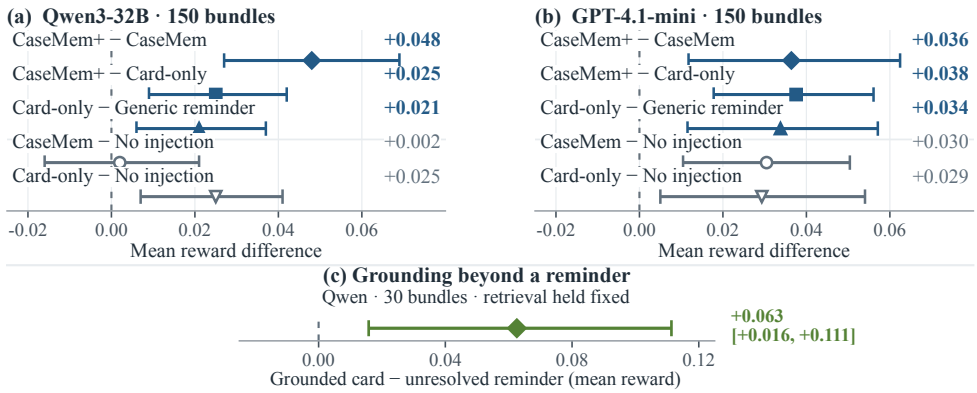


Figure 3: **Retrieval and grounding contribute complementary gains, and binding helps beyond a reminder.** (a–b) Paired differences over all 150 shopping bundles for each reader, blue between components and gray against no injected memory; (c) the 30-bundle Qwen control that keeps CASE retrieval and replaces an unresolved reminder with the grounded card. Points right of zero favor the first method, bars are 95% bundle-bootstrap intervals, and the ledger is off throughout (Tables 17 and 21).

132, out of 900 steps (Table 18). The Qwen binder supplies a card on 709 of 900 steps and withholds it when the earlier product or the required relation is unresolved. From identical saved states at home step 6, grounding adds 0.119 over ten bundles, while the step 3 estimate is inconclusive (Appendix F).

4.4 *RQ3:* Outcome feedback over a fixed memory store

We now hold the agent and stored knowledge fixed and let outcomes change future selection. The controlled store creates two errors: a useful procedure is overlooked, or a harmful one is repeatedly retrieved. Each setting has six development streams of 120 subtasks, and the shared-rule query makes the static selector keep the same pair, so any change in selection comes from the ledger (Figure 4).

Exploration finds an overlooked procedure. With lexical retrieval, the useful full-search procedure starts outside the selected pair. Utility updates alone cannot learn about a record that is never supplied. Exploration exposes it in five of six streams and raises mean reward from 0.173 to 0.209; utility-only retrieval scores 0.150.

Outcome updates reduce harmful reuse. With dense retrieval, the static pair already contains a harmful procedure. Utility updates reduce its exposure and raise mean reward from 0.160 to 0.210 (Figure 4b). In the final 12-round block, the harmful record is supplied on two-thirds of rounds, averaged across streams, versus every round under static retrieval. Utility plus exploration scores 0.196: continued exploration sometimes supplies the harmful record again. Discovery and correction need different balances of exploration and reuse.

Current feedback improves selection. We replace each eligible update's outcome with an outcome sampled from an earlier round of the same stream, keeping the update's recipients and weights unchanged. Current feedback gives higher reward by $+0.044$ for utility-only retrieval and $+0.053$ with exploration, with descriptive 95% intervals $[+0.003, +0.086]$ and $[+0.019, +0.086]$ (Figure 4c). The exploration-enabled contrast is positive in five of six streams and on average in each development category. The ledger benefits from feedback tied to the current use, not from accumulated reward alone. This control changes when feedback arrives, not which record receives it; credit within a supplied set is examined below.

Useful memories are not always ranked first. The controlled store specifies which procedures should help; natural stores can also contain useful alternatives that retrieval misses. From an identical saved task state, we supply the first-ranked record A, the second-ranked record B, both, or neither. Across 33 eligible development bundles, B outperforms A on 16 bundles, loses on four, and

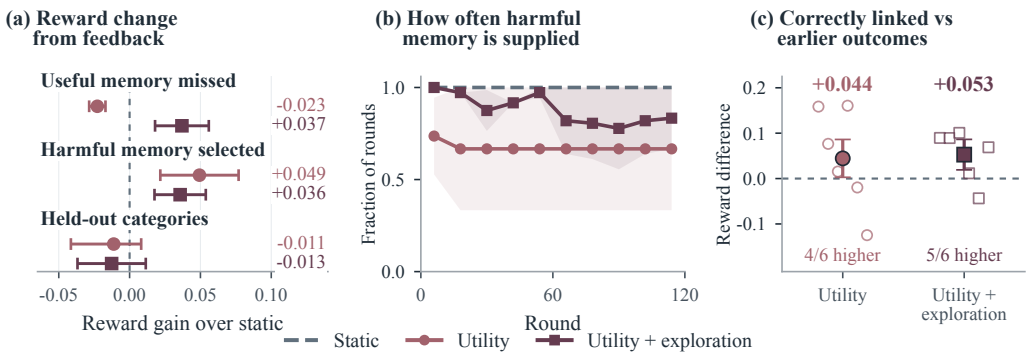


Figure 4: **Outcome feedback corrects selection errors without rewriting memory.** (a) Reward change from feedback relative to static retrieval in three settings, six runs each, including the held-out categories where it does not carry over; (b) how often the harmful memory is supplied over 120 rounds, in 12-round averages; (c) reward with correctly linked feedback minus reward when updates use an earlier round's outcome, small points per run and large points their mean. Bars and bands are 95% intervals from resampling runs within category, and grounding is off (Tables 23, 24, and 25).

ties on 13, and relative to no injection B adds 0.076 mean reward, with interval $[+0.001, +0.166]$ (Appendix G). Relevance ranking can miss a useful alternative, and this comparison shows it without relying on the ledger's credit rule. Beyond the planted store, the adaptation rests on outcomes associated with the supplied set rather than on credit to individual members, and it does not carry over to a natural store or to the held-out categories, where useful procedures are present but the two-record learner does not improve selection (Appendix E.4).

5 Discussion

Our findings suggest that progress in agent memory can come from better application and reuse as well as richer storage. Simple construction supports competitive static accuracy, while grounding adds value beyond retrieving experience or repeating task rules. Across two readers, retrieved experience and grounding contribute complementary gains. The fixed-store experiments show a further role for outcomes in correcting selection without rewriting knowledge. They point to learning over grounded procedures as the next step. Attaching each grounded card to a procedure identity would let the ledger track when that procedure helps, bringing grounding and feedback into one evaluated loop. Comparisons from identical saved states could provide more informative credit when several memories are supplied together. Broader procedure families and longer task streams would then test how well these learned preferences transfer beyond the settings studied here.

6 Conclusion

We introduced CaseMem, a memory framework that preserves reusable knowledge and uses task context and subsequent outcomes to guide its application and selection. Segmentation and indexing achieve competitive accuracy on three memory benchmarks without generative construction calls. In agentic evaluation across different model families, combining retrieved experience with grounding improves reward over either component alone, and binding helps beyond a procedural reminder. Separate controlled experiments show that outcome feedback can recover overlooked procedures and reduce harmful reuse while stored knowledge stays fixed. Together, these results provide a practical foundation for agent memory that is economical to build, grounded in the current task, and responsive to evidence from use.

Reproducibility Statement

Code, benchmark adapters, scoring scripts, protocol patches, and per-question outputs will be released later. Section 4.1 and Appendix C document models, prompts, splits, and checks for query parity and frozen-ledger isolation.

AI USE STATEMENT

A coding assistant helped write and debug adapters and analysis scripts and edit text. The authors executed and inspected the code and its outputs. AI tools are used to polish the writing and verify grammar errors.

REFERENCES

Tanzim Ahad, Ismail Hossain, Md Jahangir Alam, Sai Puppala, Syed Bahauddin Alam, and Sajedul Talukder. The misattribution gap: When memory poisoning looks like model failure in agentic AI systems. *arXiv preprint arXiv:2605.22842*, 2026. URL https://arxiv.org/abs/2605.22842.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002. doi: 10.1023/A:1013689704352. URL https://link.springer.com/article/10.1023/A:1013689704352.

Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 19–26. PMLR, 2011. URL https://proceedings.mlr.press/v15/beygelzimer11a.html.

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://ebooks.iospress.nl/doi/10.3233/FAIA251160. arXiv:2504.19413.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.18866. arXiv:2510.18866.

Tao Feng, Chongrui Ye, Tianyang Luo, Jingjun Xu, Xueqiang Xu, Haozhen Zhang, Zhigang Hua, Yan Xie, Shuang Yang, Ge Liu, and Jiaxuan You. ExpGraph: Model-agnostic experience learning with graph-structured memory for LLM agents. *arXiv preprint arXiv:2605.30712*, 2026. URL https://arxiv.org/abs/2605.30712.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515. PMLR, 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026. URL https://arxiv.org/abs/2602.16313.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45836–45853. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.acl-long.2125. URL https://aclanthology.org/2026.acl-long.2125/.

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

Satyen Kale, Lev Reyzin, and Robert E. Schapire. Non-stochastic bandit slate problems. In *Advances in Neural Information Processing Systems*, volume 23, pp. 1054–1062, 2010. URL https://proceedings.neurips.cc/paper/2010/hash/390e982518a50e280d8e2b535462ec1f-Abstract.html.

Junwei Liao, Haoting Shi, Ruiwen Zhou, Jiaqian Wang, Shengtao Zhang, Wei Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Bo Tang, Weinan Zhang, and Muning Wen. MemQ: Integrating Q-learning into self-evolving memory agents over provenance DAGs. *arXiv preprint arXiv:2605.08374*, 2026. URL https://arxiv.org/abs/2605.08374.

Weitao Ma, Xiaocheng Feng, Lei Huang, Xiachong Feng, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Bing Qin. Fine-Mem: Fine-grained feedback alignment for long-horizon memory management. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19666–19684. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.900. URL https://aclanthology.org/2026.acl-long.900/.

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/.

Marina Mao, Alexandr Liu, Pengbo Li, Siheng Li, Bo Zhou, and Xiang Wang. Tree-based credit assignment for multi-agent memory system. *arXiv preprint arXiv:2605.04811*, 2026. URL https://arxiv.org/abs/2605.04811.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25140. arXiv:2509.25140.

Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. REMem: Reasoning with episodic memory in language agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.13530. arXiv:2602.13530.

Jiashuo Sun, Yixuan Xie, Jimeng Shi, Shaowen Wang, and Jiawei Han. TaSR-RAG: Taxonomy-guided structured reasoning for retrieval-augmented generation, 2026. URL https://arxiv.org/abs/2603.09341.

Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 814–823. PMLR, 2015. URL https://proceedings.mlr.press/v37/swaminathan15.html.

Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miroslav Dudík, John Langford, Damien Jose, and Imed Zitouni. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*, volume 30, pp. 3632–3642, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5352696a9ca3397beb79f116f3a33991-Abstract.html.

Jiejun Tan, Zhicheng Dou, Liancheng Zhang, Yuyang Hu, Yiruo Cheng, and Ji-Rong Wen. MemSifter: Offloading LLM memory retrieval via outcome-driven proxy reasoning. *arXiv preprint arXiv:2603.03379*, 2026. URL https://arxiv.org/abs/2603.03379.

Zihao Tang, Xin Yu, Ziyu Xiao, Zengxuan Wen, Zelin Li, Jiaxi Zhou, Hualei Wang, Haohua Wang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. Mnemis: Dual-route retrieval on hierarchical graphs for long-term LLM memory. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23914–23928. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1096. URL https://aclanthology.org/2026.acl-long.1096/.

Yu Wang and Xi Chen. MIRIX: Multi-agent memory system for LLM-based agents. *arXiv preprint arXiv:2507.07957*, 2025. URL https://arxiv.org/abs/2507.07957.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025a. URL https://arxiv.org/abs/2509.25911.

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, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10813. arXiv:2410.10813.

Di Wu, Zixiang Ji, Asmi Kawatkar, Bryan Kwan, Jia-Chen Gu, Nanyun Peng, and Kai-Wei Chang. LongMemEval-V2: Evaluating long-term agent memory toward experienced colleagues. *arXiv preprint arXiv:2605.12493*, 2026. URL https://arxiv.org/abs/2605.12493.

Menglin Xia, Xuchao Zhang, Shantanu Dixit, Paramaguru Harimurugan, Rujia Wang, Victor Rühle, Robert Sim, Chetan Bansal, and Saravan Rajmohan. Memora: A harmonic memory representation balancing abstraction and specificity. *arXiv preprint arXiv:2602.03315*, 2026. URL https://arxiv.org/abs/2602.03315. Accepted to ICML 2026.

Buqiang Xu, Yijun Chen, Jizhan Fang, Ruobin Zhong, Yunzhi Yao, Yuqi Zhu, Lun Du, and Shumin Deng. StructMem: Structured memory for long-horizon behavior in LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 122–146. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-short.12. URL https://aclanthology.org/2026.acl-short.12/.

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 38 (NeurIPS 2025)*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html. arXiv:2502.12110.

Sikuan Yan, Ahmed Bahloul, Ercong Nie, Susanna Schwarzmann, Riccardo Trivisonno, Volker Tresp, and Yunpu Ma. Memory-R2: Fair credit assignment for long-horizon memory-augmented LLM agents. *arXiv preprint arXiv:2605.21768*, 2026a. URL https://arxiv.org/abs/2605.21768.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, and ChengXiang Zhai. PlugMem: A task-agnostic plugin memory module for LLM agents. *arXiv preprint arXiv:2603.03296*, 2026a. URL https://arxiv.org/abs/2603.03296.

Yi Yang, Zhennan Chen, Yihong Zhuang, Tiehan Fan, Yinan Chen, Jian Li, Jian Yang, and Ying Tai. RoMeRL: Balancing feedback coverage and the memory-reward trap in self-evolving agent memory via reduced-order utility states. *arXiv preprint arXiv:2608.02508*, 2026b. URL https://arxiv.org/abs/2608.02508.

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

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. AMA-Bench: Evaluating long-horizon memory for agentic applications. *arXiv preprint arXiv:2602.22769*, 2026. URL https://arxiv.org/abs/2602.22769. Accepted to ICML 2026.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/52f050499cf82fa8efb588e263f6f3a7-Abstract-Conference.html. arXiv:2306.07863.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*, 2025. URL https://arxiv.org/abs/2508.16153.

A Detailed related work

Figure 1 places prior work by research emphasis along construction, application, and feedback; Table 3 organizes it by what feedback changes and what remains inspectable afterward.

Table 3: Where downstream outcome feedback persists. The distinction is not whether a method uses feedback, but whether it changes content, trains a controller, or records use evidence against an inspectable object.

Approach	Feedback changes	Persistent state	Relation to stored content
Content consolidation	memory text or structure	revised/generated memory	the content itself changes
Policy learning	manager or retriever	model parameters or strategy	feedback leaves the item
Object utility (MemRL)	scalar attached to an item	one Q-value per memory	content fixed; one aggregate score
CaseMem (Case + ledger)	**object's reuse ledger**	**exposure and outcome ledger**	**support, applicability, and use remain separate**

Static memory representation and retrieval. Memory systems represent information as extracted facts, linked notes, summaries, typed records, or graphs. Examples include mem0, A-MEM, LightMem, EverMemOS, Memora, Mnemis, StructMem, REMem, MIRIX, PlugMem, and HippoRAG 2 (Chhikara et al., 2025; Xu et al., 2025; Fang et al., 2026; Hu et al., 2026a; Xia et al., 2026; Tang et al., 2026; Xu et al., 2026; Shu et al., 2026; Wang & Chen, 2025; Yang et al., 2026a; Gutiérrez et al., 2025). These designs preserve detail and improve access in different ways. Case connects stored content and its source evidence to a record of later use. Section 2 explains how this interface relates to construction methods and item-level utility learning.

Outcome-aware memory policies. Mem-α learns a construction policy from QA rewards; Memory-R1 trains memory operations with PPO and GRPO (Wang et al., 2025a; Yan et al., 2026b). Fine-Mem assigns evidence-based credit, Memory-R2 compares operations through local re-rollouts, and TreeMem assigns Monte-Carlo credit to builder, summarizer, and retriever agents (Ma et al., 2026; Yan et al., 2026a; Mao et al., 2026). These signals train policies. ReasoningBank distills strategy text from successful and failed episodes (Ouyang et al., 2026). AgentRunbook-C instead stores trajectories as files and uses a coding agent to retrieve supporting evidence at query time (Wu et al., 2026). In contrast, our controlled experiments leave both content and model parameters fixed and update the record attached to each supplied object. A trained manager could also write objects with this interface.

Object-level adaptation. MemRL keeps an episodic memory's content fixed and learns its utility from task feedback; retrieval combines normalized similarity, utility, and exploration within a semantic candidate set (Zhang et al., 2026). Our comparison concerns how use evidence is recorded: CaseMem separates exposure from weighted outcomes and keeps family-specific statistics alongside source evidence. Our shared-store MemRL comparison uses the repaired adapter in the single-object study (Appendix C); pre-repair runs are excluded from the comparison tables. Three concurrent works extend this setting. MemQ (Liao et al., 2026) propagates credit through a provenance graph, ExpGraph (Feng et al., 2026) keeps skills and failure lessons in a graph and updates retrieval utilities from outcomes, and RoMeRL (Yang et al., 2026b) studies misleading rewards shared by co-retrieved memories. Our shared-store, single-object, and fixed-partner diagnostics examine how candidate quality, credit sharing, and context interactions affect outcome-based selection (Appendix G). The exploration term is UCB-inspired (Auer et al., 2002); no regret guarantee is claimed for the combined score.

Attribution. Counterfactual composition tests measure the effect of removing a memory by rerunning the agent without it (Ahad et al., 2026). Such tests are useful references for cheaper credit rules, but require additional executions. We evaluate exposure and behavioral credit; we do not evaluate a counterfactual action-likelihood estimator.

Benchmarks. LoCoMo and LongMemEval test answers from conversational histories (Maharana et al., 2024; Wu et al., 2025). AMA-Bench and LongMemEval-V2 use agent trajectories (Zhao et al., 2026; Wu et al., 2026). MemoryAgentBench evaluates incremental interaction (Hu et al., 2026b). We use MemoryArena for the feedback experiments because it provides execution outcomes and repeated opportunities to retrieve memory during interdependent tasks (He et al., 2026).

B METHOD DETAILS

Table 4: One turn of the flywheel over a fixed store. The right column says what each step may change. The grounding experiments disable ledger updates; the controlled feedback experiments hold everything fixed except the ledger.

Step	Operation	May change
Retrieve	Drop invalid or out-of-scope records; rank the rest by Eq. 3.	nothing
Ground	Fill procedure slots from the observable state; withhold if unresolved.	a computed card
Supply	Give the agent the selected content; record identities R_t.	nothing
Observe	Receive reward r_t and, for behavioral credit, the action trace.	nothing
Update	Apply Eq. 1 to the family ledgers of the records in R_t.	ledger counts

Causal target. For object m in family g, the quantity a retrieval decision would ideally use is the incremental value

$$\Delta(m, g) \;=\; \mathbb{E}\big[R \mid \mathrm{do}(\text{use } m), g\big] \;-\; \mathbb{E}\big[R \mid \mathrm{do}(\text{not use } m), g\big], \tag{4}$$

which ordinary task feedback does not directly provide. The reuse estimate of Eq. 2 is an outcome-conditioned preference, not an estimate of Δ. In the general form the retrieval decision is $R_t = \arg\,\text{top-k}_{m:\,V_t(m)=1}[\mathrm{Rel}(m, q_t) + \mathrm{App}(m, s_t) + U_t(m, g_t) + \mathrm{Bonus}_t(m, g_t)]$; Eq. 3 is the implemented instance.

Object identity and maintenance. The CASE tuple is a common interface; each adapter fills the fields available in its data. Static adapters retain source chunks and retrieval fields. The episode writer produces procedures and exceptions linked to their source episodes and construction logs. The schema has previous-version and original-version links for explicit revisions. The controlled experiments do not revise content, addresses, or scope: they update reuse counts over fixed object identities, and therefore do not test whether a revision correctly resets inherited credit. Explicit invalidation uses a separate path from reward updates; because a valid object can still fail on any task with positive probability, a finite run of failures never identifies invalidity by itself, which is why reward updates and invalidation are kept apart.

Table 5: The three evidence roles in CaseMem. Object fields are fixed within a version; ledger counts change after use.

C	content	knowledge supplied to the agent: a fact, state, procedure, or exception
A	address	compact retrieval key, lexical cues, and available facets
Φ	applicability	observable conditions under which the content may be used
E	provenance	source span or trajectory and its source outcome
g_t	family	task group over which a usage record is shared
$P_{m,g}$	opportunity	number of tasks on which m was supplied in family g
$N_{m,g}$	credit	sum of credit weights assigned to m in family g
$S_{m,g}$	successful credit	credited exposure associated with successful outcomes
$V(m)$	validity	whether this version remains supported
$R_t, w_{t,m}$	supply, weight	exposed objects and each object's share of the outcome
$U(m,g)$	reuse estimate	outcome-derived preference relative to family baseline b_g

Beta-prior interpretation. Conditional on a chosen baseline b_g, a Beta prior $p_{m,g} \sim \mathrm{Beta}(\kappa b_g, \kappa(1-b_g))$ and the weighted Bernoulli likelihood $p_{m,g}^{S_{m,g}}(1-p_{m,g})^{N_{m,g}-S_{m,g}}$ give mean $(\kappa b_g + S_{m,g})/(\kappa + N_{m,g})$. Subtracting b_g gives Eq. 2. In deployment, b_g is re-estimated from pooled observations, so this is an empirical-Bayes shrinkage rule rather than a posterior under a fixed prior. Shrinking toward the same baseline that is subtracted makes an untried object's utility exactly zero. For continuous reward, the same algebra uses the weighted reward sum in S; the binary rule is the default, and the credit-rule ablation is Table 29.

Family baseline. The relative-utility arms pool post-use statistics across objects and smooth the resulting rate:

$$b_g = \frac{\sum_m S_{m,g} + \eta b_0}{\sum_m N_{m,g} + \eta}, \qquad \eta = 5, \quad b_0 = \frac{2}{7}.$$

If the family has no weighted evidence, the implementation uses pooled evidence across families, or b_0 if the ledger is empty; an object's own missing statistics remain $N_{m,g} = S_{m,g} = 0$, hence $U(m,g) = 0$. The pool includes the object being scored, so an object that dominates retrieval can come to define the baseline and have its utility approach zero. The baseline is a reference under the current selection policy, not a memory-free control; the interface separately supports an external memory-free baseline, $b_g = \mathbb{E}[R \mid \text{no injected memory}, g]$, examined in Appendix G.

Admission and use. The outcome of the episode an object was distilled from is source evidence, not an observation of that object's subsequent use. It is retained with provenance and can determine admission; it does not seed P, N, or S. Post-use feedback accumulates only after retrieval and execution. For MemoryArena, r_t is one for an exact match and otherwise the benchmark's partial attribute reward; with $\theta = 0.5$, a positive binary label can reflect a partially successful subtask. Section H traces an object acquired from failure through later updates.

Credit rules. Exposure credit sets $w_{t,m} = 1/|R_t|$. For behavioral credit, let $I_{t,m} \in [0,1]$ be the maximum similarity between the memory text concatenated with its cue anchors and each nonempty recorded action (embedding cosine in dense runs, the configured lexical similarity otherwise), clipped to $[0,1]$. Sum normalization sets $w_{t,m} = I_{t,m}/\max(1, \sum_{j \in R_t} I_{t,j})$, so total credit is at most one and can be below one even at $k = 1$; max normalization sets $w_{t,m} = I_{t,m}/\max_{j \in R_t} I_{t,j}$, giving the highest-scoring object unit credit when the maximum is positive. These rules estimate action agreement, not the probability that an object caused an action. A counterfactual rule based on the change in action log-likelihood when an object is removed is an unevaluated alternative.

Implementation and cost. A shared memory-store module implements the update and selection rules through policy flags and weights; tests verify that matched static, utility, and utility-plus-exploration configurations differ only in the corresponding selection flag. These configurations use no recency, source-outcome, or trust terms in the base retrieval score. The default learning settings are $\kappa = 7$, $\theta = 0.5$, and $\lambda_U = 4$; exploration uses $\lambda_E = 0.05$ inside the utility channel, for an effective coefficient of 0.2. The shortlist setting restricts final selection to the top eight candidates by base score. A report updates at most k objects; exposure credit requires no model call, and dense behavioral credit requires action and memory embeddings. Per retrieval, the scorer aggregates family baselines and exposure totals over the store and scores the candidates; post-outcome updates touch only the supplied records. Grounding parses the current request and renders a card without a model call. The card's reader-token overhead is measured separately in Section 4.3.

C EVALUATION PROTOCOLS

Models and judges. Table 1 reports Qwen3-32B and GPT-4.1-mini readers. LoCoMo is judged by GPT-4o-mini with the lenient prompt for both readers; the AMA-Bench and LongMemEval-S strict judgments are by Qwen3-32B for the Qwen-reader column and by GPT-4o-mini for the GPT-reader column. The complete static tables below retain their labelled judges. The MemoryArena agent is Qwen3-32B (GPT-4.1-mini in the reader replication); the matched per-bundle studies use temperature 0, the ledger streams temperature 1.0. Embeddings are adapter-specific: AMA retrieval uses Qwen3-Embedding-8B, the LoCoMo and LongMemEval-S chunk adapters use BGE-small-en-v1.5, and Table 15 evaluates text-embedding-3-small. A shared endpoint does not imply a shared internal writer or embedder.

Configuration names. CaseMem denotes CASE storage and retrieval; CaseMem+ adds deterministic task grounding. Their matched per-bundle comparison keeps the ledger off. Grounding-only, reminder, utility-only, and utility-plus-exploration rows are component controls. Historical rows labelled static or declared use the ledger-off and ledger-on settings of CaseMem without grounding.

Baselines. The controlled comparisons use static CASE (ledger disabled), utility-only retrieval, and utility plus exploration over identical stores; a shared-store MemRL adapter provides an additional selection rule. End-to-end baselines run released code with patches for endpoints, context limits, and judge routing. Internal models are configuration-specific: the LoCoMo Memora-P row uses a matched Qwen3-32B writer, while its released LongMemEval-S configuration uses GLM-5.

Development and held-out data. The credit rule, reward threshold, shortlist size, and exploration weight were developed on the baking, grocery, and beauty categories of MemoryArena. The configuration was then frozen ($\kappa = 7$, $\lambda_U = 4$, $\lambda_E = 0.05$, $\theta = 0.5$, behavioral weights, top-8 shortlist, dense prior) and evaluated in three fresh streams per category on electronics and home, which had not been inspected during that development. In the ledger streams, memory persists across bundles within a product category; the matched component studies follow the official per-bundle reset. Credit-rule variants and the complete repeated-run summaries appear in Appendix G.

MemRL comparison provenance. Our MemRL adapter ports its retrieval-and-update rule to a shared store: normalized similarity–utility ranking, ϵ-greedy exploration, and per-item Q updates. An earlier adapter generated candidates from full record content but scored similarity on record keys. Results from that adapter are excluded from the comparison tables and retained in the experiment archive. The repaired adapter was evaluated only in the single-object study: three repetitions of 120 subtasks per category, task-specific queries, a dense prior, and a hard top-eight candidate set. Baking scores are 0.225 for MemRL, 0.197 for the pooled learner, and 0.217 for static retrieval; electronics scores are 0.229, 0.225, and 0.223. These shared-store comparisons do not evaluate MemRL's native construction pipeline.

Streams and variance. Adaptation results report mean $\pm$ SD over independent 120-subtask streams; full-benchmark runs contain 180 subtasks per category. Hosted agents are not seedable: ten bundles executed twice at temperature 0 under identical sealed configurations give a test-retest SD of 0.092 in bundle-mean reward over 70 paired arm runs, and no pair reproduced its six step re-

wards. Bootstrap intervals therefore pair arms by bundle and resample whole bundles (2,000 draws unless stated); they are descriptive and do not account for preceding configuration search.

Controls named in the main text. The controlled store contains five procedures with predefined helpful, harmful, or irrelevant roles (listed below) and lets us test a known retrieval error before studying natural records. Stores distilled from evaluation episodes are labelled *naturalized*; *disjoint-source* stores use separate source bundles. A *header* query contains the shared task rules, whereas a *task-specific* query includes the current request. Each feedback arm uses the same frozen store and agent with $k = 2$ unless stated, and only the ledger changes across tasks; each stream contains 120 subtasks, and we report variation across repeated streams and paired stream-level contrasts. Table and figure captions identify these settings.

Stores and their provenance. The controlled experiments fix the memory store to separate construction from selection. Every arm loads the same objects, writes no episodes, and normally supplies two objects per subtask. Four stores are used:

- *Planted*: five procedures derived from real trajectories, with two assigned helpful roles, two harmful roles, and one irrelevant role.
- *Naturalized*: 120 objects distilled from a memoryless GLM-5 agent on the same 20 baking bundles used for evaluation; 81 have positive-outcome sources and 39 have failed sources (a development setting with shared source and evaluation tasks).
- *Disjoint-source*: 69 baking objects and 70 electronics objects distilled from items 20–29; evaluation uses items 0–19.
- *Rebuilt*: 38 baking objects from a writer given each source task block and scored outcome, used only for the construction diagnostic.

Query variants. An interface audit found that the closed-loop runs built the retrieval query from the shared instruction header, which is identical for every subtask, whereas the dense-retrieval reference queried with the full prompt. Results are therefore labelled by query variant: *header-based* for the original development, naturalized, disjoint-source, and held-out streams; in the *task-specific* rerun every arm retrieves with the current-step request, verified identical across arms on every round.

Credit normalization by stream set. Planted and held-out streams use sum-normalized behavioral weights; naturalized, disjoint-source, and corrected-query streams use max-normalized weights. These streams use binary feedback; the single-object comparison of Appendix G uses unit exposure credit.

Planted objects. The five procedures were derived from a 60-subtask no-memory run on the baking category.

- **good_fullsearch** (good): search with the full product phrase from the option description, click the matching product id, then Buy Now.
- **good_validmoves** (good): on a product page only option values, Buy Now, or Back to Search do anything; repeating other clicks ends the episode without a purchase.
- **bad_headersearch** (harmful): search with the section header only and buy the first result.
- **bad_cheapest** (harmful): whatever the preference says, buy the cheapest option.
- **irrelevant_reviews** (irrelevant): how to read more customer reviews.

With lexical similarity over the instruction, the static arm's top two objects were `good_validmoves` and `irrelevant_reviews` in every category.

Protocol checks. Offline probes verify that all arms use the intended store, query, candidate set, and update rule; reassociated feedback is sampled from actual earlier outcomes; final selection is restricted to the shortlist; every learner is frozen before the first test retrieval; shared frozen stores are loaded without merging, so objects are identical across arms; and query hashes are logged on every round. The learning-reserved bundles (home 10–29, electronics 16–29) were executed under fixed-content arms in the matched comparisons and are no longer untouched for a future adaptive study.

Result-to-configuration map. Table 6 states, for each reported result, the store, query, grounding and ledger switches, credit rule, models, repetitions, and implementation version that produced it, and where the complete numbers appear.

Table 6: Configuration behind each reported result. "Per-bundle": every arm starts each bundle with an empty memory (official reset); "persistent": memory carries across bundles within a category. Versions name the immutable package that executed the run; "planted-era" is the shared-store backend of the controlled feedback studies. A dash means not applicable.

Result	Store	Query	Grounding	Ledger (credit)	Agent / reader; judge	Reps	Version	Full numbers
Table 1 (static)	chunk store per benchmark	question	—	off	Qwen3-32B; GPT-4.1-mini replay; judges per caption	1	static adapters	Tables 8, 10, 13, 12
Table 2, persistence block	each system's own, persistent	step	off	static off; declared on (binary)	Qwen3-32B; native judged reward	1 stream per category	historical runs	Table 22
Table 2, matched block	per-bundle, cold start	step	card on / off	off	Qwen3-32B, T=0; string reward	1 (10 bundles re-executed)	03e853d	Table 16
GPT components	per-bundle, cold start	step	as above	off	GPT-4.1-mini, T=0; string reward	1	741a9b2	Table 18
Table 17, grocery rows	none / fixed procedure	step	on vs. off	off	Qwen3-32B, GPT-4.1-mini; string reward	1	—	Table 17
Baseline extension	per-bundle, cold start	step	as above	off	Qwen3-32B; string reward	1	741a9b2 (native), 03e853d (ours)	Table 19
Component repeats	per-bundle, cold start	step	as above	off	Qwen3-32B; string reward	2	03e853d	Table 20
Breadth study	persistent	step	card-only on	off	Qwen3-32B; string reward	1	—	Table 27
Feedback (Tables 23, 24, 25)	planted (5 procedures) or naturalized	header	off	on vs. static (binary helped, pull-decayed bonus)	Qwen3-32B, T=1.0; string reward	6 streams (3 categories $\times$ 2 seeds)	planted-era	Tables 23, 24, 25
Query-matched natural store	disjoint-source natural (69 objects)	step	off	on vs. static (binary)	Qwen3-32B; string reward	3 streams	planted-era	Table 26
Co-exposed records	per-bundle prefix, natural records	step	off	off	Qwen3-32B, T=0; string reward	2	5dad137	Table 31

D COMPLETE STATIC-MEMORY RESULTS

The ledger is disabled throughout. Tables 8–15 are the complete versions of the results summarized in Table 1. Table 7 lists the baselines that were run on only some of the three benchmarks, under the same readers, judges, and token accounting as Table 1.

D.1 AMA-BENCH

AMA-Bench (Zhao et al., 2026) contains 2,496 questions about 208 agent trajectories across six domains, with histories of 6k–289k tokens. Table 8 evaluates the systems in our pipeline with a Qwen3-32B reader and judge. Our matched-depth comparisons use one object per trajectory step: at $k = 5, 10, 15$, CASE improves over dense retrieval by 3.1, 3.6, and 3.0 points. The two main-table configurations differ in entity grouping; their repeated-run means are 0.492 and 0.491, with four runs ranging from 0.476 to 0.505. The representation ablations at the 18k cap are small: the dense channel alone scores 0.401 against 0.414 for the full object, and removing recency gives 0.417, so this test shows no recency benefit. Increasing depth from five to fifteen raises accuracy from 0.409 to 0.474, and a token-overlap diagnostic associates this with better evidence coverage (evidence-absent error share $0.37 \rightarrow 0.29$; reader-error share unchanged at 0.22).

Table 7: Baselines run on a subset of the static benchmarks; readers, judges, and token accounting as in Table 1. Build calls are the generative construction calls on LoCoMo. †Memora-P runs its own pipeline and cannot be replayed: the Qwen column uses the matched Qwen3-32B writer with Memora's own reader, and the GPT column its released GPT-4.1-mini configuration (Table 12); ‡includes query-time compression.

Method	Benchmark	Qwen	GPT	Tok.	Build calls
A-MEM	AMA-Bench	0.384	0.378	1.8k	—
A-MEM	LoCoMo	0.528	0.559	5.0k	5,872
LightMem	LoCoMo	0.702	0.738	5.8k	3,707
LightMem	LongMemEval-S	0.642	0.632	70k‡	—
HippoRAG 2	AMA-Bench	0.460	0.451	6.1k	—
HippoRAG 2	LoCoMo	0.740	0.776	6.3k	11,764
Memora-P†	LoCoMo	0.799	0.856	15.3k	internal

Table 8: AMA-Bench real-world subset (2,496 questions). Every row is measured in our pipeline under the same Qwen3-32B reader and judge, the same Qwen3-Embedding-8B embedder, and a uniform 12k-token context cap; third-party systems run from their released code. Run-to-run noise is about ± 1.5 points. The published table of Zhao et al. (2026) (same reader and judge) puts BM25 at 0.344, A-MEM at 0.319, mem0 at 0.210, and HippoRAG2 at 0.448; our BM25 row reproduces theirs.

Method	Recall	Causal	State-upd.	Abstraction	Avg
matched k=5 objects per question					
BM25	0.338	0.404	0.309	0.237	0.330
Dense (Qwen3-Emb-8B)	0.391	0.455	0.346	0.290	0.378
CASE objects	**0.426**	**0.500**	**0.376**	**0.295**	**0.409**
matched k=10					
BM25	0.398	0.478	0.366	0.280	0.389
Dense	0.434	0.485	0.385	0.341	0.418
CASE objects	**0.474**	**0.522**	**0.445**	0.329	**0.454**
matched k=15					
BM25	0.437	0.490	0.394	0.307	0.417
Dense	0.452	0.502	0.445	0.341	0.444
CASE objects	**0.483**	**0.525**	**0.484**	**0.365**	**0.474**
LLM-extraction memories (released code, their own retrieval)					
mem0	0.275	0.337	0.263	0.210	0.276
A-MEM	0.391	0.451	0.394	0.258	0.384
HippoRAG 2 (released code, $k = 15$)	0.496	0.527	0.423	0.348	0.460
Representation ablations at k=5 (18k-token cap; compare with 0.414 for the full object under that cap)					
dense channel only	0.410	0.480	0.372	0.314	0.401
no recency term	0.429	0.503	0.391	0.309	0.417
entity grouping, k=15 (vs. 0.478)	0.498	0.564	0.459	0.379	0.484

D.2 LoCoMo

LoCoMo contains ten multi-session conversations with 5,882 turns (Maharana et al., 2024); we evaluate the 1,540 questions in categories 1–4 used by Chhikara et al. (2025). Each released baseline keeps its retrieval protocol: mem0 extracts dated memories per turn and retrieves up to 200 items with its reasoning answer prompt; A-MEM constructs timestamped notes with top-10 retrieval; LightMem uses turn pairs and top-60 retrieval; Memora-P uses the official runner's cue index and prompted retrieval; HippoRAG 2 indexes turns through OpenIE at $k = 8$. CASE uses 600-character chunks with session information and a direct-answer prompt at $k = 8$. In the GPT-reader column of Table 1, the stores of mem0, A-MEM, LightMem, and HippoRAG 2 are those built with the Qwen3-32B writer and only the saved answer prompts are replayed through GPT-4.1-mini; Memora-P cannot be replayed, so its GPT-column value comes from its released configuration, which builds, retrieves, and answers with GPT-4.1-mini and reproduces the paper's reported 0.863 under Memora's own evaluator (0.862 on all 1,540 questions; 0.856 under our run of the same judge).

Table 9: Cost on AMA-Bench (2,496 questions), same Qwen3-32B reader; judge calls excluded. Prompt tokens are the median over the run's reader calls, the same measurement Table 1 reports. Memory build for CASE is embedding-only.

Method	Acc.	Calls / Q	Input tok.	Output tok.	Min.
CASE objects, k=15	0.474	1.00	5.7k	87	7.9
mem0 (internal extraction not logged)	0.276	1.00 + extraction	0.6k	74	43.1
A-MEM (sequential; internal writing not logged)	0.384	2.00 + note construction	1.8k	77	703
HippoRAG 2 (internal OpenIE not logged)	0.460	2.00 + extraction	6.1k	89	64

Table 10 gives the Qwen3-32B results under the strict judge. Retrieval units and context sizes differ, so equal item counts would not equalize reader cost: with the reader and mem0's answer prompt fixed, CASE at $k = 32$ reaches 0.723 with 5.6k median prompt tokens against 0.674 and 9.4k tokens for mem0 at top-200. Table 11 reports the measured construction and answer costs, and Table 12 the judge and reader sensitivity of every row. A hybrid BM25 floor matched to dense retrieval in every respect except the ranking function (same chunks, de-duplication, $k = 8$, and prompt) scores 0.555 on LoCoMo and 0.504 on LongMemEval-S under the Qwen judge.

Table 10: LoCoMo (1,540 questions, categories 1–4), Qwen3-32B reader and strict Qwen3-32B judge for every row; released baselines under their own protocols with only the model endpoint redirected. Panel A compares systems at their own retrieval settings; Panel B holds the reader and answer prompt (mem0's) fixed. Published rows use GPT-4o-mini and mem0's lenient judge and are not comparable.

Method	Multi-hop	Temporal	Open-domain	Single-hop	Overall
Panel A: shared reader; retrieval settings shown					
Dense RAG (same chunks, $k = 8$)	0.390	0.280	0.438	0.698	0.538
A-MEM, released protocol, top-10	0.241	0.330	0.302	0.546	0.430
LightMem, released protocol, top-60	0.411	0.542	0.458	0.728	0.614
Memora-P, official runner, matched Qwen3-32B writer	0.496	0.639	0.469	0.832	**0.708**
HippoRAG 2, released code, $k = 8$	0.433	0.492	0.521	0.747	0.622
CaseMem, $k = 8$	0.397	0.502	0.531	0.775	0.634
Panel B: shared reader and answer prompt (mem0's); different context sizes					
mem0, released default, top-200 (9.4k prompt tokens)	0.507	0.682	0.531	0.743	0.674
CASE objects, $k = 32$ (5.6k prompt tokens)	0.507	**0.732**	**0.562**	**0.811**	**0.723**
Reference					
Full context, Qwen3-32B, 14k-token window	0.426	0.271	0.500	0.620	0.504
Full context, GLM-5 reader, 110k window	0.610	0.262	0.615	0.880	0.685
Published, GPT-4o-mini reader and judge: full context 0.729, Mem0 0.669, A-MEM 0.484 (Chhikara et al., 2025)					

Table 11: What the LoCoMo rows cost (measured, same endpoint). *Ingestion* is the one-time cost of building the ten conversation stores from 5,882 turns; mem0's 51M tokens arise because each per-turn extraction call re-sends the memories already stored. *Answer* is the per-question cost with prompt and completion tokens separated. Our objects are built without generative calls; embedding time is included.

	Ingestion (10 conversations)			Answer (per question)		
Method	LLM calls	prompt tokens	wall-clock	seconds	prompt tok.	completion tok.
mem0, top-200	5,882	51.5M	2.3 h	4.6	9,390	598
A-MEM, top-10	5,872	5.9M	4.4 h	0.8	1,208	37
LightMem, top-60	3,707	2.8M	2.4 h	0.6	3,941	21
Memora-P, official runner, matched writer	internal	not logged	63 min	3.8	≈15,000	not logged
HippoRAG 2, $k = 8$	11,764	4.5M	26 min	1.1	3,416	75 (+1 query-side LLM call)
CASE objects, $k = 8$	0	0	113 s	0.6	1,183	35
CASE objects, $k = 32$, direct-answer prompt	0	0	113 s	1.0	4,532	38
CASE objects, $k = 8$, mem0's answer prompt	0	0	113 s	3.2	2,566	454
CASE objects, $k = 32$, mem0's answer prompt	0	0	113 s	5.7	5,621	512

D.3 LONGMEMEVAL-S

LongMemEval-S (Wu et al., 2025) attaches about 40 timestamped chat sessions ($\approx$ 115k tokens) to each of 500 questions. Our configuration chunks sessions into roughly 600-character windows, also indexes each user turn as a record, embeds each record's full text with BGE-small-en-v1.5, and re-

Table 12: LoCoMo judge and reader sensitivity for every LoCoMo row of Tables 1 and 7: the same saved predictions scored by the strict judge and by the mem0/Memora answer-accuracy judge (both GPT-4o-mini), with the GPT-4.1-mini reader replaying the identical retrieved contexts. Memora-P rows come from their own runner and have no replay. The lenient judge raises every row by 0.12–0.25 and preserves the ordering at the top under both readers; under the strict judge with the Qwen reader, released Memora-P (0.679, own GPT-4.1-mini reader) is above CaseMem (0.642). Memora rows cover the 1,518 questions shared with the matched-writer run; Memora's own evaluator, which uses the same lenient prompt, gives 0.862 on all 1,540 questions against the reported 0.863, and its full-context GPT-4.1-mini reference is 0.825 (Xia et al., 2026).

	Qwen3-32B reader		GPT-4.1-mini reader	
Method	strict	mem0/Memora	strict	mem0/Memora
BM25	0.490	0.671	0.518	0.677
Dense retrieval	0.475	0.666	0.509	0.681
Hybrid dense+BM25, RRF over the same chunks, $k = 8$	0.524	0.704	0.543	0.723
mem0, top-200	0.574	0.819	0.613	0.875
A-MEM, top-10	0.390	0.528	0.423	0.559
LightMem, top-60	0.532	0.702	0.559	0.738
HippoRAG 2, $k = 8$	0.532	0.740	0.593	0.776
Memora-P, matched Qwen writer (own reader)	0.632	0.799	—	—
Memora-P, released configuration (own GPT-4.1-mini reader)	—	—	0.679	0.856
CaseMem, $k = 8$, direct-answer prompt	0.550	0.749	0.602	0.783
CaseMem, $k = 32$, mem0's answer prompt	0.642	0.842	0.685	0.889

trieves dense-only at a fixed depth; the reader receives the retrieved text and its session date. Table 13 reports the full set under the strict Qwen3-32B judge. Fixed-depth CASE reaches 0.630 at $k = 8$, improving over dense retrieval by 10.8 points and mem0 by 10.4, with the largest differences on temporal reasoning (0.474 vs. 0.323) and single-session assistant questions (0.929 vs. 0.607). LightMem reaches 0.642 at about 920 seconds and 71k prompt tokens per question, including haystack compression. Memora-P's released configuration, with its GLM-5 writer and Qwen3-32B answers, reaches 0.792 under the strict judge.

Table 13: LongMemEval-S, all 500 questions, Qwen3-32B reader unless marked GLM-5, strict Qwen3-32B judge. "Objects" is CASE with chunk and user-turn records, full-text dense keys, and dense-only retrieval.

Method	Know. upd.	Multi-sess.	Single (user)	Single (asst.)	Single (pref.)	Temporal	Overall
CASE objects, GLM-5 reader	0.744	0.526	0.900	0.929	0.833	0.526	0.676
Full context (GLM-5, 110k)	0.667	0.436	0.886	0.964	0.367	0.414	0.584
LightMem, released code	0.756	0.617	0.914	0.357	0.700	0.564	**0.642**
CASE objects, k+2 on multi-session by question type	0.705	0.466	0.886	0.911	0.667	0.519	0.638
CASE objects, fixed k=8 (main text)	0.705	0.451	0.886	0.929	0.767	0.474	0.630
CASE objects, fixed k=10	0.705	0.436	0.900	0.964	0.633	0.526	0.638
Dense RAG (same chunks)	0.667	0.451	0.786	0.607	0.567	0.323	0.522
mem0 OSS (released code)	0.603	0.429	0.814	0.589	0.733	0.353	0.526

Retrieval configurations on LongMemEval-S. Table 14 isolates what carries the static gain. Dense-only retrieval scores 0.618 against 0.478 for the multichannel configuration; the dense-only setting disables lexical fusion, entity boosting, score normalization, cue expansion, and recency together, so this is a configuration comparison rather than a single-cue ablation. A second execution of the dense-only setting scores 0.620, so run-to-run variation of this pipeline is about 0.002 overall with per-category swings up to 0.04. Removing date handling (session-date prefix, date anchors, and temporal query expansion) lowers the dense-only setting to 0.564, with the loss concentrated in temporal reasoning (0.481 → 0.308) and knowledge updates (0.654 → 0.538), while single-session assistant questions are slightly higher without dates (0.982 vs. 0.911). One execution per cell; a development finding.

D.4 LOCOMO UNDER A SECOND EMBEDDING MODEL

Three static configurations are repeated with text-embedding-3-small and compared with their BGE-small-en-v1.5 condition on the same 1,540 questions, with gpt-4.1-mini as reader and gpt-4o-mini as strict judge. The $k = 8$ advantage over dense retrieval is 0.093 under both embedders and both

Table 14: LongMemEval-S retrieval configurations, 500 questions per row, Qwen3-32B reader and strict Qwen judge, $k = 8$. Dense-only disables recency; the multichannel "Off" row pins the clock so recency cannot distinguish records.

Retrieval configuration	Recency ranking	Accuracy
Dense only	Off	**0.618**
Dense only, second execution	Off	0.620
Dense only, date handling removed	Off	0.564
Multichannel	On	0.478
Multichannel	Off	0.486

methods improve by 0.053 with the second embedder (Table 15), so the relative advantage survives the change rather than depending on it.

Table 15: Static LoCoMo retrieval under two embedding conditions, 1,540 questions, GPT reader and strict GPT judge, one pass per cell. The $k = 32$ row uses mem0's answer prompt; both $k = 8$ rows use the direct-answer configuration.

Method	BGE-small-en-v1.5	text-embedding-3-small
Dense retrieval, $k = 8$	0.5091	0.5623
CaseMem, $k = 8$	0.6019	0.6552
CaseMem, $k = 32$, mem0 prompt	0.6851	0.7123
CaseMem minus dense, $k = 8$	$+0.0929$	$+0.0929$

E Complete agent-task results

Every table here uses the protocols of Appendix C; header-query and query-matched runs are labelled in each caption.

E.1 Matched per-bundle comparison

Table 16 gives the per-category rewards behind the five-category ablations of Table 17: eight arms on all 150 bundles, every arm from a fresh agent and an empty memory at each bundle start, one execution per bundle and arm at temperature 0, arm order rotated across bundles. Contrasts are paired bundle means with bundle-bootstrap 95% intervals. The mem0 writer runs inside the mem0 client and its calls are not instrumented; its store is built from the bundle's own earlier steps with the Qwen3-32B writer.

Table 16: Matched per-bundle comparison of CaseMem, CaseMem+, and component controls (Qwen3-32B, 30 bundles per category, ledger off). Paired contrasts over all 150 bundles support the five-category ablations in Table 17.

Arm	Baking	Beauty	Electronics	Grocery	Home	Mean
No memory	0.205	0.173	0.269	0.164	0.183	0.199
Long context (12k history)	0.108	0.091	0.178	0.170	0.245	0.158
Dense over episodes	0.159	0.177	0.206	0.173	0.260	0.195
mem0	0.155	0.190	0.223	0.147	0.176	0.178
Generic reminder	0.194	0.184	0.258	0.178	0.199	0.202
CaseMem+ w/o CASE retrieval (card-only)	0.214	0.192	0.268	0.216	0.229	0.224
CaseMem	0.187	0.161	0.242	0.173	0.241	0.201
CaseMem+	0.221	0.237	0.269	0.296	0.221	0.249

Paired contrasts (mean difference, 95% interval, bundles $+/=/-$). Card $-$ none: $+0.025$ $[+0.007, +0.041]$, 90/7/53; card $-$ reminder: $+0.021$ $[+0.006, +0.037]$, 81/5/64; reminder $-$ none: $+0.004$ $[-0.009, +0.016]$. CaseMem+ $-$ CaseMem: $+0.048$ $[+0.027, +0.069]$, 99/10/41; CaseMem+ $-$ card: $+0.025$ $[+0.009, +0.042]$, 78/16/56; CaseMem $-$ none: $+0.002$ $[-0.016, +0.021]$.
Dense $-$ none: -0.003 $[-0.026, +0.019]$; mem0 $-$ none: -0.021 $[-0.038, -0.003]$; long context $-$ none: -0.041 $[-0.063, -0.019]$. Interaction $(\mathrm{SP} - \mathrm{S}) - (\mathrm{P} - \mathrm{N})$: $+0.023$ $[-0.001, +0.048]$.

Table 17: Full-set component comparisons with two readers: five categories, 150 bundles per reader, ledger off. Paired string-reward differences and descriptive bundle-bootstrap 95% intervals; card-only removes CASE retrieval from CaseMem+. Exact-target counts are in Table 18.

Reader	Contrast	Δ	95% interval
Qwen3-32B; all 150 bundles			
Qwen	CaseMem − none	$+0.002$	$[-0.016, +0.021]$
Qwen	card − none	$\mathbf{+0.025}$	$[+0.007, +0.041]$
Qwen	card − reminder	$\mathbf{+0.021}$	$[+0.006, +0.037]$
Qwen	CaseMem+ − CaseMem	$\mathbf{+0.048}$	$[+0.027, +0.069]$
Qwen	CaseMem+ − card	$\mathbf{+0.025}$	$[+0.009, +0.042]$
GPT-4.1-mini; all 150 bundles			
GPT	CaseMem − none	$\mathbf{+0.030}$	$[+0.010, +0.050]$
GPT	card − none	$\mathbf{+0.029}$	$[+0.005, +0.054]$
GPT	card − reminder	$\mathbf{+0.034}$	$[+0.012, +0.057]$
GPT	CaseMem+ − CaseMem	$\mathbf{+0.036}$	$[+0.012, +0.062]$
GPT	CaseMem+ − card	$\mathbf{+0.038}$	$[+0.018, +0.056]$

Full-set GPT component replication. The five component arms are repeated with GPT-4.1-mini on all 150 bundles with fresh state at each bundle start, temperature 0, and the ledger disabled; the dataset combines 139 sealed bundles with 11 disjoint recovery bundles after an interruption, with identical configurations, task and package hashes, and complete arm coverage verified before combining. Retrieved experience improves reward over no injection with this reader, and combining it with grounding improves over either component alone (Table 17); exact-target counts do not show a corresponding gain over retrieval alone (Table 18).

Table 18: Full-set GPT-4.1-mini component replication: five categories, 150 bundles, 900 steps per arm, ledger off. Exact-target success is a separate endpoint from mean string reward.

Method	String reward	Exact target (/900)
No injection	0.288	102
Generic reminder	0.283	102
Card-only	0.317	99
CaseMem	0.318	132
CaseMem+	**0.355**	130

Baseline extension on a frozen 50-bundle subset. ReasoningBank, A-MEM, and LightMem run under the same per-bundle protocol on items 0–9 of every category, and Table 19 restricts every arm, ours included, to those 50 bundles. CaseMem+ has the highest mean and beats ReasoningBank with an interval above zero; its margins over LightMem and A-MEM, and theirs over no memory, include zero at this size. Native writers run inside their own clients and are not instrumented. Fifty-bundle means are not comparable with the 150-bundle means above.

Component interaction. Complementary gains compare the combination with each component alone. For the full-set Qwen run, the interaction estimate $(\mathrm{SP} - \mathrm{S}) - (\mathrm{P} - \mathrm{N})$ is $+0.023$ $[-0.001, +0.048]$, where SP is retrieval plus grounding, S is retrieval, P is card-only, and N is no injection. Thus, the component contrasts support the combination without establishing a superadditive effect.

Repeatability of the component gains. Two further sealed executions of CaseMem, the grounded card, and CaseMem+ on a predeclared 30-bundle subset (items 0, 5, 10, 15, 20, 25 of every category; arm order reversed in the second) test whether the component contrasts repeat (Table 20). Repetitions are averaged within each bundle before contrasting, and the bootstrap resamples bundles. The grounding gain keeps its sign ($+0.026$, $[-0.001, +0.053]$); the retrieval gain over the card does not (-0.015, $[-0.042, +0.011]$, against $+0.034$ in the original execution). The per-arm test-retest SD of bundle-mean reward is 0.08–0.10, so effects of 0.03 need the full 150 bundles. These bundles had already been evaluated; this is a repeatability study, not a held-out test.

Table 19: Baseline extension on the same 50 bundles (items 0–9 per category), Qwen3-32B agent, per-bundle protocol: bundle-mean reward, exact-target fraction, reader prompt tokens per subtask, and the paired contrast against no memory with a bundle-bootstrap 95% interval and bundles improved/tied/worse.

Arm	Mean	Exact	Tokens	vs. no memory
No memory	0.196	0.040	17.7k	—
Long context (16k)	0.145	0.013	58.4k	-0.051 $[-0.091, -0.013]$ 15/3/32
Dense over episodes	0.184	0.043	16.9k	-0.013 $[-0.049, +0.027]$ 20/1/29
mem0	0.162	0.033	17.6k	-0.034 $[-0.072, +0.004]$ 19/5/26
ReasoningBank	0.173	0.027	23.7k	-0.024 $[-0.054, +0.007]$ 16/7/27
A-MEM	0.203	0.037	17.6k	$+0.006$ $[-0.029, +0.041]$ 26/3/21
LightMem	0.215	0.040	15.5k	$+0.019$ $[-0.014, +0.055]$ 22/4/24
Ungrounded procedure	0.196	0.033	15.7k	-0.000 $[-0.024, +0.024]$ 19/10/21
Grounded procedure	0.197	0.023	17.2k	$+0.001$ $[-0.029, +0.029]$ 26/4/20
CaseMem	0.190	0.040	12.7k	-0.007 $[-0.044, +0.029]$ 19/7/24
CaseMem+	**0.237**	**0.050**	15.9k	$+0.040$ $[-0.001, +0.082]$ 26/5/19
Pairwise contrasts against CaseMem+ on the same bundles (category-stratified bundle bootstrap)				
CaseMem+ − LightMem	—	—	—	$+0.024$ $[-0.007, +0.057]$ 28/2/20
CaseMem+ − A-MEM	—	—	—	$+0.041$ $[-0.002, +0.080]$ 31/1/18
CaseMem+ − ReasoningBank	—	—	—	$+0.066$ $[+0.035, +0.098]$ 35/2/13

Table 20: Component repeats on 30 predeclared bundles, Qwen3-32B agent, per-bundle protocol. Paired contrasts with category-stratified bundle-bootstrap 95% intervals and bundles improved/tied/worse; the aggregate averages the two repetitions within each bundle first. The original run is restricted to the same bundles and reported separately.

Execution	CaseMem / card / CaseMem+	CaseMem+ − CaseMem	CaseMem+ − card
Repetition 1	0.203 / 0.264 / 0.233	$+0.029$ $[-0.008, +0.066]$ 14/3/13	-0.031 $[-0.065, +0.006]$ 11/3/16
Repetition 2 (reversed order)	0.208 / 0.231 / 0.231	$+0.023$ $[-0.014, +0.060]$ 18/0/12	$+0.000$ $[-0.046, +0.040]$ 16/3/11
Paired aggregate	0.206 / 0.247 / 0.232	$+0.026$ $[-0.001, +0.053]$ 16/0/14	-0.015 $[-0.042, +0.011]$ 14/1/15
Original run, same bundles	—	$+0.038$ $[+0.007, +0.069]$ 16/3/11	$+0.034$ $[+0.001, +0.070]$ 16/2/12

Grounding controls with retrieval present. A contemporaneous four-arm run on the same 30 bundles adds the two controls the full set lacks: CaseMem retrieval with the procedure left unresolved (the reminder replaces the card, and the card-sized slot is reserved so the reminder receives no extra retrieved text) and dense retrieval with the identical grounded card (Table 21). Binding beyond the unresolved procedure is positive with retrieval present; CaseMem retrieval under identical grounding is ahead of dense retrieval with an interval that touches zero; dense retrieval differs from card-only by $+0.004$ $[-0.043, +0.047]$. Exact-target success favours CaseMem+ here (0.083 against 0.044). Across the four executions of the card and CaseMem+ arms on these bundles, the retrieval-over-card increment is $+0.034$, -0.031, 0.000, and $+0.051$.

Table 21: **Grounding improves reward beyond a reminder with retrieval held fixed.** On 30 predeclared bundles, replacing an unresolved procedure with a grounded card raises mean reward from 0.212 to 0.275. Qwen3-32B agent, per-bundle reset, ledger off. Columns report mean string reward, exact-target fraction, and reader prompt tokens per subtask. Contrasts give bundle-bootstrap 95% intervals and counts of bundles improved/tied/worse.

Arm	Mean	Exact	Tokens
Grounded card only	0.223	0.044	17.2k
CaseMem + unresolved procedure	0.212	0.044	14.9k
Dense retrieval + grounded card	0.227	0.056	19.9k
CaseMem+ (CaseMem + grounded card)	**0.275**	**0.083**	14.2k
CaseMem+ − CaseMem+unresolved: $+\mathbf{0.063}$ $[+0.016, +0.111]$ 19/2/9			
CaseMem+ − dense+card: $+0.047$ $[-0.003, +0.095]$ 19/0/11			
dense+card − card: $+0.004$ $[-0.043, +0.047]$ 15/1/14			
CaseMem+ − card: $+0.051$ $[+0.012, +0.093]$ 16/4/10			

E.2 FULL BENCHMARK UNDER PERSISTENCE AND UNDER THE OFFICIAL PROTOCOL

Table 22 covers five categories with 30 bundles each (900 subtasks) under two protocols: memory persisting within a category, and the official per-bundle reset. Our configurations distill at most two site-generic objects per successful episode, query with the current step, and use a hard top-eight shortlist. Under persistence, static CASE scores 0.190 against mem0 at 0.192, LightMem at 0.186, and no memory at 0.174; the ledger-on configuration scores 0.194; long context has the highest mean (0.200) with category scores from 0.011 to 0.303. Under the official protocol, static CASE (0.179), dense retrieval (0.178), and ledger-on CASE (0.178) are close, mem0 scores 0.157, and no memory 0.158. One stream per cell; no system completes a whole bundle under either protocol.

Table 22: Full MemoryArena shopping benchmark (5 categories $\times$ 30 bundles, 900 subtasks), Qwen3-32B agent: string reward per category, five-category mean, judged mean, and subtasks attaining full string reward (of 900). Full reward need not imply an exact product-ID match. HippoRAG 2 uses our online writer with independently evolving stores.

Method	baking	grocery	beauty	electronics	home	mean	judged	full reward
Persistence within a category (every system)								
No memory	0.198	0.131	0.119	0.227	0.195	0.174	0.253	32
BM25 over episodes	0.167	0.225	0.081	0.221	0.140	0.167	0.230	53
Dense retrieval over episodes	0.119	0.081	0.059	0.271	0.221	0.150	0.197	37
Long context (16k tokens)	0.108	0.011	0.284	0.294	0.303	0.200	0.268	49
A-MEM	0.177	0.089	0.063	0.243	0.266	0.168	0.225	35
mem0	0.200	0.144	0.175	0.223	0.219	0.192	0.276	33
LightMem	0.181	0.135	0.141	0.271	0.204	0.186	0.268	29
HippoRAG 2 retriever over our objects	0.183	0.140	0.170	0.234	0.224	0.190	0.273	35
CaseMem, ledger off	0.185	0.145	0.177	0.222	0.223	0.190	0.275	41
CaseMem, ledger on (utility + exploration)	0.210	0.114	0.198	0.249	0.199	0.194	0.281	37
Official per-bundle initialisation (every system)								
No memory	0.175	0.123	0.117	0.199	0.177	0.158	0.234	24
BM25 over episodes	0.150	0.134	0.111	0.224	0.233	0.170	0.247	27
Dense retrieval over episodes	0.171	0.168	0.143	0.194	0.217	0.178	0.255	38
Long context (16k tokens)	0.141	0.160	0.087	0.210	0.264	0.172	0.243	41
A-MEM	0.152	0.135	0.110	0.210	0.241	0.170	0.245	36
mem0	0.170	0.127	0.137	0.171	0.181	0.157	0.233	29
HippoRAG 2 retriever over our objects	0.169	0.099	0.148	0.239	0.196	0.170	0.246	30
CaseMem, ledger off	0.187	0.156	0.152	0.220	0.179	0.179	0.249	34
CaseMem, ledger on (utility + exploration)	0.212	0.122	0.114	0.241	0.202	0.178	0.252	35

E.3 CONTROLLED FEEDBACK EXPERIMENTS

Table 23 summarizes the shared-store mechanism tests and Table 24 gives the full comparison by retrieval regime.

Natural stores. On the naturalized store with lexical retrieval, utility-only selection scores 0.217 versus 0.152 for static retrieval in one stream: static retrieval repeats the same pair, while the learner promotes a procedure from a failed source episode after a positive later outcome (Section H). Under the dense prior over the same store, utility-only and static scores are 0.204 and 0.203, and each released system's own retriever over the same frozen objects scores 0.226–0.240, all within the ± 0.03 run-to-run noise of one another; the reassociated-outcome, equal-credit, and plain-content controls on this store lie within 0.035 of static.

Held-out categories. The frozen exploration-enabled configuration obtains 0.245 mean reward on the six held-out-category streams, compared with 0.257 for its static counterpart; the paired difference is -0.013, with two streams improving and four declining (Table 25); the utility-only arm is -0.011. The logs show coupled selection and credit: in both categories the static dense prior repeatedly supplies the full-search and harmful header-search procedures together, their learned estimates rise and fall together, and the harmful partner remains on 118 of 120 home rounds. Nearly shared credit is a measured property of the implementation and a candidate contributor to this result; the experiment does not isolate it from memory quality or candidate selection.

Table 23: Ledger ablations on MemoryArena (fixed planted store, grounding off, header query, string reward; mean ± SD over six 120-subtask streams per condition). "U+E" enables utility and exploration; "utility" disables exploration. The transfer row freezes the same settings for six held-out streams. Outcome-association rows add descriptive paired 95% intervals (streams resampled within category) and streams improving.

Test	Contrast	Reference	Outcome-aware	Result
Discovery	useful object outside lexical pair; found in 5/6 streams	static 0.173 ± 0.030	U+E 0.209 ± 0.079	$+0.037$
Correction	harmful object inside dense pair; reduced exposure	static 0.160 ± 0.034	utility 0.210 ± 0.090	$+0.049$
Outcome association	utility; reassociate outcomes	reassoc. 0.165 ± 0.043	correct 0.210 ± 0.090	$+0.044$ $[+0.003, +0.086]$ 4/6 streams
Outcome association	U+E; reassociate outcomes	reassoc. 0.143 ± 0.049	correct 0.196 ± 0.057	$+0.053$ $[+0.019, +0.086]$ 5/6 streams
Transfer	frozen held-out categories	static 0.257 ± 0.014	U+E 0.245 ± 0.028	-0.013; 2/6

Table 24: Shared-store adaptation on MemoryArena web shopping: identical frozen store per arm, Qwen3-32B agent, header query, grounding off, and string reward. Planted rows give mean ± SD over six independent streams; naturalized rows give one 120-subtask stream. The columns separate utility from exploration under each retrieval prior.

Regime	Store, prior	Static	+ Utility	+ Utility + exploration	Δ utility	Δ declared
Useful memory buried	planted, lexical	0.173 ± 0.030	0.150 ± 0.040	$\mathbf{0.209 \pm 0.079}$	-0.023	$+0.037$
Harmful memory surfaced	planted, dense	0.160 ± 0.034	$\mathbf{0.210 \pm 0.090}$	0.196 ± 0.057	$+0.049$	$+0.036$
Natural store, weak prior	naturalized, lexical	0.152	**0.217**	0.205	$+0.065$	$+0.053$
Natural store, strong prior	naturalized, dense	0.203	0.204	0.170	$+0.001$	-0.033

Query-matched comparison on a disjoint-source store. Under the header-based query, the utility arm improves on its matched static arm on the baking disjoint-source store (0.224 vs. 0.187) but not on electronics (0.191 vs. 0.207), with one stream per arm. Repeating the baking comparison with identical task-specific queries, verified on every round, over three streams (Table 26) gives mean string reward 0.179 with no memory, 0.226 for static CASE, 0.220 for dense retrieval, 0.205 for the utility-only learner, and 0.207 for the declared configuration; both learners are below static on all three streams. Retrieval now varies with the subtask (16 distinct injected pairs per static stream, 8 to 41 per utility stream, and 43 to 46 per declared stream, against one under the header query). The corrected comparison does not show a mean-reward advantage for the ledger.

E.4 FEEDBACK BEYOND THE PLANTED STORE

This section collects the feedback results that go beyond the planted store of Section 4.4: a natural store, credit within a supplied set, and transfer to the held-out categories.

A failed source episode can still teach a useful procedure. The baking disjoint-source store contains 69 records from separate source bundles. With the current request used in every arm, static CaseMem scores 0.226, versus 0.179 without memory and 0.220 for dense retrieval (Table 26). Supplying individual records throughout the stream reveals two search procedures at 0.249 and 0.242, close to a fixed human procedure at 0.252. Both came from failed episodes and would be discarded by success-only admission. The two-record learner scores 0.207: the useful knowledge is present, and selecting it from shared outcome feedback is the harder problem.

Outcome association and individual credit. A task produces one outcome even when two memories are supplied. In a judge-labelled audit of 840 memory–action pairs, the behavioral uptake proxy has AUC 0.36 (cluster-bootstrap 95% interval 0.17–0.52) and gives nearly uniform weights. Replacing it with equal credit changes mean reward by 0.030 for utility-only retrieval and 0.003

Table 25: Held-out evaluation on the two MemoryArena categories never used during development (electronics, home): planted store, frozen configuration (dense prior, binary feedback, sum-normalized weights, top-8 shortlist), three fresh streams per category, string reward per 120-subtask stream; mean ± SD over the six streams and the paired difference against the static arm of the same stream.

Category	Arm	seed 1 / 2 / 3	mean ± SD	paired Δ vs static
Electronics	Static (ledger off)	0.265 / 0.246 / 0.282	0.264 ± 0.018	—
	+ Utility	0.312 / 0.297 / 0.236	0.282 ± 0.040	$+0.018$ (2 up, 1 down)
	+ Utility + exploration (declared)	0.246 / 0.273 / 0.247	0.255 ± 0.015	-0.009 (1 up, 2 down)
Home	Static (ledger off)	0.249 / 0.256 / 0.246	0.250 ± 0.005	—
	+ Utility	0.220 / 0.217 / 0.196	0.211 ± 0.013	-0.039 (0 up, 3 down)
	+ Utility + exploration (declared)	0.276 / 0.201 / 0.226	0.234 ± 0.038	-0.016 (1 up, 2 down)
Both (6 streams)	Static		0.257 ± 0.014	—
	+ Utility		0.246 ± 0.047	-0.011 (2 up, 4 down)
	+ Utility + exploration (declared)		0.245 ± 0.028	-0.013 (2 up, 4 down)

Table 26: Query-matched content–selection diagnostic on the baking disjoint-source store (69 objects; items 0–19; dense prior; $k = 2$; task-specific query in every arm; three independent streams; string reward mean ± SD, judged mean, exact matches per stream, and the paired difference against static). Row 0 injects nothing; rows A–D share the store; row F supplies a fixed procedure. Rows B–D are paired by stream; rows 0, A, and F are from separate executions and compared against the mean of row B.

	Arm	String	Judged	Exact (s1 / s2 / s3)	paired Δ vs static
0	No memory	0.179 ± 0.018	0.252	2 / 2 / 8	-0.047 (unpaired)
A	Dense retrieval over the store, no learning	0.220 ± 0.006	0.317	5 / 4 / 4	-0.006 (unpaired)
B	CASE static (ledger off)	0.226 ± 0.016	0.313	10 / 9 / 8	—
C	CASE + utility	0.205 ± 0.008	0.296	6 / 4 / 5	-0.021 (0 up, 3 down)
D	CASE + utility + exploration (declared)	0.207 ± 0.011	0.292	5 / 6 / 6	-0.019 (0 up, 3 down)
F	Fixed human procedure, every round	0.252 ± 0.025	0.354	8 / 12 / 11	$+0.026$ (unpaired)

for utility plus exploration, within stream variation. The demonstrated adaptation therefore uses outcomes associated with the supplied set; it does not reliably distinguish which member helped. Fixed-partner tests show that adding another record can reduce the value of an otherwise useful procedure (Appendix G).

Transfer across categories and memory stores. When frozen settings transfer to six electronics and home streams, the learner scores 0.245 against 0.257 for static retrieval (Figure 4a). The natural-store result above has the same difficulty: useful candidates exist, but the two-record learner does not improve selection. Single-record studies find small differences in both baking and electronics. The remaining challenge is identifying useful candidates and assigning credit when records interact, not preserving or reconstructing their content.

Repeated use across tasks. The planted-store streams show reduced harmful exposure persisting into later rounds (Figure 4b); the shopping comparison tests grounding across complete bundles in all five categories.

F GROUNDING: COVERAGE AND BREADTH

The binder reads only the observable pre-action query. For every completed step it parses the benchmark-provided feedback record, resolves exactly one prior ground-truth product, and binds the current requirement, avoid-list, and preference to that product; it refuses missing, malformed, or ambiguous histories and does not read the current target, task file, product catalog, reward, or post-action trace. The procedure text is fixed and each rendered binding carries its source spans and a digest. An earlier binder that instantiated the rule from the agent's own previous purchase was wrong whenever that purchase missed its target, because the benchmark's next step follows the fixed

ground-truth chain; its structural gate admitted the procedure in every bundle and could not identify the failures, which motivated the corrected operation.

We first evaluate the operation without retrieval or learning: each bundle receives either no injected content or the grounded procedure at every applicable step. Table 27 covers 116 complete Qwen3-32B bundles under the persistence protocol. Averaging the five categories equally gives a reward gain of $+0.045$ (descriptive 95% interval $[+0.023, +0.066]$, resampling bundles within category), with 79 of 116 bundles improving, three tied, and 34 declining. Grocery has the clearest category-level gain; the other category intervals include zero. Exact-target success rises in grocery and beauty, stays level in baking, and falls in electronics, where the procedure improves attribute satisfaction without locating the exact target. In the matched per-bundle run (Table 16) the binder applies on 709 of 900 steps and abstains on the first step of each bundle and on 41 records it cannot parse; the card's gain over no injection is $+0.038$ $[+0.018, +0.057]$ on bound steps and -0.023 $[-0.060, +0.011]$ on abstained steps.

Table 27: Breadth of the fixed grounding operation with a Qwen3-32B reader. Each arm runs complete six-step bundles from fresh state. Intervals are descriptive bundle-bootstrap 95% intervals; $+/=/-$ counts bundle-level differences. Exact is the number of exact-target steps under no injection and the grounded procedure.

Category	n	None	Grounded	Difference	Interval	$+/=/-$	Exact
Grocery	30	0.156	0.247	$+0.092$	$[+0.043, +0.138]$	24/0/6	$10 \to 17$
Electronics	16	0.216	0.262	$+0.046$	$[-0.006, +0.103]$	10/0/6	$3 \to 0$
Baking	30	0.169	0.209	$+0.039$	$[-0.001, +0.078]$	20/1/9	$4 \to 4$
Beauty	30	0.181	0.220	$+0.039$	$[-0.006, +0.085]$	20/2/8	$11 \to 14$
Home	10	0.139	0.145	$+0.006$	$[-0.045, +0.063]$	5/0/5	$0 \to 0$

The grocery result replicates with GPT-4.1-mini on the same 30 bundles: arm means are 0.226 with no injection, 0.219 with a generic reminder, and 0.301 with the grounded procedure; the grounded procedure improves over no injection by $+0.074$ ($[+0.018, +0.128]$, 19/1/10 bundles) and over the generic reminder by $+0.082$ ($[+0.033, +0.128]$, 17/3/10), and the reminder differs from no injection by -0.007. Exact-target counts are 20, 21, and 26 of 180 steps. Forking two home decisions from identical saved states, grounded minus no injection is $+0.119$ at step 6 (window treatments; $[+0.041, +0.220]$, 6/4/0 over ten bundles) and -0.003 at step 3 ($[-0.072, +0.080]$, 2/4/4). Across all these studies the operation is fixed and receives no outcome update, so the results establish task grounding rather than adaptive selection.

G Ablations and diagnostics

Outcome-association controls. To test whether the association between an exposure and its outcome contributes to learning, we replace the current outcome with one sampled from an earlier round while retaining the current recipients and update weights (Table 28, dense-prior planted streams). Mean reward falls from 0.210 to 0.165 for the utility-only learner and from 0.196 to 0.143 for the exploration-enabled learner, close to static (0.160) or below it; paired real-minus-reassociated gains are $+0.044$ $[+0.003, +0.086]$ and $+0.053$ $[+0.019, +0.086]$ (2,000 draws resampling the two paired streams within each of baking, beauty, and grocery), with category means $+0.118$, $+0.088$, and -0.072 for utility and $+0.089$, $+0.056$, and $+0.013$ with exploration. Correct outcome association therefore contributes to the gain, with the exploration-enabled contrast positive in all three categories; this is corrupted earlier-outcome feedback rather than a pure test of memory identity, and it does not establish causal credit at the individual-memory level. Equal credit ($w = 1/|R_t|$ through the same estimator) gives 0.180 and 0.193: the behavioral rule's mean advantages are 0.030 and 0.003 against stream SDs of 0.06–0.09, so these results do not establish an advantage for behavioral credit. Plain-content rows, which remove address facets and scope while retaining the same text, change reward by -0.002 and $+0.015$ under feedback against $+0.049$ and $+0.036$ for CASE; with stream SDs of 0.04–0.09, this small-store test does not establish a representation advantage.

Credit rules. Table 29 compares exposure credit, continuous-reward behavioral credit, the binary helped signal used in the reported runs, and max normalization on the planted store. Binary feedback with the pull-decayed bonus gives 0.209 against 0.173 static, and continuous sum-normalized credit

Table 28: Outcome-association controls on the planted store, dense prior, development categories (six streams; string reward, mean ± SD, paired difference against the real-feedback arm of the same stream). *Reassociated outcomes*: this round's injected memories are credited with the outcome of a random earlier round. *Equal credit*: the same learner and estimator with $w = 1/|R_t|$. *Plain content*: the same texts with no address facets or scope. Static is the no-feedback reference.

Feedback condition	Arm	Reward	vs. real feedback	vs. static
Static (no feedback)	—	0.160 ± 0.034	—	—
Real feedback	+ Utility	0.210 ± 0.090	—	$+0.049$ (4 up, 2 down)
Real feedback	+ Utility + exploration	0.196 ± 0.057	—	$+0.036$ (5 up, 1 down)
Reassociated outcomes	+ Utility	0.165 ± 0.043	-0.044 (2 up, 4 down)	$+0.005$ (3 up, 3 down)
Reassociated outcomes	+ Utility + exploration	0.143 ± 0.049	-0.053 (1 up, 5 down)	-0.017 (2 up, 4 down)
Equal credit	+ Utility	0.180 ± 0.063	-0.030 (2 up, 4 down)	$+0.020$ (3 up, 3 down)
Equal credit	+ Utility + exploration	0.193 ± 0.066	-0.003 (2 up, 2 down, 2 tie)	$+0.033$ (5 up, 1 down)
Plain content, static	—	0.172 ± 0.043	—	$+0.011$ (4 up, 2 down)
Plain content, real feedback	+ Utility	0.170 ± 0.049	-0.039 (2 up, 4 down)	-0.002 (3 up, 3 down; vs plain static)
Plain content, real feedback	+ Utility + exploration	0.187 ± 0.057	-0.009 (2 up, 4 down)	$+0.015$ (4 up, 2 down; vs plain static)

gives 0.166 against 0.171; max normalization accelerates evidence accumulation but changes discovery and early demotion. These development comparisons motivate the declared settings without establishing a superior attribution rule.

Table 29: Credit-rule ablation on the planted store (six runs per arm, 720 subtasks, string / judged reward, exact matches, runs in which the good low-similarity object was discovered). Exposure credit shares the outcome equally. Behavioral variants differ in reward (continuous or binary) and normalization (sum or maximum); all use a pull-decayed exploration bonus. Binary feedback uses $y_t = \mathbb{1}[r_t \geq 0.5 \vee \text{exact}]$. Binary, sum-normalized credit is the configuration in Tables 23 and 24.

Credit rule	Arm	Reward (str.)	Reward (judge)	Exact	Discovers good object
Exposure	Static	0.162	0.231	34	0 / 6
	Flywheel	0.191	0.266	44	2 / 6 (by chance)
	Flywheel + UCB	0.220	0.299	60	5 / 6
Continuous, sum-normalized	Static	0.171	0.232	35	0 / 6
	Flywheel	0.180	0.246	45	1 / 6
	Flywheel + UCB	0.166	0.236	37	2 / 6 (abandoned in 4)
Binary, sum-normalized	Static	0.173	0.246	39	0 / 6
	Flywheel	0.150	0.219	35	0 / 6
	Flywheel + UCB	**0.209**	**0.284**	**63**	5 / 6 (kept in 3)
Binary, max-normalized	Static	0.160	0.228	31	0 / 6
	Flywheel	0.177	0.248	31	0 / 6
	Flywheel + UCB	0.187	0.261	47	3 / 6 (kept in 3)

Uptake audit. Over 840 memory–action pairs from the dense-prior learning streams, an independent judge that never sees the weight labels each injected memory, from the agent's actions, as followed (595), ignored (129), or ambiguous (116). The logged weight has an AUC of 0.36 (cluster bootstrap over seven streams; 95% interval 0.17–0.52) and maximum memory–action cosine an AUC of 0.38 (Table 30); in the 91 decisions where the judge labelled one memory followed and the other ignored, the weight ranks the followed memory higher in only 34. The credit reaching the ledger is nearly constant (0.500 ± 0.01 per memory per round on the sum-normalized planted streams), so the reported online results are object-level outcome learning with approximately shared credit, not fine-grained causal attribution.

Content versus selection. A fixed human-written procedure that instructs the agent to filter options by compatibility, apply the current preference, verify the variant, and buy scores 0.252 ± 0.025 when supplied on every round without retrieval or learning (Table 26, row F). Injecting each of the twelve objects most often selected by the query-matched arms alone on every round shows that the store contains procedures with higher solo estimates than any retrieval configuration: two procedures prescribing precise search queries score 0.249 ± 0.032 and 0.242 ± 0.003 over three streams each, above static (0.226) and dense retrieval (0.220) and level with the hand-written procedure, while ten others lie within one-stream noise of static or below it. Both strong procedures were admitted from failed source episodes; the static arm never injected the first of them and the learners

Table 30: Uptake audit on 840 memory–action pairs (dense-prior learning streams). Labels from an independent judge reading the agent's actions; signals never seen by the judge. AUC is for FOLLOWED vs. IGNORED; 0.5 is chance.

Signal	mean, followed	mean, ignored	AUC
Logged influence weight w (sum-normalized)	0.521	0.631	0.358
Max memory–action embedding cosine	0.684	0.716	0.376
Mean memory–action embedding cosine	0.580	0.618	0.312
Lexical overlap (max over actions)	0.070	0.062	0.580

sampled it without settling on it. Because the solo diagnostic changes both selection and the number of supplied objects, it does not attribute the gap to selection alone.

One injected object (k=1). Repeating the planted dense-prior development streams with a single injected object removes shared credit by construction. The dense prior ranks the good full-search procedure first, so the static arm injects it on every round and scores 0.272 ± 0.056; the utility-only learner scores 0.192 ± 0.093, below static in all six streams, and in both grocery streams becomes stuck on a harmful procedure after early outcomes. The declared configuration scores 0.230 ± 0.033 (-0.042), returning to the good procedure but spending 10–21 rounds per stream on harmful objects. Giving one object the full reward therefore does not improve on a static ranking that already selects the useful procedure; the k=2 development gain of Table 23 arises because the prior's top pair there contains a harmful object that the ledger can demote.

Single-object feedback on a natural store. On the 69-object baking disjoint-source store, nine runs per arm use task-specific queries, a dense prior, a hard top-eight candidate set, k=1, unit exposure credit, and no exploration. Static retrieval averages 0.2184 ± 0.0160 and the binary-feedback learner 0.2288 ± 0.0328, a difference of $+0.010$. Static selects the same object in every run but ranges from 0.197 to 0.255. The learner concentrates on one object for 54–111 of 120 rounds, with the selected object varying across runs. The first three runs, which motivated the extension, had a larger mean difference of $+0.030$; all nine are included in the reported aggregate. On electronics, six matched same-store runs give 0.2220 ± 0.0273 for the learner and 0.2172 ± 0.0128 for static ($+0.005$). A separate three-run study gives 0.2074 and 0.2266 (-0.019). These full-run summaries place the natural-store differences alongside the observed execution variation.

Baseline centering and the no-memory reference. Shrinking toward the same reference that is subtracted gives zero adjustment to an untried object (Eq. 2). The reported online runs estimate this reference from outcomes observed when memory was supplied; it is a relative ranking baseline, not a no-memory counterfactual. A matched no-memory calibration using the learner's binary endpoint gives family references of 0.158 for baking, 0.138 for electronics, 0.104 for home, 0.083 for grocery, and 0.063 for beauty. Replaying six logged fixed-store streams under the pooled reference and this matched reference, holding every candidate score and outcome fixed, changes the selected set on 22 of 720 decisions (3.1%): the reference changes the absolute interpretation of utility more than its ranking, because a common shift largely cancels when candidates have similar evidence counts. Rescoring the planted logs also shows that the pooled reference lets an object that dominates retrieval define the baseline and end near zero utility, and that a leave-one-out reference cannot repair this, since a dominant object has no rounds without it; only a fixed memory-free reference separates the dominant good objects from the harmful ones. This supports a matched no-memory reference when utility is to be read relative to no memory; it does not turn U into a causal treatment effect.

Comparing co-exposed records from identical states. We test natural records under controlled exposure. From 50 declared development bundles, 33 have two distinct retrievable records before branching (two baking, six beauty, eight electronics, eight grocery, and nine home bundles); the other 17 are ineligible. We freeze the first- and second-ranked records (A and B), then restore the same pre-action state at step 6 for four arms, no injection, A, B, and $A+B$, with two repetitions per arm (264 branch outcomes); grounding and ledger updates are disabled. Mean rewards are 0.148 with no injection, 0.129 with A, 0.224 with B, and 0.184 with both; the lower-ranked record wins on 16 bundles, loses on four, and ties on 13 (Table 31). Relevance rank can therefore miss a useful alternative. Adding A to B lowers the point estimate, but its interval includes zero; this study does

not establish a general interference effect or test whether a learned selector can identify the better record. Exact-target successes are 0, 2, 4, and 3 out of 66 branches, respectively.

Table 31: Controlled exposure of natural records: 33 eligible development bundles, two repetitions per arm. Paired string-reward differences with descriptive bundle-bootstrap 95% intervals; counts compare bundle means. A and B are the first- and second-ranked records before intervention.

Contrast	Difference	95% interval	Higher / tied / lower
$A-$ no injection	-0.019	$[-0.069, +0.040]$	6 / 15 / 12
$B-$ no injection	$+0.076$	$[+0.001, +0.166]$	12 / 13 / 8
$(A+B)-A$	$+0.055$	$[-0.018, +0.126]$	12 / 18 / 3
$(A+B)-B$	-0.041	$[-0.092, +0.008]$	6 / 16 / 11

Interference from a second memory. To separate effects of co-injection from the ledger update, we fix the injected content on every subtask: the useful procedure `n39_gotcha` alone, or the same procedure followed by `n34_good` or `n49_good`, with three repetitions of the same 120 baking subtasks and no adaptive selection or episode writing. Mean string reward is 0.249 ± 0.032 alone, 0.216 ± 0.019 with `n34`, and 0.192 ± 0.014 with `n49`; exact matches are 23, 14, and 12 out of 360, and judged means 0.336, 0.300, and 0.286. Adding these partners reduces average reward even with selection fixed, supporting interference in the agent's use of the combined context as a contributor to the two-object results. This diagnostic does not establish that every second memory is harmful; the solo scores of `n34` (0.193) and `n49` (0.202) each come from one repetition.

H A WORKED CASE: OUTCOME, UPDATE, AND LATER REUSE

This development trace follows a Qwen3-32B agent over 120 baking subtasks with a frozen, naturalized store of 120 objects, lexical retrieval, and the shared header query. Grounding is off. The procedure `n43` came from a failed episode; later use changes its ranking while its content stays fixed. Table 24 reports the aggregate result and the corresponding dense-prior comparison.

Content	`PROCEDURE: When purchasing a bundle of technically compatible items, verify that the current item is compatible with previous purchases before clicking 'Buy Now' to ensure the set works together.`
Address	abstraction = first 160 characters (dense key); cue anchors = {`Buy Now`, `PROCEDURE`, `When`} (BM25 + entity channels)
Scope	`baking`; kind = `how-to`; family for U = `baking`
Evidence	writer = GLM-5; source episode = `baking_item_15_step_3`; source reward = 0.40, exact = no; writer label = *gotcha*
Ledger, round 0	$V = 1$; pulls $= 0$; $N = 0$, $S = 0$; $U($`baking`$) = 0.000$ (untried $\Rightarrow$ exactly the family prior)
Ledger, round 7	pulled once, credited weight $w = 1.0$, reward $0.5 \geq \theta$: $N = 1$, $S = 1$; $U = +0.102$
Ledger, round 10	$N = 4$, $S = 3$ (an exact match at round 10); $U = +0.161$
Ledger, round 120	$V = 1$ (no contradiction ever recorded); pulls $= 114$; $N = 114$; $U = +0.006$

Figure 5: A fixed procedure and its evolving ledger. The failed source episode is provenance; the reuse statistics begin at zero and accumulate only after the record is supplied.

Closing the loop. Static retrieval repeatedly supplies `n42` and `n118`, which advise against buying after a single item even though each subtask requires one purchase. The utility arm starts with that pair. After the first failure, their utilities fall to -0.026 and -0.024, allowing other candidates to enter. At round 7, `n43` receives its first exposure, unit credit, and reward 0.5. Its utility rises to $+0.102$. At round 8 it moves to the first slot, and it is supplied on 114 of 120 rounds.

Table 32: The next retrieval reads the updated ledger. Every utility below belongs to `n43` and is measured after the update.

Round	Supplied pair	r_t	$U($`n43`$)$
7	n4, **n43**	0.50	$+0.102$
8	**n43**, n4	0.50	$+0.160$
10	n43, n4	1.00	$+0.161$

By round 120, $U($`n43`$)$ is close to zero because this frequently selected record increasingly defines the pooled family baseline. It remains selected: utility need not keep rising for earlier feedback to affect later reuse. The uptake audit in Section E.4 separately evaluates the credit weights.

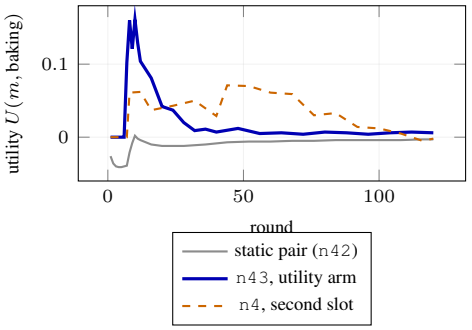


Figure 6: Utility over the same stream. The early increase promotes `n43`; its later decline reflects the moving family reference (Appendix G).