CROSS-TASK EXPERIENCE MEMORY THROUGH SELECTIVE STRATEGY REVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Experience memory lets language-model agents reuse strategies from earlier tasks. As related tasks reveal new conditions, the memory system must decide which parts of a strategy to preserve and which to revise. We introduce Cross-Task Experience Memory (CTEM), which makes this decision using the relationship between new trajectory evidence and a stored strategy. CTEM links compact strategy items to source fragments: confirming evidence is attached without rewriting, while refinements, counterexamples, and unresolved conditions guide selective updates. The resulting strategy fields provide concise guidance for later tasks, with source context retained for future revisions. On WebArena, CTEM improves macro accuracy over ReasoningBank by 2.1–7.7 percentage points across three backbones; on AppWorld, dense scores improve by 8.4 and 8.8 points. With query enhancement disabled, expanding the update policy from support-only to support+refine and then full relations raises WebArena macro accuracy from 35.1 to 36.3 to 38.2. These results support selective revision as a useful design for maintaining reusable experience across tasks.

1 INTRODUCTION

An agent working through a stream of tasks repeatedly encounters useful procedures for finding objects, checking their state, and carrying out operations. Experience memory lets a fixed language model carry this knowledge into later interactions. Reflections, reusable workflows, and reasoning memories have demonstrated the value of learning from completed trajectories (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2025; Ouyang et al., 2026). As experience accumulates, related tasks also reveal when a familiar procedure needs an additional check or a different action. Maintaining useful memory therefore requires deciding how these observations should change an existing strategy.

A strategy extracted from one trajectory may leave its broader applicability underspecified. Storing related experiences as separate items leaves their differences to be reconciled during retrieval and action. Consolidating them into a shared strategy provides a place to express what carries across tasks and where the procedure needs qualification. This requires more than finding a semantically similar memory: the system must assess how the new evidence bears on that memory. Repeated confirmation can strengthen its evidence base while leaving its guidance intact; a newly observed condition may require a targeted revision. Keeping the source context available lets later updates consult the observations behind the earlier abstraction.

Figure 1 contrasts separately stored task memories with shared strategies maintained across tasks. Related completed tasks can contribute to the same memory, while other experience forms a separate item. This organization provides a place to revise guidance as its conditions become clearer. It motivates our central question: *how can an agent revise a shared strategy as new task conditions emerge, while preserving its useful content?* We study this problem through cross-task memory consolidation, in which completed tasks supply evidence for maintaining guidance used by later tasks.

We introduce **Cross-Task Experience Memory (CTEM)**, which couples compact shared strategies with the trajectory evidence used to maintain them. Its Memory Library stores actionable guidance, and its Mirror Environment retains linked source context. After a task ends, CTEM determines how

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

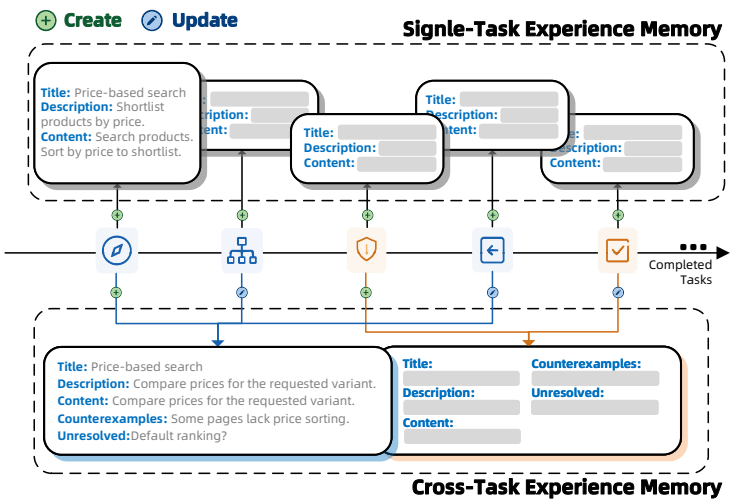


Figure 1: Separate task memories and cross-task shared strategies. In this illustration, completed tasks create new memories or update related ones.

new evidence relates to a candidate strategy and selects the corresponding update. Confirmation attaches evidence while preserving the text; other relations guide revisions to conditions, exceptions, or unresolved questions. The writer consults both the new evidence and relevant historical fragments, while the acting agent receives the compact strategy fields. Historical situation descriptions also support an auxiliary query-enhancement stage for retrieving memory before later tasks.

We evaluate CTEM on online task streams from WebArena and AppWorld (Zhou et al., 2024; Trivedi et al., 2024), using only earlier completed tasks to build memory and reserving benchmark evaluator labels for scoring. Both full CTEM and CTEM-direct, which disables only query enhancement, outperform ReasoningBank on all five reported benchmark–backbone aggregates. The relation-subset ablation examines the update policy with direct retrieval held fixed: WebArena macro accuracy rises from 35.1 with support-only to 36.3 with refinement and 38.2 with the full policy. These comparisons connect the proposed update design to task performance. Analyses of memory formation, subsequent retrieval, and token use further show how shared strategies accumulate evidence and what their maintenance costs.

Our contributions are:

- CTEM, a framework that uses the relationship between new evidence and stored guidance to maintain shared strategies, linking compact action guidance to source context for selective revision.
- Evidence of improved task performance across browser and API interaction: both full CTEM and its direct-retrieval variant exceed ReasoningBank on all five reported benchmark–backbone aggregates.
- An analysis showing higher accuracy as the update policy expands from confirmation to refinement and full relations, alongside broader item retrieval coverage and a modest total-token increase over ReasoningBank in the measured settings.

2 Cross-Task Experience Memory

CTEM maintains shared strategies using evidence accumulated over a task stream. Figure 2 shows its three operations: retrieve memories before acting, preserve evidence after execution, and selectively update related strategies. The actor needs concise guidance for its next action; an updater needs the context behind a rule to decide whether new experience changes it. CTEM therefore separates the representation read during action from the evidence consulted during maintenance.

We consider a stream of tasks $q_1, \ldots, q_N$ solved by an LLM agent with fixed parameters. Before task q_i, the agent can access only memory constructed from earlier tasks. It receives an initial observation $o_{i,0}$, interacts with the environment, and produces a trajectory H_i. Memory changes occur after the trajectory ends. Official evaluator labels are used for reporting task outcomes and are excluded from memory construction.

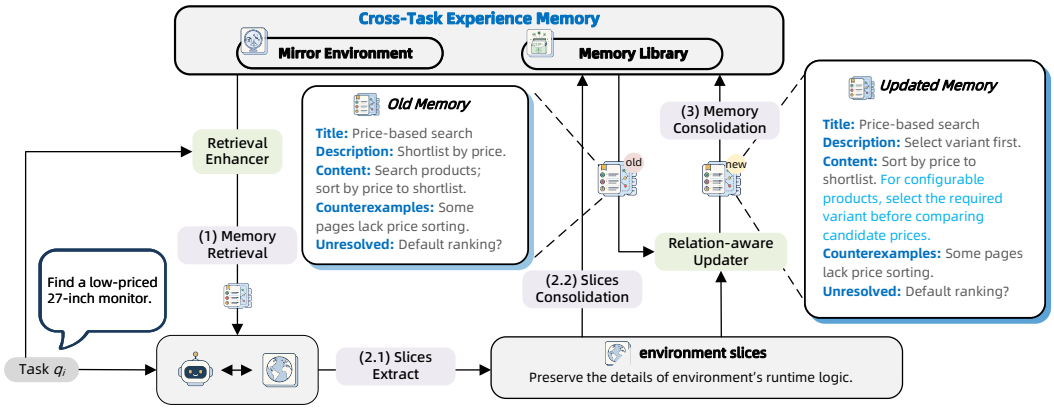


Figure 2: CTEM overview. Memories are retrieved before a task and updated using trajectory evidence afterward. The monitor example illustrates strategy refinement.

2.1 Preserving Evidence Behind Shared Strategies

CTEM separates the evidence used to revise a strategy from the strategy supplied to the acting agent. The *Mirror Environment* stores evidence from completed trajectories. The *Memory Library* stores compact memory items that accumulate related evidence across tasks. Keeping these objects linked allows a later update to consult the source context behind an earlier abstraction.

After each task, an internal LLM judge assigns a success or failure signal using the task, trajectory, final observation, final answer, and observed errors. This signal guides extraction. Successful trajectories provide candidate strategies; failed trajectories provide avoidable mistakes, missing checks, or unresolved conditions. The judge is conservative about exhaustive lists and no-match answers: insufficient evidence of coverage or absence is treated as failure. This internal judgment is a construction signal, and does not replace the benchmark evaluator.

The extractor produces at most three evidence anchors from a trajectory. Each anchor $z_{i,r}$ contains a title, a reusable statement, its task and environment context, and a boundary hint. It also identifies a contiguous source span, yielding a trajectory fragment $T_{i,r}$. A separate generation step constructs a retrieval anchor $u_{i,r}$ describing when the experience would be relevant to a future task. An environment slice links these objects:

$$E_{i,r} = (T_{i,r}, z_{i,r}, u_{i,r}). \tag{1}$$

The evidence anchor supports matching and revision; the retrieval anchor supports lookup. The latter contains a recall query, environment cues, and a descriptive retrieval document.

Each memory item contains a title, description, main content, counterexamples, and unresolved conditions. The description identifies the applicable situation, while the content states the strategy. The item also links to its accumulated environment slices, grouped by their relation to the strategy. These links preserve the evidence available for subsequent updates without inserting entire trajectories into the action prompt.

2.2 Selective Updates Across Tasks

For each new slice, CTEM embeds its evidence anchor and retrieves the most similar candidate memory. The relation classifier compares the anchor and source context with that candidate. It decides whether the new experience supports, refines, contradicts, or leaves uncertain the existing

Table 1: CTEM selects the memory operation according to the relationship between new evidence and the existing strategy.

Relation	Memory operation
Support	Attach evidence; leave memory text unchanged.
Refine	Add a supported condition, variation, or boundary.
Contradict	Record a concrete exception or failure case.
Uncertain	Preserve an unresolved condition.
No match	Create a separate memory from the new slice.

strategy. A no-match decision creates a separate memory. Embedding retrieval proposes a candidate; the classifier checks whether the new evidence concerns the same reusable experience and what change it supports. If the candidate is below the update similarity threshold or is classified as unrelated, the new slice initializes its own memory item.

Table 1 summarizes the update policy. Support attaches the new slice without rewriting any memory text. Repeated confirmation thus adds evidence without another paraphrase of the strategy. Refine updates the description or content when the evidence introduces an additional condition, variation, or applicability boundary. The classification prompt explicitly prefers refine over support when such information is present.

Contradict adds a concrete exception or failure case to the counterexamples field. It revises the main strategy only when the conflict changes the aggregate rule. Uncertain stores the missing evidence or unresolved boundary without treating it as an established strategy. For a rewriting relation ρ, the writer receives the old memory m_j, the new slice, and a relation-dependent selection $\mathcal{E}_j^\rho$ of its accumulated evidence:

$$m_j \leftarrow \mathrm{Update}_\rho(m_j, E_{i,r}, \mathcal{E}_j^\rho), \qquad \rho \in \{\mathrm{refine}, \mathrm{contradict}, \mathrm{uncertain}\}. \tag{2}$$

Refinement can consult all four evidence groups; contradiction and uncertainty use their corresponding groups. A bounded evidence pack supplies related anchors and source fragments (Appendix F). The writer is instructed to preserve useful old content and restrict additions to the supplied evidence. Memory embeddings are recomputed after creation or rewriting. The resulting item expresses the shared strategy together with the conditions, exceptions, and unresolved questions recorded through subsequent updates.

In the illustrative example in Figure 2, refinement preserves price-based shortlisting and adds a check of the requested variant's price to the guidance for subsequent tasks.

2.3 Retrieving and Applying Memory

Before each task, CTEM uses the task query and initial observation to retrieve relevant retrieval anchors from the Mirror Environment. It concatenates their recall queries, environment cues, and retrieval documents with the current query and observation. This deterministic rendering forms an enhanced query for embedding search over the Memory Library. The default configuration retrieves $k_h = 3$ anchors and then $k = 3$ memory items. CTEM without retrieval enhancement skips the anchor stage and searches the same memory library directly using the task and initial observation.

The memory renderer constructs the action prompt from the non-empty title, description, content, counterexamples, and unresolved fields of selected memories, within the shared prompt budget. Source trajectories, evidence anchors, and retrieval anchors remain available to the memory system but are excluded from this prompt. The memory preamble instructs the agent to check each strategy against the current task and observation, and to verify carefully when a listed exception or unresolved condition applies. Applicability is therefore assessed by the acting agent using the retrieved guidance. After execution, the new trajectory enters the same extraction and update procedure, allowing subsequent tasks to use the revised memory library. Appendix G gives the online procedure, and Appendix H records the prompt templates.

Table 2: WebArena accuracy (%), averaged over three fixed-order runs; Macro averages the four domains. Parentheses show point changes from no memory; underlining marks the best baseline. CTEM-direct disables only query enhancement. Domain-level mean ± SD for ReasoningBank and the CTEM variants is in Table 5.

Method	Shopping (187)	Admin (182)	Reddit (106)	GitLab (180)	Macro
			GPT-5.4-mini		
No memory	24.1 (+0.0)	38.5 (+0.0)	29.2 (+0.0)	19.4 (+0.0)	27.8 (+0.0)
AWM	29.4 (+5.3)	43.4 (+4.9)	<u>35.8</u> (+6.6)	22.2 (+2.8)	32.7 (+4.9)
ReasoningBank	<u>38.0</u> (+13.9)	44.5 (+6.0)	33.0 (+3.8)	<u>26.7</u> (+7.3)	<u>35.5</u> (+7.7)
A-MEM	32.6 (+8.5)	<u>45.1</u> (+6.6)	34.9 (+5.7)	25.6 (+6.2)	34.5 (+6.7)
CTEM	**42.2** (+18.1)	**48.9** (+10.4)	**36.8** (+7.6)	**31.1** (+11.7)	**39.8** (+12.0)
CTEM-direct	41.2 (+17.1)	46.7 (+8.2)	35.8 (+6.6)	28.9 (+9.5)	38.2 (+10.4)
			GPT-5.4		
No memory	34.8 (+0.0)	48.4 (+0.0)	39.6 (+0.0)	31.1 (+0.0)	38.5 (+0.0)
AWM	41.2 (+6.4)	53.8 (+5.4)	46.2 (+6.6)	35.6 (+4.5)	44.2 (+5.7)
ReasoningBank	<u>48.7</u> (+13.9)	<u>59.3</u> (+10.9)	<u>47.2</u> (+7.6)	<u>41.1</u> (+10.0)	<u>49.1</u> (+10.6)
A-MEM	44.4 (+9.6)	56.6 (+8.2)	<u>47.2</u> (+7.6)	38.9 (+7.8)	46.8 (+8.3)
CTEM	**55.6** (+20.8)	**64.8** (+16.4)	**58.5** (+18.9)	**48.3** (+17.2)	**56.8** (+18.3)
CTEM-direct	53.5 (+18.7)	63.7 (+15.3)	54.7 (+15.1)	47.8 (+16.7)	54.9 (+16.4)
			Qwen3-32B		
No memory	20.9 (+0.0)	34.1 (+0.0)	25.5 (+0.0)	16.7 (+0.0)	24.3 (+0.0)
AWM	24.1 (+3.2)	37.4 (+3.3)	27.4 (+1.9)	18.9 (+2.2)	26.9 (+2.6)
ReasoningBank	<u>26.7</u> (+5.8)	<u>38.5</u> (+4.4)	<u>28.3</u> (+2.8)	<u>21.7</u> (+5.0)	<u>28.8</u> (+4.5)
CTEM	**27.3** (+6.4)	**40.7** (+6.6)	**33.0** (+7.5)	**22.8** (+6.1)	**30.9** (+6.6)
CTEM-direct	26.2 (+5.3)	40.1 (+6.0)	30.2 (+4.7)	22.2 (+5.5)	29.7 (+5.4)

3 EXPERIMENTS

We evaluate how CTEM performs over online task streams, then examine the update and retrieval choices underlying the system. Token accounting and memory-bank analyses characterize the cost and organization of the accumulated experience.

3.1 EXPERIMENTAL SETUP

Tasks and metrics. WebArena covers browser interaction in Shopping (187 tasks), Shopping Admin (182), Reddit (106), and GitLab (180) (Zhou et al., 2024). We report domain accuracy and its unweighted macro average. AppWorld covers programmatic interaction across applications; we use the 168-task `test_normal` split and report dense score scaled by 100 (Trivedi et al., 2024). This metric differs from strict task success. Interaction steps average successful and failed episodes.

Agents and protocol. WebArena uses GPT-5.4-mini, GPT-5.4, and Qwen3-32B; AppWorld uses the two GPT backbones. We compare no memory, AWM (Wang et al., 2025), ReasoningBank (Ouyang et al., 2026), and A-MEM (Xu et al., 2025); A-MEM covers the two GPT WebArena settings. **CTEM-direct** retains CTEM's evidence store and update mechanism, disabling only retrieval-query enhancement. Each method uses memory from preceding tasks and updates it after the current task ends, without official evaluator labels. Methods share the task order, acting prompt, action budget, decoding settings, and final retrieval count $k = 3$, while constructing their own memories. WebArena averages three fixed-order runs with deterministic acting and a 15-step budget; AppWorld averages three evaluations per task with acting temperature 0.7. CTEM construction uses temperature 0.0 and retrieval enhancement uses $k_h = 3$ historical anchors unless specified otherwise.

Table 3: AppWorld `test_normal`: dense score $\times 100$ and average interaction steps, over three evaluations per task. Parentheses show score changes from no memory; underlining marks the best baseline.

	GPT-5.4-mini		GPT-5.4	
Method	*Dense score*	*Steps*	*Dense score*	*Steps*
No memory	55.2 (+0.0)	12.55	63.5 (+0.0)	10.56
AWM	50.6 (-4.6)	10.03	60.9 (-2.6)	8.24
ReasoningBank	<u>57.5</u> (+2.3)	12.22	<u>67.2</u> (+3.7)	10.55
CTEM	**65.9** (+10.7)	11.89	**76.0** (+12.5)	10.50
CTEM-direct	62.5 (+7.3)	12.00	75.0 (+11.5)	10.53

3.2 Task Performance

CTEM improves task accuracy across all three WebArena backbones, attaining the highest mean among the compared methods in each backbone–domain setting (Table 2). Macro gains over ReasoningBank are 4.3, 7.7, and 2.1 percentage points for GPT-5.4-mini, GPT-5.4, and Qwen3-32B, respectively. The improvement spans the four domains, although its magnitude varies: GPT-5.4 Reddit rises from 47.2 to 58.5, while Qwen3-32B Shopping rises from 26.7 to 27.3. Appendix A reports domain-level variation across three runs, with SDs of 0.31–0.94 points for ReasoningBank and the CTEM variants.

The direct-retrieval comparison shows that CTEM's aggregate advantage is present with query enhancement disabled. CTEM-direct exceeds ReasoningBank by 2.7, 5.8, and 0.9 macro points; adding query enhancement yields a further 1.6, 1.9, and 1.2 points. Both variants retain evidence and the full update policy, so this comparison assesses the retrieval procedure. At the domain level, CTEM-direct trails ReasoningBank on Qwen3-32B Shopping, where full CTEM obtains a small gain.

The same aggregate ordering holds for programmatic interaction in AppWorld: both CTEM variants exceed ReasoningBank, and full CTEM performs best (Table 3). Full CTEM reaches dense scores of 65.9 and 76.0, gains of 8.4 and 8.8 points over ReasoningBank. These gains accompany slightly fewer mean interaction steps. CTEM also uses fewer macro-average steps on each WebArena backbone (Table 7), indicating that its higher reported scores accompany shorter average interaction sequences across both benchmarks.

The AWM results illustrate why interaction length must be read together with task performance. AWM uses approximately 20–22% fewer AppWorld steps than no memory, while its dense scores are 4.6 and 2.6 points lower. One possible explanation is negative transfer: a retrieved workflow may fit a local subroutine while omitting dependencies needed by the complete task. This remains an interpretation of the aggregate score–step pattern. The adapter shares CTEM's runner, action budget, evaluator, internal outcome signal, and retrieval count while retaining AWM's workflow representation and induction prompt; it improves over no memory on all three WebArena backbones. The contrasting benchmark outcomes motivate examining when a reusable procedure applies to the current task.

Table 4: Average tokens per task on WebArena with GPT-5.4-mini. Memory update includes outcome judgment, evidence extraction, relation classification, and memory creation/update.

Method	Anchor generation	Action	Memory update	Total
No memory	—	50,847.4	—	50,847.4
AWM	—	53,819.6	6,672.5	60,492.1
ReasoningBank	—	49,306.1	4,658.3	53,964.4
CTEM-direct	—	47,862.9	6,376.8	54,239.7
CTEM	1,486.2	46,984.7	6,489.4	54,960.3

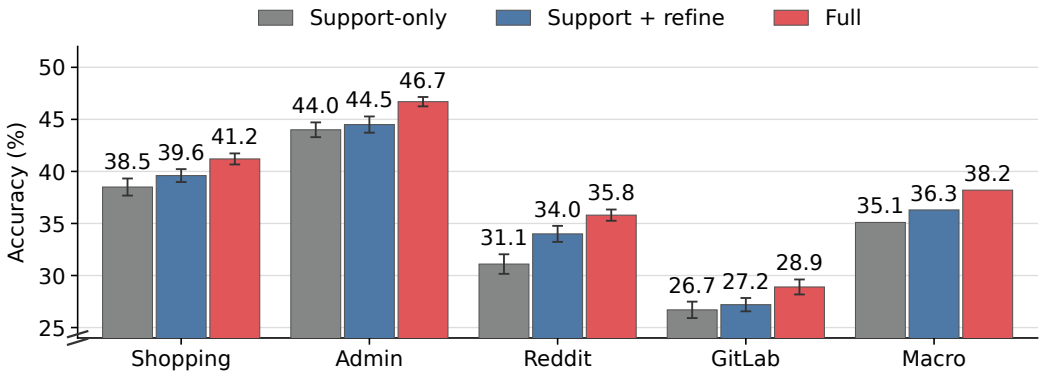


Figure 3: Relation-update ablation on WebArena with GPT-5.4-mini and query enhancement disabled. Bars show mean accuracy; error bars show domain-level SD over three runs. Macro reports means only. Full includes support, refine, contradict, and uncertain.

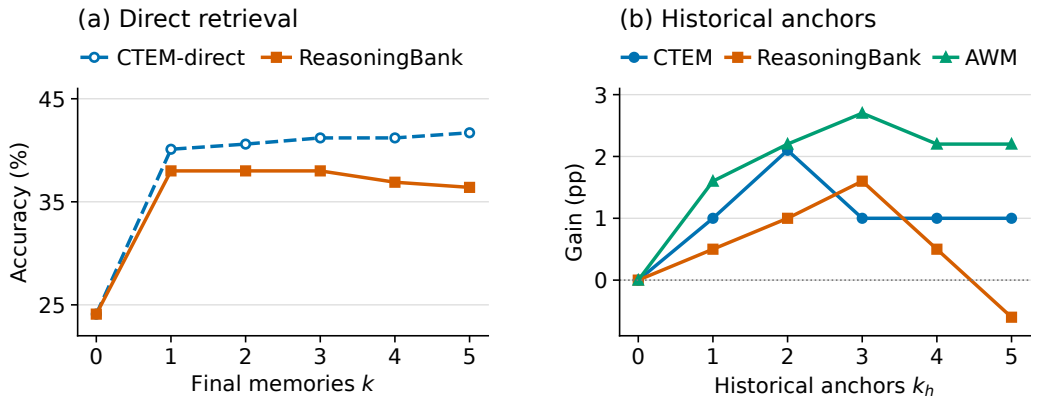


Figure 4: Retrieval sensitivity on WebArena Shopping with GPT-5.4-mini. (a) Accuracy versus final memory count k with direct retrieval. (b) Gain over each method's own $k_h = 0$ baseline when varying historical anchors, with final memory count $k = 3$.

3.3 Token Cost

CTEM spends more tokens constructing memory, with lower action-token use partly offsetting that overhead (Table 4). On WebArena with GPT-5.4-mini, construction requires 7,975.6 tokens per task, including anchor generation, versus ReasoningBank's 4,658.3. Action tokens decrease by 4.7%, giving 54,960.3 total tokens: 1.8% above ReasoningBank and 9.1% below AWM. CTEM-direct is 0.5% above ReasoningBank. Thus, in this measurement, maintaining and retrieving CTEM's shared strategies adds a modest total-token overhead relative to ReasoningBank. The domain coverage of the token average is not documented, so the comparison is limited to the reported aggregate token use.

3.4 Update Ablation

The relation ablation tests whether an updater benefits from using new evidence to qualify an existing strategy. With query enhancement disabled, support-only merges confirming evidence, support+refine also revises applicable conditions, and the full policy additionally retains counterexamples and unresolved conditions. Macro accuracy increases from 35.1 to 36.3 to 38.2 (Figure 3), and full updates perform best in all four domains (Table 6). The progression supports the value of refinement and of the additional relations together within CTEM's update policy.

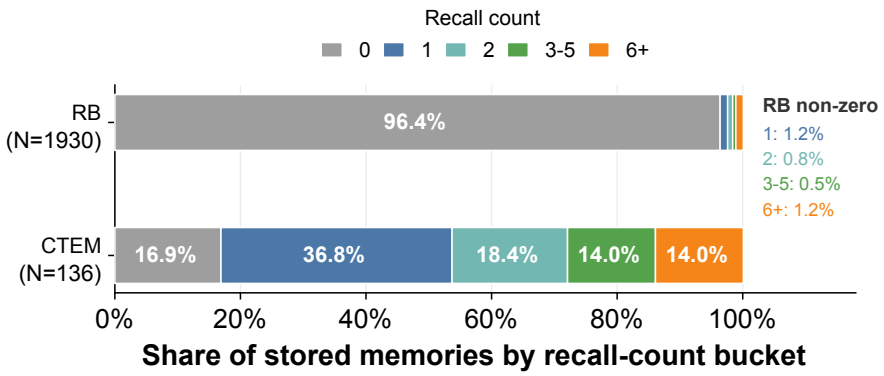


Figure 5: Memory retrieval frequency on WebArena Shopping with GPT-5.4-mini. Bars show the rounded proportion of items in each bucket: 136 CTEM items and 1,930 ReasoningBank items.

3.5 Retrieval Sensitivity

Retrieval budget affects how the resulting memories serve a task (Figure 4). On Shopping, increasing the number of directly retrieved memories from $k = 1$ to $k = 5$ raises CTEM-direct accuracy from 40.1 to 41.7, while ReasoningBank falls from 38.0 to 36.4. The benefit of retrieving more entries therefore depends on the memory system. Historical-anchor enhancement improves all three tested methods at moderate anchor counts, with further increases offering no uniform benefit. These sweeps favor selecting memory and anchor budgets for the particular memory system and task setting. Appendix E provides the full curves and gains over each method's unenhanced baseline.

3.6 Memory Formation and Subsequent Retrieval

The formation analysis shows that shared strategies draw on experience from multiple tasks. On WebArena Shopping with GPT-5.4-mini, 92 memories (67.6%) are linked to multiple tasks. The relation statistics contain 136 strategy items, 62 with refinement-labeled evidence, and 514 relation records: 347 support, 155 refine, 10 contradict, and two uncertain entries. Support and refine account for 97.7% of these records, showing that the observed update activity is dominated by confirmation and revision. Appendix D presents the formation and consolidation statistics.

The retrieval distribution shows how broadly later tasks access the stored entries (Figure 5). Subsequent tasks retrieve 113 of CTEM's 136 items at least once (83.1%), compared with 70 of ReasoningBank's 1,930 items (3.6%); 46.3% of CTEM items are retrieved at least twice. CTEM thus has fewer strategy entries, a higher fraction retrieved, and more distinct entries retrieved in absolute terms. These statistics describe retrieval coverage at the item granularity produced by each memory system and complement the task-level performance comparisons.

4 Related Work

Reusable experience for agents. Experience memory adapts behavior through information carried across interactions while keeping model parameters fixed. Reflexion uses verbal feedback to improve subsequent trials (Shinn et al., 2023), AWM induces reusable workflows in offline and online settings (Wang et al., 2025), and ReasoningBank extracts reasoning strategies from self-judged successes and failures (Ouyang et al., 2026). Contextual Experience Replay maintains a dynamic buffer of environment dynamics and procedural skills, including an online setting evaluated on WebArena (Liu et al., 2025). These systems motivate learning from completed interactions. CTEM addresses how related interactions should maintain a shared strategy as evidence accumulates.

Editing and consolidating memory. Prior work offers several ways to revise memory as interactions accumulate. ExpeL adds, edits and votes on cross-task insights (Zhao et al., 2024); A-Mem

lets a new note trigger changes to the context, keywords and tags of linked memories (Xu et al., 2025). Source context can also remain available alongside compact memories: ReasoningBank retains source trajectories with memory items and reports consolidation by appending items without further pruning (Ouyang et al., 2026), while RecMem revisits raw interactions during consolidation to recover information omitted from summaries (Dai et al., 2026). MSCE organizes traces and evidence-backed policies with triggers, applicability boundaries, verification rules and lifecycle updates (Tang et al., 2026). CTEM connects these concerns through an explicit interface between evidence and strategy revision. The relation of new evidence to an existing strategy determines whether its text is preserved, its conditions are refined, or an exception or unresolved question is recorded. Linked fragments supply context for the selected update, and the resulting compact fields guide the actor on later tasks.

Interactive evaluation. WebArena evaluates browser tasks in realistic websites (Zhou et al., 2024), while AppWorld tests programmatic interaction across applications (Trivedi et al., 2024). Together, they let us examine experience reuse under two forms of interaction, with memory carried from earlier to later tasks in each online stream. Appendix F describes the baseline adaptations and shared evaluation settings used for these comparisons.

5 Conclusion

CTEM maintains reusable experience by deciding how new evidence should change a shared strategy. Linked trajectory fragments provide context for selective revision, while compact strategy fields carry the resulting guidance into later tasks. Both full CTEM and its direct-retrieval variant improve over ReasoningBank on all five reported WebArena and AppWorld benchmark–backbone aggregates. With query enhancement disabled, the relation-subset ablation shows higher accuracy when refinement and then the full update policy are enabled. The memory analyses show cross-task evidence accumulation and broad item retrieval coverage; token accounting identifies the accompanying construction overhead. Together, these findings support an experience-memory design that preserves established guidance while incorporating new conditions and qualified exceptions as tasks accumulate.

Limitations

CTEM depends on LLM judgments for outcome assessment, evidence extraction, relation classification, and strategy revision. Errors in these judgments can propagate into memory despite the availability of source context. The actor also assesses applicability from the retrieved guidance, with correctness depending on its interpretation of the task and recorded conditions. Matching each anchor to one update candidate limits the strategies considered for revision, and retained trajectory evidence continues to increase storage requirements as tasks accumulate.

Our evaluation covers fixed task orders in web and application environments; sensitivity to task order and distribution shifts remains untested. The relation-subset ablation measures the policy within CTEM, leaving the individual effects of contradiction and uncertainty, the contribution of source access, and comparison with a generic updater given identical evidence unresolved. Retrieval counts measure item coverage, with an item's effect on subsequent actions still to be established. The reported token profile covers model-token usage; latency, monetary cost, and total storage require separate measurements.

AI Use Statement

Generative AI tools assisted with manuscript organization and revision, literature search, reference checking, interpretation of reported results, Chinese review translations, and figure and table preparation. The authors are responsible for the scientific content and for verifying all reported results.

REFERENCES

Zijie Dai, Shiyuan Deng, Sheng Guan, Yizhou Tian, Xin Yao, Xiao Yan, and James Cheng. RecMem: Recurrence-based Memory Consolidation for Efficient and Effective Long-Running LLM Agents, 2026. URL https://arxiv.org/abs/2605.16045. arXiv:2605.16045.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual Experience Replay for Self-Improvement of Language Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14179–14198, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.694.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377.

Bo Tang, Yang Zhang, Guomian Zhuang, Wenqiang Wei, Gaoyang Zheng, Lindong Xie, Yanchao Tan, Feiyu Xiong, Qingyu Yang, Edward Chung, and Zhiyu li. From Memory to Skills: Evidence-Grounded Co-Evolution Governance for Long-Horizon LLM Agents, 2026. URL https://arxiv.org/abs/2607.16621. arXiv:2607.16621.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic Memory for LLM Agents. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

A Reported Run-to-Run Variation

Table 5 reports domain-level mean accuracy and standard deviation for ReasoningBank and the two CTEM variants on WebArena. SD summarizes variation across three fixed-order runs. Macro accuracies in Table 2 are means without uncertainty estimates; recovering macro SD would also require the covariance between domains. The domain SDs are descriptive measures of variability and do not establish statistical significance.

Table 5: WebArena domain accuracy (%): mean ± SD across three runs for the main comparison. CTEM-direct retains the full update mechanism and disables only retrieval-query enhancement.

Method	Shopping	Admin	Reddit	GitLab
GPT-5.4-mini				
ReasoningBank	38.0 ± 0.87	44.5 ± 0.84	33.0 ± 0.94	26.7 ± 0.85
CTEM-direct	41.2 ± 0.76	46.7 ± 0.63	35.8 ± 0.77	28.9 ± 0.72
CTEM	42.2 ± 0.53	48.9 ± 0.45	36.8 ± 0.54	31.1 ± 0.32
GPT-5.4				
ReasoningBank	48.7 ± 0.82	59.3 ± 0.84	47.2 ± 0.94	41.1 ± 0.85
CTEM-direct	53.5 ± 0.69	63.7 ± 0.78	54.7 ± 0.77	47.8 ± 0.64
CTEM	55.6 ± 0.31	64.8 ± 0.45	58.5 ± 0.54	48.3 ± 0.45
Qwen3-32B				
ReasoningBank	26.7 ± 0.82	38.5 ± 0.84	28.3 ± 0.94	21.7 ± 0.85
CTEM-direct	26.2 ± 0.62	40.1 ± 0.71	30.2 ± 0.77	22.2 ± 0.79
CTEM	27.3 ± 0.53	40.7 ± 0.45	33.0 ± 0.54	22.8 ± 0.56

Relation-update ablation. Table 6 compares support-only, support+refine, and full relation updates with GPT-5.4-mini and query enhancement disabled. The macro means of 35.1, 36.3, and 38.2 correspond to Figure 3. Full updates improve every domain over either subset: gains are 2.2–4.7 points over support-only and 1.6–2.2 points over support+refine. Each table reports variability within its corresponding evaluation.

Table 6: Relation-update ablation on WebArena with GPT-5.4-mini and query enhancement disabled. Domain results are mean ± SD over three runs; Macro contains the reported mean only. Full enables support, refine, contradict, and uncertain operations.

Update policy	Shopping	Admin	Reddit	GitLab	Macro
Support-only	38.5 ± 0.82	44.0 ± 0.71	31.1 ± 0.94	26.7 ± 0.79	35.1
Support + refine	39.6 ± 0.62	44.5 ± 0.78	34.0 ± 0.77	27.2 ± 0.64	36.3
Full	41.2 ± 0.53	46.7 ± 0.45	35.8 ± 0.54	28.9 ± 0.72	38.2

B Additional Interaction and Token Results

Table 7 gives the domain-level WebArena interaction counts corresponding to Table 2, averaging all episodes, including failures. These counts exclude post-task judging, extraction, anchor generation, and memory writing.

Table 7: Mean WebArena interaction steps. Macro is the unweighted mean across the four domains.

Method	Shopping	Admin	Reddit	GitLab	Macro
GPT-5.4-mini					
No memory	10.0	10.7	13.7	11.1	11.4
AWM	9.3	9.8	12.5	12.5	11.0
ReasoningBank	11.3	12.0	12.6	13.8	12.4
A-MEM	9.9	10.6	12.2	12.0	11.2
CTEM-direct	9.5	10.5	12.3	10.9	10.8
CTEM	9.4	10.4	12.1	10.4	10.6
GPT-5.4					
No memory	7.7	8.3	10.8	8.9	8.9
AWM	7.9	8.0	10.2	9.3	8.9
ReasoningBank	8.0	8.6	10.3	9.7	9.2
A-MEM	8.5	8.4	10.7	9.5	9.3
CTEM-direct	7.2	7.7	10.4	8.4	8.4
CTEM	7.1	7.5	10.3	8.3	8.3
Qwen3-32B					
No memory	11.2	11.6	14.3	12.2	12.3
AWM	10.7	10.9	13.5	12.8	12.0
ReasoningBank	12.1	12.7	13.6	14.1	13.1
CTEM-direct	11.1	12.1	13.2	12.9	12.3
CTEM	11.0	12.2	13.2	12.0	12.1

Token accounting. Figure 6 records action and memory-construction tokens on WebArena with GPT-5.4-mini. The numerical breakdown is given in Table 4. Memory construction includes the internal judge, extraction, classification and writing, plus retrieval-anchor generation. The domain coverage of the token average is not documented; the profile is not paired with domain-level accuracy.

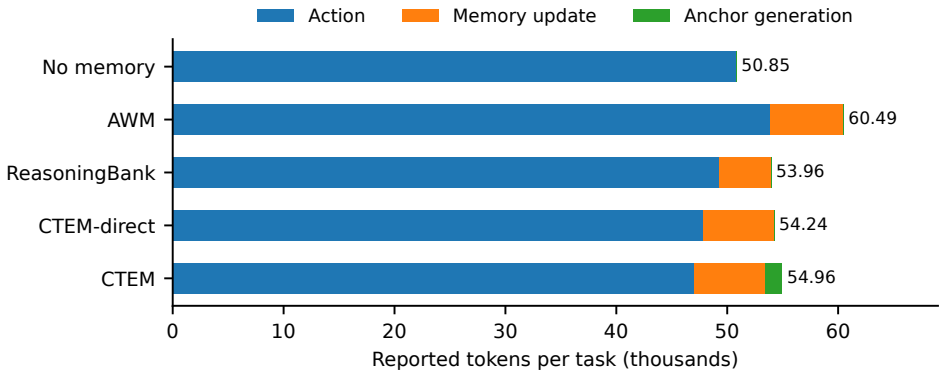


Figure 6: Recorded token usage on WebArena with GPT-5.4-mini, split into action, memory update, and anchor generation. Token categories follow Table 4.

C Evidence Construction and Update Details

The main architecture separates reading strategies from updating them. Figure 7(a) shows evidence construction: an evidence anchor remains linked to a source span, while a retrieval anchor describes when that experience may be useful. Panel (b) illustrates candidate matching, relation classification, and the refinement path. Support attaches evidence without rewriting; no match creates a separate item, as summarized in Table 1.

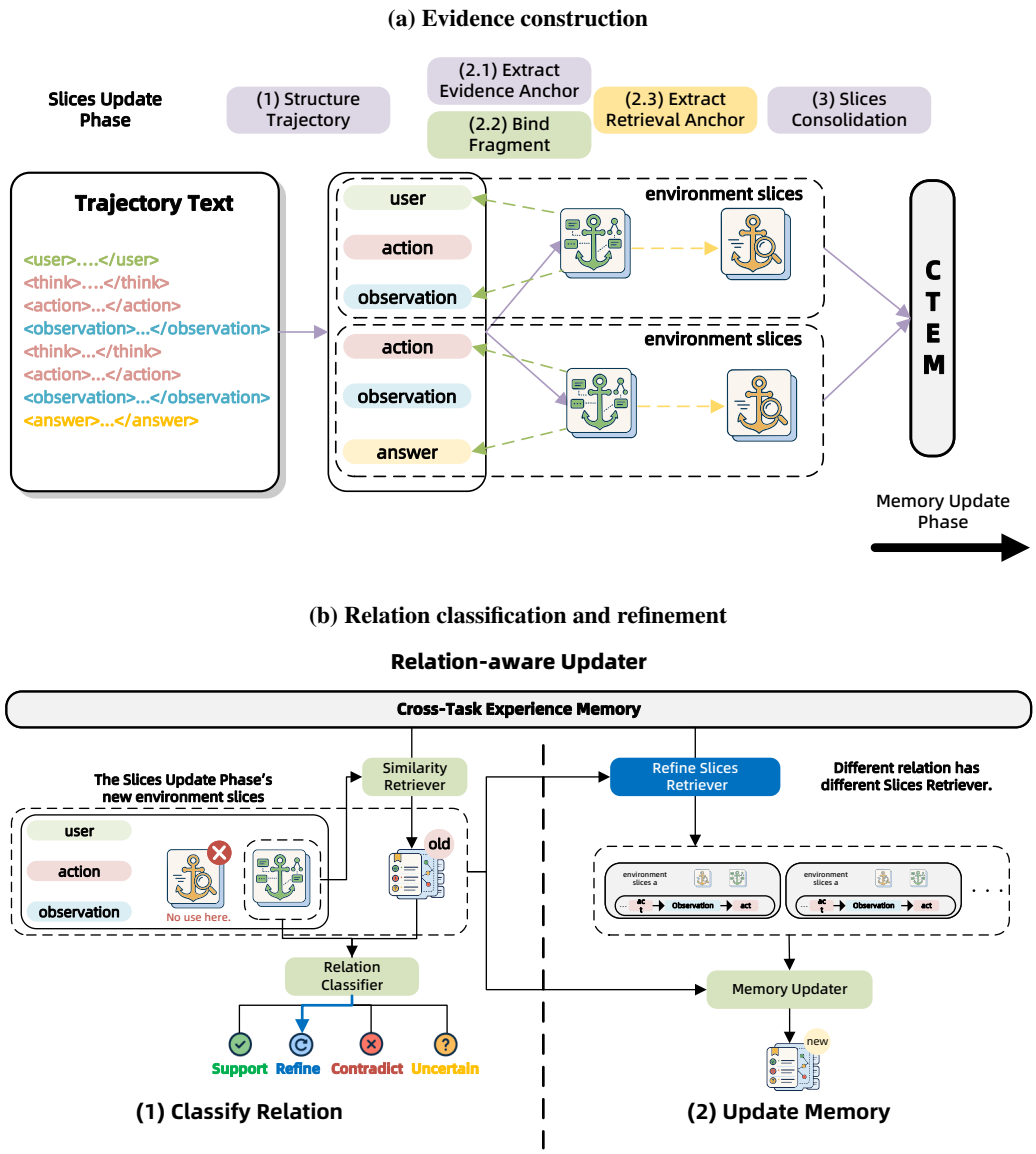


Figure 7: Evidence construction and memory refinement. (a) Source fragments, evidence anchors, and retrieval anchors form linked environment slices. (b) The illustrated refine branch retrieves historical slices to update a matched memory.

The writer receives the existing strategy, the new evidence, and relevant historical slices. Source references remain available after the update. The actor-facing renderer includes only non-empty strategy fields within its prompt budget; source fragments and anchors remain in the memory system.

D MEMORY FORMATION STATISTICS

Relation counts characterize update activity, and consolidation statistics describe aggregation across source tasks; neither is a direct measure of the usefulness of a strategy during action.

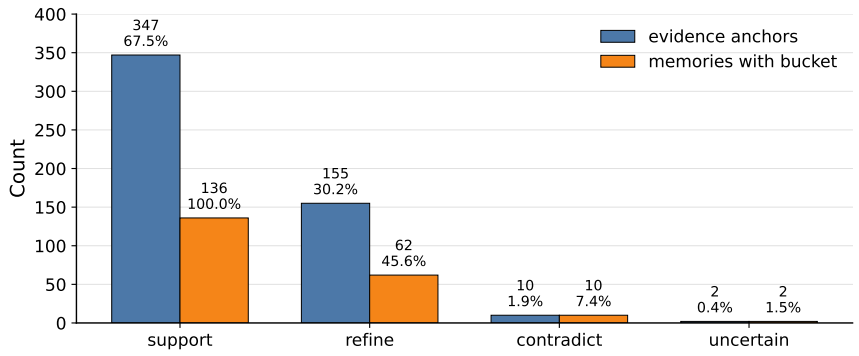


Figure 8: Relation counts for evidence anchors and memory items. Percentages use the corresponding totals of 514 anchors and 136 items; relation buckets can overlap within an item.

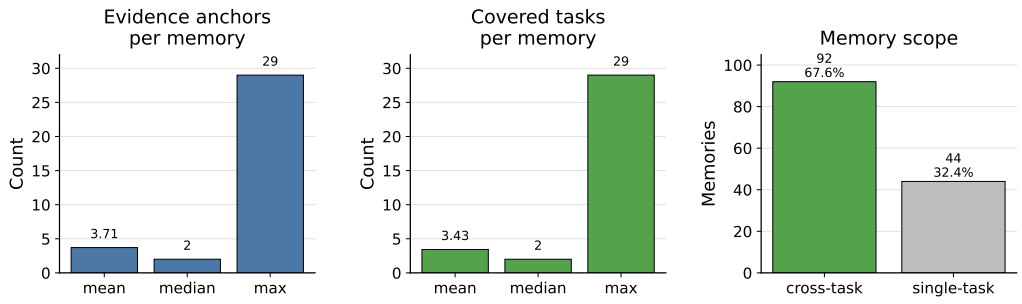


Figure 9: Cross-task consolidation. The first two panels summarize evidence anchors and covered source tasks per memory; the third shows the cross-task/single-task split.

E Retrieval Controls

Figure 10 expands the sweeps in Figure 4, giving each method's historical-anchor curve a separate panel to show how its gains vary with the retrieval budget.

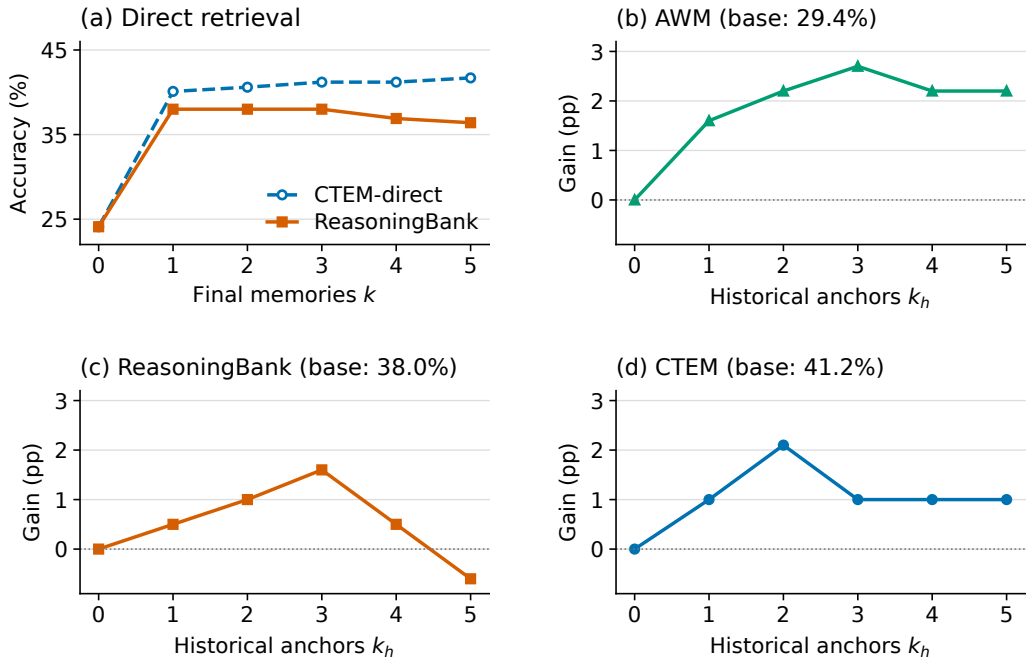


Figure 10: Expanded retrieval sweeps. Panel (a) varies final memory count with direct retrieval. Panels (b–d) separate the three methods in Figure 4(b), using each method's own $k_h = 0$ reference and three final memories.

On WebArena Shopping with GPT-5.4-mini, direct retrieval gives a shared 24.1% no-memory baseline. ReasoningBank reaches 38.0% at $k = 1$ and remains there through $k = 3$, before decreasing to 36.4% at $k = 5$. CTEM-direct increases from 40.1% at $k = 1$ to 41.7% at $k = 5$. These observations motivate examining redundancy and applicability, which the sweep does not directly measure.

The historical-anchor sweep applies enhancement to AWM, ReasoningBank, and CTEM. At the default $k_h = 3$, CTEM reaches 42.2% accuracy, a gain of 1.0 percentage point over its $k_h = 0$ baseline of 41.2%. The highest observed CTEM accuracy in this sweep is 43.3% at $k_h = 2$; the default is not the best setting for this domain.

F Experimental and Implementation Details

Task streams and reporting. WebArena contains 655 tasks in the four evaluated domains: Shopping (187), Shopping Admin (182), Reddit (106), and GitLab (180). Each condition is evaluated in three runs using the same ascending task-ID order. The environment is reset through the benchmark adapter before each task instance, while the memory store persists across tasks within the stream. AppWorld uses 168 tasks from `test_normal`, with three evaluations per task. Its reported score is the dense evaluation score multiplied by 100, rather than strict task success. Interaction steps average successful and failed episodes. Table 8 records the specified settings; the uncertainty tables in Appendix A retain the domain-level statistics from each reported evaluation.

Baseline adaptations. All methods use the same online interfaces: retrieval before a task and memory update after termination. They share the actor, task order, action budget, decoding settings,

Table 8: Experimental and memory-pipeline settings. Core defaults are distinguished from the paper-facing retrieval setting.

Setting	Value or procedure
WebArena acting temperature / budget	0.0 / 15 environment steps
WebArena maximum output tokens	4096
AppWorld acting temperature	0.7
Construction temperature	0.0 (judge, extract, generate anchors, classify, write)
Final memories k / historical anchors k_h	3 / 3 in the default configuration
Evidence anchors per trajectory	At most 3
Update candidates per anchor	1
Update similarity threshold	0.62 (documented core default)
Query-time retrieval threshold	0.0 in the paper-facing adapter; 0.30 core default
Related evidence pack	Up to 100 anchors and 5 source fragments
Malformed judge response	Retry once, then assign failure
Memory rendering	Shared prompt budget; keep retrieval order

evaluator and retrieval count. Each method constructs memory from the trajectories produced by its own actor runs, using the common internal outcome-judgment procedure; official evaluator labels are not supplied to the writer. AWM retains workflow items and its induction prompt, ReasoningBank retains reasoning-rule items, and A-Mem retains its note representation. The shared runner uses a JSONL-backed store and common top-k wrapper where an upstream retrieval component is incompatible; A-Mem uses its adapter's note retrieval. These are evaluations of the adapted memory methods in the common runner. A-Mem is included for the two GPT WebArena settings where its adapter is compatible. Historical-anchor enhancement of AWM or ReasoningBank changes the retrieval query without changing those baselines' memory writers.

Rendering and evidence access. The actor receives non-empty title, description, content, counterexamples and unresolved fields. If the memory block exceeds the shared prompt budget, items remain in retrieval order and fields are shortened from least to most essential: unresolved conditions, counterexamples, content, description, then title. Source fragments and anchors are retained for construction and retrieval, but are not inserted into the actor memory block. Refinement can draw on support, refine, contradict and uncertain evidence; the contradict and uncertain writers use their respective evidence groups. The evidence pack limits bound the context supplied to an update, rather than the total size of the persistent source store.

Construction operations. For a completed task, CTEM judges the outcome, extracts up to three anchors, and generates one retrieval anchor per extracted anchor. Each anchor is compared with at most one candidate. Support adds evidence without a writer call; no-match creates an item, and refine, contradict or uncertain invoke the corresponding writer. Thus the nominal pipeline contains one judgment, one extraction, at most three anchor generations, at most three classifications and at most three creations or revisions; malformed-response retries can add calls. These operations occur after termination and are included in the construction-token accounting, rather than environment interaction steps.

Configuration detail. Exact replication is limited by the unavailable identifiers for the embedding backend and memory-pipeline LLM, the numerical memory-block token limit, and the AppWorld memory reset policy between repeated evaluations.

G Online Memory Algorithm

The following procedure summarizes the documented update and retrieval interfaces. Each environment slice contains a contiguous trajectory fragment, an evidence anchor and a retrieval anchor. The relation-conditioned evidence pack follows the policy described in Section 2.2.

1. Before task q_i, retrieve k_h historical anchors using q_i and the initial observation. Render their recall queries, environment cues and retrieval documents with the current query. CTEM-direct bypasses this enhancement.

2. Retrieve k memory items, render their non-empty strategy fields within the shared budget, and execute the task with fixed model parameters.
3. After termination, normalize and retain the trajectory. Judge the outcome using observed interaction evidence, excluding official evaluator labels, then extract at most three evidence anchors with source-span pointers.
4. For each anchor, recover its source fragment, generate its retrieval anchor, and store the resulting environment slice.
5. Retrieve the most similar candidate memory. Create a new item if none meets the threshold or the classifier judges the candidate unrelated. Otherwise classify the matched relation as support, refine, contradict or uncertain.
6. Add the slice to the selected relation group. For support, retain the memory text unchanged. For the other relations, assemble the corresponding evidence pack and invoke its writer while preserving useful old content.
7. Recompute the embedding after creation or rewriting, save the memory, and advance to the next task.

H MEMORY-PIPELINE PROMPT INSTRUCTIONS

The following system instructions specify the judgment, extraction and update behavior. Variable task, memory and trajectory fields are supplied to the corresponding calls. We reproduce the system instructions from the reported templates; repeated input-field scaffolding is omitted. The three update writers use the same five-field JSON output schema shown for initial creation in Section H.5. Retrieval-query enhancement uses deterministic rendering rather than an additional query-writing LLM call.

H.1 TRAJECTORY JUDGMENT

```
You are a strict judge that decides whether the agent successfully
    completed the user task.

Judge the final result from the user task, the trajectory observations,
     and the agent's final answer.

Important rules:
1. First infer the task's required answer cardinality: a single answer,
     a set/list of answers, an exhaustive list, or a no-match/absence
    answer.
2. A trajectory is success only if the final answer or final action
    fully satisfies the user task.
3. A trajectory is failure if the final answer is partial, premature,
    unsupported, or answers the wrong target.
4. For set/list/extraction tasks, success requires a complete enough
    search for all requested items, not just one matching item.
5. If visible counts, pagination, scrollable regions, or unresolved
    candidates suggest unchecked evidence, mark failure.
6. Do not treat the final answer text itself as evidence; supporting
    evidence must appear in observations.
7. If the trajectory does not provide enough evidence to verify
    completion, mark failure.
8. Ignore execution metadata such as reward, score, passed, ok,
    terminated, or truncated.

Format your response as exactly one lowercase token:
success or failure
```

H.2 EVIDENCE-ANCHOR EXTRACTION

```
You construct the Mirror Environment structured evidence space from one
     completed agent trajectory.

Extract evidence anchors: compact, reusable evidence units grounded in
    the trajectory. Each anchor should represent one distinct hypothesis
     about what future agents can reuse or avoid.

Important rules:
1. Extract at most {max_anchors} anchors.
2. Anchors must be semantically diverse.
3. Each anchor must be supported by one contiguous span of trajectory
    steps.
4. The anchor statement should describe a reusable rule, not a task-
    specific summary.
5. The context description should state the task/environment context
    where the experience appeared.
6. The boundary hint should state source span, applicability limits,
    missing checks, exceptions, or failure conditions visible in the
    trajectory.
7. Do not mention hidden official evaluator labels.
8. If the internal judgment is failure, do not phrase the failed
    trajectory as a successful strategy. Extract what should be avoided,
     what verification was missing, what assumption failed, or what
    boundary condition remains unresolved.

Return valid JSON:
{
  "anchors": [
    {
      "title": "string",
      "statement": "string",
      "context_description": "string",
      "boundary_hint": "string",
      "start_step": 0,
      "end_step": 0
    }
  ]
}
```

H.3 RETRIEVAL-ANCHOR GENERATION

```
You generate one retrieval anchor for memory retrieval enhancement.

A retrieval anchor describes when a future task should recall the given
     evidence anchor. It should connect the reusable evidence anchor,
    the source task query, local environment/context cues, and the
    boundary or failure mode that makes the anchor relevant.

Write the retrieval anchor as descriptive text, not an instruction to
    the agent. Do not include instructions to the acting agent; this
    text is used only for retrieval. Avoid overfitting to exact names,
    IDs, or one-off strings unless they define the task type.

Return valid JSON with recall_query, environment_cues, and
    retrieval_document.
```

H.4 RELATION CLASSIFICATION

```
You compare one new evidence anchor with one existing CTEM memory item.

Guidelines:
1. Read the CTEM item as a stored cross-task experience backed by
    environment slices.
2. Read the new anchor as a grounded evidence unit that may join that
    CTEM item.
3. Decide whether the new anchor is related to the CTEM item.
4. If related, choose the relation that best describes how it should be
     stored.
5. Choose "refine" rather than "support" if the new anchor adds any
    applicability condition, exception, or scope restriction.
6. Choose "contradict" only when the new anchor provides a concrete
    failure, exception, or opposite case; do not use it for merely
    narrower context.

Update relation labels:
- support: The anchor supports the same rule or judgement.
- refine: The anchor adds a useful condition, context, variation, or
    boundary.
- contradict: The anchor provides a counterexample, exception, opposite
     case, or failure case.
- uncertain: The anchor seems related, but the relationship or boundary
     is unclear.

No-match option:
- none: The anchor is not related to the CTEM item and should remain
    separate.

Return valid JSON with matched, relation, and reason. Use relation = "
    none" only when matched is false.
```

H.5 INITIAL MEMORY CREATION

```
You create one new CTEM memory item from one Mirror Environment
    evidence slice.

Guidelines:
- Use only information supported by the evidence anchor and source span
    .
- If the trajectory was successful, write a reusable positive strategy.
- If the trajectory failed, do not turn the failed behavior into a
    positive rule. Instead, record the avoidable failure, missing check,
     or boundary condition.
- Keep the memory concise and agent-facing.
- Put exceptions in counterexamples and missing applicability
    conditions in unresolved.
- Do not invent conditions, exceptions, or unresolved issues that are
    not supported by the provided evidence.
- Output only valid JSON. Do not include markdown, comments, or
    explanation.

Output format:
{
  "memory": {
    "title": "string",
    "description": "string",
    "content": "string",
    "counterexamples": "string",
```

```
    "unresolved": "string"
  }
}
```

H.6 REFINEMENT

```
You update one CTEM memory item with one related evidence anchor.

Guidelines:
- Read the old CTEM memory item.
- Read the new evidence anchor and its Mirror Environment source span.
- Read related evidence anchors as additional context.
- Keep useful information from the old CTEM item.
- Add useful conditions, context, or boundaries from the new anchor.
- If related evidence differs, adjust the boundary instead of creating
    an over-broad rule.
- Do not invent conditions, exceptions, or unresolved issues that are
    not supported by the provided evidence.
- Write one updated CTEM memory item.
- Output only valid JSON. Do not include markdown, comments, or
    explanation.
```

H.7 CONTRADICTION

```
You update one CTEM memory item with one related evidence anchor that
    gives a conflicting case.

Guidelines:
- Read the old CTEM memory item.
- Read the new evidence anchor as a conflicting case, exception, or
    failure case.
- Read related contradict evidence anchors as additional context.
- Keep useful information from the old CTEM item.
- Add the conflicting case from the new anchor into counterexamples.
- If the conflict changes how the memory should be read, update the
    description or content.
- Do not invent conditions, exceptions, or unresolved issues that are
    not supported by the provided evidence.
- Write one updated CTEM memory item.
- Output only valid JSON. Do not include markdown, comments, or
    explanation.
```

H.8 UNCERTAINTY

```
You update one CTEM memory item with one related evidence anchor whose
    relationship is unclear.

Guidelines:
- Read the old CTEM memory item.
- Read the new evidence anchor as related but unclear evidence.
- Read related uncertain evidence anchors as additional context.
- Keep useful information from the old CTEM item.
- Add what is unclear from the new anchor into unresolved.
- If the unclear evidence suggests a possible boundary, mention it as a
     possible difference.
- Do not invent conditions, exceptions, or unresolved issues that are
    not supported by the provided evidence.
```

```
- Write one updated CTEM memory item.
- Output only valid JSON. Do not include markdown, comments, or
    explanation.
```

H.9 ACTOR MEMORY GUIDANCE

```
You are given CTEM memory items distilled from related Mirror
    Environment evidence anchors.

Each memory item summarizes a cluster of related evidence anchors and
    may mention important differences, boundaries, exceptions, or
    uncertainty.

Use these memory items as guidance, not as absolute rules.
Adapt them to the current task and current environment.
Use the current page state to decide which memory items apply.
If a memory item's counterexamples or unresolved conditions match the
    current state, verify carefully before applying the memory.
Keep the action syntax required by the environment.
```