ON THE FRAGILITY OF SELF-IMPROVING AGENTS: VARIANCE, TASK ORDER, AND UNDERSPECIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory-based self-improving agents—those that learn from an online stream of tasks and improve over time by maintaining a textual memory bank—have shown great promise in recent literature. However, the reliability aspects of these methods have been critically overlooked. In this work, we conduct a comprehensive re-evaluation of two memory-based methods, broadening the scope of evaluation along two axes: (1) including multiple self-improving runs to quantify *variance*, and (2) randomly shuffling the tasks to investigate the effect of *task order*. Through these experiments, we make two observations that expose the fragility of current methods: First, agent evaluation is inherently noisy in complex environments and on multi-step tasks, and stacking a self-improving loop on top can further amplify this noise. Empirically, when these methods are applied, we observe the variance across runs increase in 71% of cases, and the gap between the best and worst runs of the same experiment can reach up to 10 percentage points. Second, the agent's improvement is highly dependent on task order. Prior works often adopt default orderings that impose an implicit curriculum, acting as a hidden prerequisite for success. In contrast, when evaluated under a shuffled task order, agent performance degrades (-4.5%) instead of exhibiting the expected improvement (+1.5%).

To better understand this fragility, we manually examine the agents' memory and hypothesize that task and environment *underspecification* contribute to this fragility. Without clear specifications, agents generate plausible yet inapplicable memories (*e.g.*, recommending API usage in a browser-only environment) that may distract the agent from feasible strategies. We validate this hypothesis by incorporating information that enables better specification, such as detailed rubrics and environment feedback, into the memory construction process. While this added information partially closes the performance degradation in previous experiments, significant gaps still remain, suggesting that other uncharacterized factors contribute to this fragility. Looking ahead, our work advocates for more rigorous evaluation protocols for self-improving agents by reporting results across multiple runs and stress-testing them under realistic, challenging conditions. Moreover, our findings on underspecification call for systems and interfaces that enable effective human oversight, preventing agents from failing in unforeseeable ways.

1 INTRODUCTION

The development of self-improving agents (Schmidhuber, 2006; Zelikman et al., 2022; Huang et al., 2023)—systems designed to learn from past experiences and autonomously refine their performance over time—has emerged as a highly promising research direction. If they realize this potential, they could not only better automate routine workflows and create significant economic impact, but also enable highly adaptable systems and facilitate open-ended research discovery. While recent works show progress in this direction (Zhuge et al., 2024; Hu et al., 2025; Zhou et al., 2026; Fang et al., 2025; Murty et al., 2025; Zheng et al., 2025), little focus has been placed on stress-testing the reliability of such systems (Rabanser et al., 2026). In practical settings, such as enterprise deployments, there is often minimal tolerance for error; an initial failure risks losing user trust. Moreover, initial mistakes can cascade silently over the long term, causing irreversible or prohibitively costly impacts if adopted in high-stakes domains. Before we can confidently deploy these systems, we must critically evaluate their reliability under realistic, challenging conditions.

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.

This work begins with a comprehensive re-evaluation of two representative memory-based self-improving systems—Agent Workflow Memory (Wang et al., 2025b) and ReasoningBank (Ouyang et al., 2025)—across three realistic web-browsing benchmarks: WebArena (Zhou et al., 2024), VisualWebArena (Koh et al., 2024), and SCUBA (Dai et al., 2025). In this evaluation, we establish a stronger baseline by upgrading both the underlying language model and the agent harness. Furthermore, we broaden the scope of evaluation along two axes: conducting multiple self-improving runs of the same experimental setting to quantify variance, and randomly shuffling the tasks to investigate the effect of task order. Through this extended re-evaluation, we identify two critical reliability issues.

First, we show that the evaluation variance can be concerningly large, an observation largely hidden in prior works, as web agent and self-improving agent evaluations typically report only single-run results. We show that the no-memory, non-self-improving baseline agent already exhibits significant variance across runs, and that incorporating a self-improving loop can further amplify the variance. For example, on the WebArena GitLab subset (comprising 180 tasks), the absolute performance gap between the best and worst baseline runs is 4.4%, whereas applying ReasoningBank enlarges this gap to 7.8%. These observations call the reliability of single-run evaluation practices into question.

Second, we demonstrate that agent improvement is highly sensitive to task order. The default task order in prior works imposes an implicit, easy-to-hard curriculum, which acts as a hidden prerequisite for these self-improving methods to succeed. Under the default order, the agent using ReasoningBank achieves an average performance gain of 1.5%; however, when evaluated under randomly shuffled task orders, the agent exhibits a performance degradation of 4.5%. This dependency introduces a pressing concern, as real-world applications cannot assume a neatly ordered stream of user requests.

To better understand this fragility, we manually investigate the memories written by the agents, and identify environment and task underspecification during memory generation as potential drivers for the large variance and degraded performance. For example, when the memory construction model is unaware that the environment is browser-only and does not support code execution, it generates suggestions to use APIs—a strategy that is plausible but unexecutable. Regarding task underspecification, we show that ambiguous task queries can give rise to misunderstanding, overthinking, and the construction of erroneous memories that propagate to subsequent tasks.

To mitigate these issues, we consider incorporating additional context—such as environment feedback and task rubrics—into the memory construction step, and we further revise the prompt with clearer environment specifications. These interventions close 31% of the performance degradation observed in the shuffled task order settings. While this demonstrates the utility of such additional information, the remaining performance gap suggests that other uncharacterized factors contribute to this fragility, highlighting the need for future work to ensure the reliability of self-improving agents.

Beyond these empirical results, our work calls for research into rigorous evaluation methodologies for self-improving AI systems, and we provide practical recommendations in this direction. Furthermore, our analysis of underspecification motivates the development of effective human intervention interfaces, where the "wrong lessons" learned by these agents can be identified and corrected timely.

2 Background: Memory-Based Self-Improving Agents

In this section, we first revisit the problem setting of memory-based self-improving agents. We then provide a brief overview of two representative methods that we use in our analysis.

2.1 Problem Setting

We follow the setting described in prior literature (Wang et al., 2025b;a; Ouyang et al., 2025). The agent is required to complete an online stream of N tasks, $\mathcal{Q} = \{q_0, q_1, ..., q_N\}$ and maintain a textual memory $\mathcal{M}$. For each task q_i, the agent, backed by a language model L, is required to generate an action $a_t \in \mathcal{A}$ from past observations ($o_{0:t-1}$) and actions ($a_{0:t-1}$) at each timestamp t, *i.e.*, $a_i \leftarrow \pi_L(o_{0:t-1}, a_{0:t-1}; \mathcal{M}, \mathcal{A})$. At the end of each task, the agent receives a reward $r \in [0, 1]$, where a reward of 1 indicates that the agent "passes" the task. A memory construction module C will be instructed to update the memory $\mathcal{M}$ based on the task trajectory and optionally the reward. See Fig. 1(a) for an illustration.

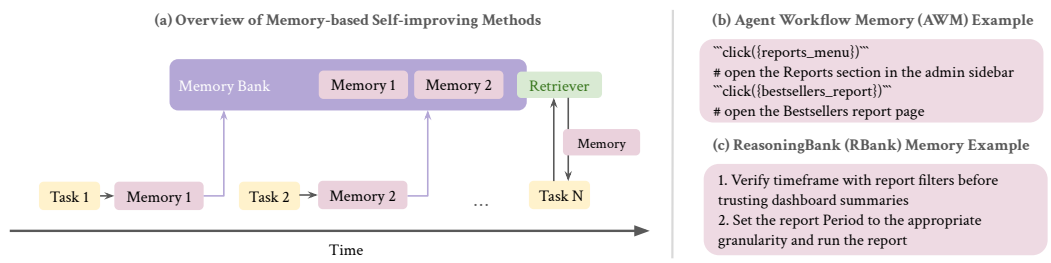

Figure 1: **(a) Overview of Memory-based Self-improving Methods.** Agents will perform a sequence of tasks and generate memories at the end of each task. The memories will be saved to a memory bank and will be retrieved for performing future tasks. **(b)(c) Example memories.** Two methods (AWM (Wang et al., 2025b) and RBank (Ouyang et al., 2025)) are considered in our evaluation.

2.2 Representative Approaches

We consider two representative approaches in our evaluation, and we describe them below.

- **Agent Workflow Memory (AWM)** (Wang et al., 2025b). At the end of each task, the memory module is instructed to summarize reusable workflows from successful trajectories (Fig. 1(b)). Following the original implementation, each new workflow is checked for duplication against existing workflows, and all workflows are included in the agent's context when performing subsequent tasks.
- **ReasoningBank (RBank)** (Ouyang et al., 2025). In this more recent approach, memory items were not limited to structured, step-by-step workflows, but more generic reasoning traces and insights that could be helpful for future similar tasks (Fig. 1(c)). Following the original implementation, memory items are generated from both successful and failed trajectories, and a retriever is used to select the most relevant past memories for subsequent tasks.

3 Exposing Fragility with Broadened Re-evaluation

3.1 Experiment Setting

Benchmarks. We focus our evaluation on the web browsing domain, and use the following three benchmarks.

- **WebArena** (Zhou et al., 2024). WebArena consists of 812 web browsing tasks involving six domains: Shopping, Shopping Admin, Gitlab, Reddit, Map and Multisite.
- **VisualWebArena** (Koh et al., 2024). VisualWebArena extends from WebArena and adds challenges by requiring agents to understand visual elements in the question or on the website. VisualWebArena has 910 tasks on three domains: Classifieds, Reddit and Shopping.
- **SCUBA** (Dai et al., 2025). SCUBA is an enterprise-centric benchmark on automating the activities of platform administrators (Admin), sales representatives (Sales), and service agents (Service) on a web-based customer relation management software. We manually excluded tasks that became invalid due to recent website updates, leaving a total of 267 tasks.

Evaluation practices in prior works. With very few exceptions (Yu et al., 2026; hattami et al., 2025), prior work on web agent evaluation reports pass rate over *one single self-improving run* of the agent. While this is permissible for early exploration in this field and for cost considerations, recent work points out that agent performance can vary across runs (Yao et al., 2024; Rabanser et al., 2026), and hence a comprehensive re-evaluation over multiple runs is needed. Further, when self-improvement methods are applied to web-browsing tasks (Wang et al., 2025b; Ouyang et al., 2025), experiments are typically conducted under *a pre-determined task order*. As our later analysis shows, this ordering induces an implicit easy-to-hard curriculum that may inadvertently favor these methods.

Evaluation practice in this work. Echoing the concerns above, we broaden our evaluation along two axes: (1) To better characterize the *variance*, we conduct *three identical self-improving runs* for each experiment of interest. In terms of evaluation metric, we focus on *run-level* statistics such as average

pass rate (pass@1) per domain, along with it standard deviation across three runs, and its best-worst gap among the three runs[1]. (2) To investigate the influence of *task order* in agent self-improving, we conduct experiments with two *shuffled task orders* (*i.e.*, Shuffle-1, Shuffle-2), in addition to the default order used in prior works.

Experiment details. We use the baseline agent harness used in Prabhu et al. (2026) for WebArena and VisualWebArena, and the baseline agent harness used in Dai et al. (2025) for SCUBA. They are referred to as "(no-memory) baseline" in the following. The self-improving methods (AWM and RBank) are built on top of these baseline agents. Unless otherwise specified, we use `GPT-5-mini` as the agent backbone model and the memory construction model. When constructing memories, we deliberately provide the ground-truth reward r to the model, in contrast to prior works which uses a proxy reward $\hat{r}$ from an LLM-Judge, which we found to be noisy and complicate our analysis[2].

3.2 VARIANCE

Web agent evaluation is intrinsically noisy. In Table 1, we first report variance-related metrics for the no-memory, non-self-improving baseline in the rows marked "Baseline". While noise and randomness across runs are expected, it is important to quantify the magnitude of variance across runs, especially since prior work typically reports average pass@1 from a single run. In the GitLab subset of WebArena, the gap between the best and worst runs reaches a surprising 4.4%, which in some contexts would often be considered a meaningful improvement. The standard deviation in this particular domain also reaches 2.0%. Similarly, in VWA and SCUBA, the domain-level best-worst gap can be up to 2.4% and 6.7% respectively, raising concerns that single-run evaluations may be misleading and could lead to inaccurate conclusions.

Self-improvement methods can amplify variance. We further compare the the rows of the baseline versus the self-improving methods in Table 1, where we see a trend that variance increases greatly when these methods are applied on top of the baseline. In 17 of the 24 cases, we see an increase in variances, and in 11 cases the relative increase exceeds 50%. This observation is a direct consequence of the *stateful* nature of self-improving methods: memory construction is conditioned on prior task outcomes and is further influenced by stochastic LLM sampling. As a result, early randomness can compound over time, leading to substantially different memory states across runs. While this phenomenon is expected, it is concerning from an evaluation perspective, as we observe standard deviations as large as 3.9%, and a best-worst gap also widened to 8.2% in the Map domain and 7.8% in the Gitlab domain of WebArena.

Self-improvement methods have diminishing effects when a strong "initialization" is used. Since the introduction of AWM and RBank, the capabilities of language models on web browsing tasks have advanced significantly. As shown in Table 2, RBank reports an average success rate of 53.9% on a subset of 684 WebArena tasks using `Gemini-2.5-Pro`. Our baseline agent, based on the more recent GPT-5-mini model, achieves comparable performance (55.3%) to their reported memory-enabled result (53.9%). On VisualWebArena, our baseline agent achieves 54.9%, which is comparable to the reported state-of-the-art result on this benchmark[3].

Building on this baseline, our re-evaluation enables a reassessment of the effectiveness of these self-improvement approaches under better "initializations". As shown in Table 3, we find that self-improvement methods struggle to yield consistent gains. For instance, although RBank achieves an average improvement of 1.5% over the no-memory baseline, a two-sided Welch's t-test on three runs gives $p = 0.23$, indicating that the gain is not statistically significant[4]. These results suggest that, while prior methods are effective for less capable web agents, more advanced approaches are required to construct high-quality memory and sustain the self-improvement process.

[1]We are primarily interested in the variance across self-improving runs, where one *run* refers to evolving over all tasks in the sequence. *Task-level* aggregated metrics such as pass@3 and pass^3 are not directly applicable here; see §D.

[2]We present selected results in the main paper and defer full result tables in §D. To validate our findings beyond web-browsing tasks and `GPT-5-mini`, §E introduces supplementary experiments with alternative models and benchmarks.

[3]The best performance on the VisualWebArena leaderboard is 54.0% as of May 1, 2026.

[4]The human performance on WebArena is 78% (Zhou et al., 2024), suggesting that improvement on WebArena is still possible.

Table 1: **Variance-related metrics across 3 runs.** Memory-based self-improving agent methods introduce increased variance in 17 out of 24 ($\approx 71\%$) cases, among which 11 cases show a relative increase of more than 50%.

Method	WebArena						VisualWebArena			SCUBA		
	Shopping (187)	Admin (182)	GitLab (180)	Map (109)	Reddit (106)	Multisite (48)	Classifieds (234)	Shopping (466)	Reddit (210)	Admin (158)	Sales (64)	Service (45)
Standard deviation of pass@1 across 3 runs (%)												
Baseline	1.53	1.19	1.98	1.30	1.78	0.98	1.01	0.83	1.03	0.30	1.95	2.77
AWM	1.26 (-18%)	2.99 (+152%)	1.20 (-39%)	1.98 (+53%)	2.48 (+39%)	0.98 (0%)	1.79 (+78%)	1.01 (+22%)	2.38 (+131%)	2.09 (+600%)	1.28 (-35%)	2.77 (0%)
RBank	1.33 (-13%)	2.12 (+79%)	3.34 (+69%)	3.89 (+200%)	2.22 (+25%)	4.28 (+336%)	1.23 (+22%)	1.58 (+91%)	1.47 (+43%)	1.58 (+429%)	1.47 (-24%)	3.78 (+36%)
Best-worst gap of pass@1 in 3 runs (%)												
Baseline	3.74	2.75	4.44	2.75	3.77	2.08	2.14	1.93	2.38	0.63	4.69	6.67
AWM	2.67 (-29%)	6.59 (+140%)	2.78 (-38%)	4.59 (+67%)	5.66 (+50%)	2.08 (0%)	4.27 (+100%)	2.15 (+11%)	5.71 (+140%)	4.43 (+600%)	3.12 (-33%)	6.67 (0%)
RBank	3.21 (-14%)	4.95 (+80%)	7.78 (+75%)	8.26 (+200%)	4.72 (+25%)	10.42 (+400%)	2.99 (+40%)	3.86 (+100%)	3.33 (+40%)	3.80 (+500%)	3.12 (-33%)	8.89 (+33%)

Relative change vs. Baseline shown in parentheses. Cell color: higher (worse) or lower (better) instability.

Table 2: **Contextualizing our WebArena results with those in prior works.** The no-memory baseline agent in this work (based on `GPT-5-mini`) already outperforms memory-enabled agents in prior works.

AWM (Claude-3.5-Sonnet) 812 Tasks		**RBank** (Gemini-2.5-Pro) 684 Tasks	
Baseline	32.7	Baseline	46.7
+AWM	36.3	+RBank	53.9
Our Baseline	54.8	Our Baseline	55.3

Table 3: **Overall results on three benchmarks.** We report average pass@1 across 3 runs. AWM/RBank cells show score and absolute gain over the baseline (in parentheses). With stronger baselines, these methods show diminishing performance gains.

Method	WebArena	VisualWebArena	SCUBA
Baseline	54.8	54.9	29.7
AWM	54.1 (-0.7)	54.5 (-0.4)	28.0 (-1.7)
RBank	56.3 (+1.5)	55.6 (+0.7)	31.1 (+1.4)

3.3 Task Order

Our previous results focus on broadening the evaluation through repeated, identical self-improving runs and examining the variance across those runs. These results were obtained using a default task order—from smaller task IDs to larger task IDs. In this section, we discuss the implicit assumptions underlying this design and further extend our analysis by exploring alternative task orderings.

The default order exhibits an implicit easy-to-hard curriculum. In Figure 2, we visualize the moving average (window = 30) of task success rates based on the no-memory baseline agent. In (a) and (b), the curves begin with high pass rates (around 75%), which subsequently decline to below 40% once the task ID exceeds 150. This pattern reflects an implicit easy-to-hard curriculum, where earlier tasks are more likely to fall within the agent's capability. In (c), the sequence also begins with easier tasks, although the second half exhibits a more complex oscillatory pattern.

The default ordering may be an artifact of the benchmark construction process, whereby annotators begin with simpler tasks and progressively introduce more challenging ones. While evaluating self-improvement approaches under this ordering is natural, real-world users may submit tasks in arbitrary orders. To stress-test self-improving systems prior to deployment, it is important to evaluate them under alternative task orderings.

Alternative task orders may lead to performance degradation. We extend our evaluation on WebArena and VisualWebArena to two additional shuffled task orders. We present the result overview in Fig. 3. On WebArena, we observe a significant performance degrade when using shuffled task orders. For example, when Shuffle-1 is used, performance drops from 54.8% to 49.1% (AWM) and 49.8% (RBank). On VisualWebArena, AWM is also significantly influenced by task order, while RBank is less sensitive to this change. While shuffled task orders create challenges for models to learn, we expect the overall success rate to be maintained at the same level as the no-memory baseline in these challenging settings; however, we see that performance often goes worse, raising concerns on the reliability of these self-improving methods in real-world settings.

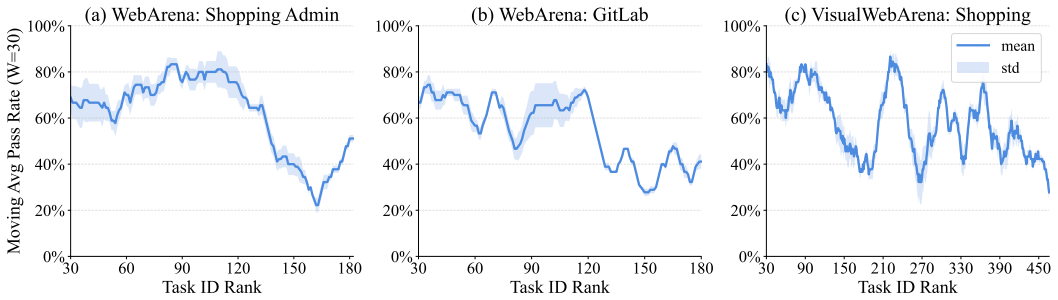

Figure 2: **Implicit curriculum in the default task order.** We visualize the moving average (window=30) of task pass rates as the no-memory baseline agent performs tasks in the default order. In (a)(b), success rates drop on tasks with larger task IDs. In (c), we observe a more complex oscillating pattern. In general, tasks of different difficulties are not distributed evenly in the default order.

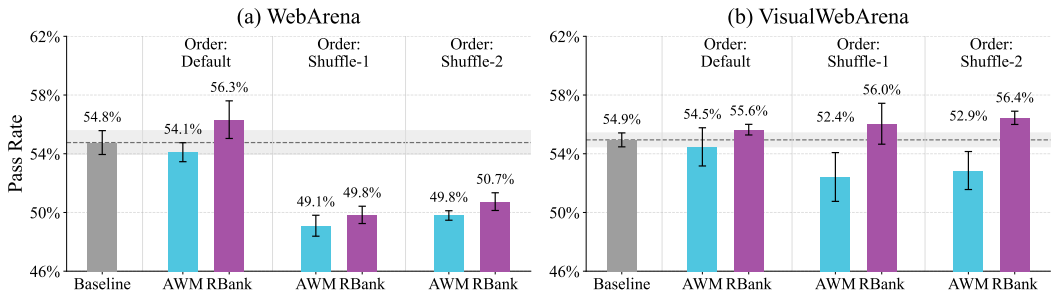

Figure 3: **Performance using different task orders.** We use the default task order and two shuffled orders. Model performance degrade significantly in 6 out of 8 cases when shuffled orders are used.

4 Indentifying Underspecification as a Potential Cause of Fragility

4.1 Uncovering the Failure Modes in Memory Construction

To understand the amplified variance and the unexpected performance drop when shuffled orders are used, we manually inspect the agent's memory produced by the RBank method, where we identify underspecification as a potential cause of these issues. We summarize three notable findings below.

Environment underspecification leads to recurring memories on unsupported actions. In our manual inspection, we identify several cases of unexpected memories that are plausible yet not applicable in the web agent's environment. For example, the agent frequently references "API" and recommends API-based solutions in its memory. However, the web browsing environment used in our experiments does not support such actions, and this key constraint is not provided to the memory proposal module in the RBank implementation, creating this mismatch.

In Fig. 4(a) and (b), we visualize the number of times the agent retrieves and conditions on memory items containing the keyword "API". We observe that such mentions appear across domains, potentially introducing distraction or confusion for the agent. Similarly, the agent frequently includes "user confirmation" in its memory (26 times in 3 runs of WebArena, and 22 times in 3 runs of VisualWebArena). While acquiring user confirmation is generally desirable, as many task queries are ambiguous, it is not supported by the evaluation environment. Consequently, such memory items may lead to situations in which the agent repeatedly issues "wait" actions until the time limit is reached.

Task underspecification leads to misunderstanding, overthinking and irrelevant memories. Task queries in WebArena are often ambiguous (hattami et al., 2025), leading to unintended behaviors from the agent. For example, in WebArena task 118, the agent is tasked with "I have jaw bruxism problem, show me something that could alleviate the problem." The intended goal is for the agent to locate a relevant product (*i.e.*, a mouth guard) on the shopping website; however, the agent instead

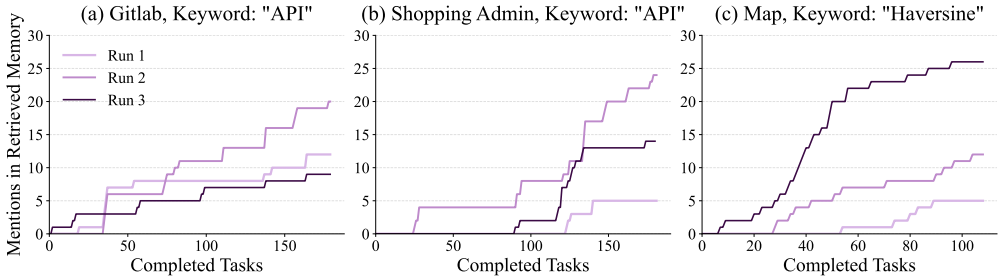

Figure 4: **Mentions of Selected Keywords in Retrieved Memory.** In (a)(b), we show that while the environment does not support API-based actions, agents increasingly save and retrieve memories about completing tasks with APIs. In (c), we show that in travel time related tasks in the map domain, the agent proposes to use Haversine Estimation instead of relying on the websites route engine, and this unintended strategy is amplified throughout the course of self-improvement.

Table 4: **Three types of additional information used in §4.2.** We hypothesize such information can reduce underspecification when constructing memories. Examples are truncated due to space limit.

Rubrics and Scores (+Rub):
Evaluator: string_match | Score: 0.0 (FAIL) Prediction: "Top-2 best-selling products in 2022: 1) Quest Lumaflex™ Band, 5 units; 2) Cruise Dual Analog Watch, 4 units." must_include: "Quest Lumaflex™ Band" → PASS must_include: "Sprite Stasis Ball 65 cm" → FAIL
Environment Feedback (+Env):
Actions: click(31), input_text(41, "05/01/2022"), input_text(44, "12/31/2022"), click(59) # Preceding Context Action error: Error executing action input_text: Failed to input text into index 41
Prompt Modification (+PMod):
- **INCLUDE**: Procedural knowledge, reusable workflows, site navigation structure, UI interaction tips. - **STRICTLY AVOID**: API or programmatic solutions, visiting external websites, requesting human confirmation, or task-specific details tied to a particular site or query.

adopts a more literal interpretation and responds with "consult your dentist or doctor for personalized advice." Further, when the agent is prompted to reflect on its failure on this task, it produces irrelevant memories such as "gather targeted patient/context details before giving medical guidance." We also observe cases in which benchmark evaluator bugs lead to false negatives during evaluation, which in turn cause the agent to overthink and generate spurious memories. See Table 14-15 for examples.

Memories are "contagious": the case of using Haversine Formula in map tasks. Surprisingly, in the map domain of WebArena, we observe references to the "Haversine Formula", a method for computing the shortest distance between two points on a sphere. Upon further inspection, we find that when the agent is tasked with obtaining the distance between two locations, the map website does not always load as expected or respond in time. As a result, the agent adopts a fallback strategy of estimating distances using the Haversine Formula. This approximation occasionally yields correct answers (see Table 16 for a detailed example), which leads the agent to reinforce these unintended strategies into its memory. As shown in Fig. 4, the introduction of this strategy appears to be stochastic, and the earlier "Haversine" is added to memory, the more frequently it is retrieved and used in later tasks. This effect may contribute to increased variance across runs.

4.2 Addressing Underspecification with Additional Information

In the original implementation of RBank, the memory construction module is minimal and generic; *i.e.*, "You need to extract and summarize useful insights in the format of memory items based on the agent's successful trajectory" (see Appendix F for the full prompt). Based on our findings in §4.1, such generic descriptions may be insufficient, particularly when the baseline agent is already strong and more specific feedback or memory is required to achieve meaningful improvements.

In the following, we address this limitation by incorporating additional information during memory construction. We describe three types of such information below and provide examples in Table 4.

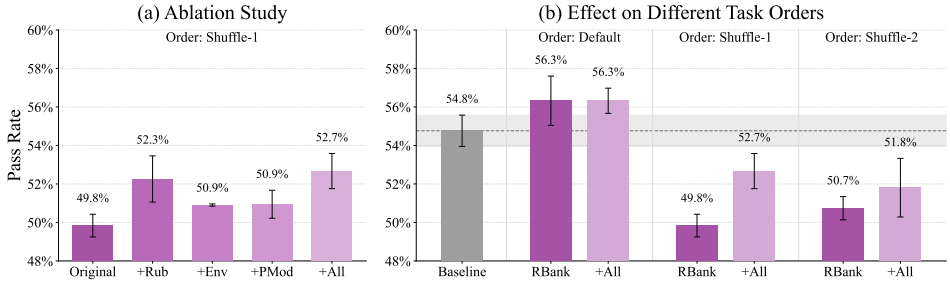

Figure 5: **Addressing Underspecification with Additional Information.** (a) We apply the three types of information individually and altogether. (b) We experiment with +All using three task orders.

- **Rubrics and Scores (+Rub).** Across all considered benchmarks, tasks are evaluated using function-based rubrics. For example, WebArena adopts three types of evaluation functions (`must_include`, `exact_match`, and `fuzzy_match`), which can be applied to the agent's returned answer or final state. In the jaw bruxism example discussed earlier (Table 14), we demonstrate that an ambiguous task query can lead to multiple interpretations and outcomes. Providing rubrics can help clarify the intended task and improve the relevance of the resulting memory. Note that these rubrics are provided only during post-completion memory construction; *i.e.*, the agent remains unaware of them while performing the task.
- **Environment Feedback (+Env).** We incorporate web environment feedback, such as whether click or selection actions are successful, into the memory construction process. In practice, we observe cases where the agent issues a sequence of actions (*e.g.*, click item 31, then fill textbox 41). However, after clicking item 31, textbox 41 may no longer be present on the page, resulting in an action error. Such information is not included in the original memory generation process, which can lead the agent to repeat similar UI interaction mistakes. Incorporating this feedback provides a useful signal for improving performance in future tasks.
- **Prompt Modification (+PMod).** Our previous analysis shows that the agent may "learn" unintended strategies, such as proposing API-based actions or requesting confirmation from human users. While these behaviors are plausible, they are not supported in the agent environment. To address this, we introduce a prompt edit that explicitly discourages such strategies. Additionally, we encourage the inclusion of procedural knowledge and site navigation structures, thereby guiding the model to generate memories that are more relevant to web-browsing tasks.
- **+All.** Finally, we experiment with including all three modifications mentioned above.

Results and Discussion. We begin by applying these modifications to the RBank experiments under the Shuffle-1 task order. The results are presented in Fig. 5(a), where we observe that each type of additional information yields modest performance gains. When combined, these modifications result in a 2.9% improvement over the original RBank method (49.8% $\rightarrow$ 52.7%). We further evaluate the +All setting under the Default and Shuffle-2 orders, with results shown in Fig. 5(b). We observe an improvement of 1.1% on Shuffle-2, while performance under the Default order is maintained.

Overall, these findings suggest that incorporating additional information during memory construction is beneficial and non-disruptive; thus, when such information is available, its incorporation during memory construction is recommended. However, even with these enhancements, the agent still underperforms the no-memory baseline under Shuffle-1 and Shuffle-2. The three types of additional information represent our initial attempt to address this issue and may not fully capture all sources of underspecification. Other factors contributing to robustness under varying task orders may also exist and remain unidentified. We leave further investigation of this issue to future work.

5 Related Works

Agent Continual Learning and Self-Improvement. This work focuses on an inference-only setting for agent self-improvement, motivated by its practical utility, accessible interface, and as a way to explore the limits of agents in forward-pass only settings. Beyond web-browsing tasks, related self-improvement methods have been studied for math, reasoning, and coding agents (Hu et al., 2025; Zhang et al., 2026; Wang et al., 2026), as well as long-term personalization (Maharana et al., 2024;

Shaikh et al., 2026). More broadly, continual learning and self-improvement (or "self-play") have long been studied prior to the inference-only paradigm, including in supervised fine-tuning (Zelikman et al., 2022; Chen et al., 2024) and reinforcement learning settings (Silver et al., 2018).

Agent Skill and Memory. One promising direction for agent continual learning is the construction of "memories", including both parametric and non-parametric forms. This work focuses on textual (non-parametric) memory, which has gained significant traction (Xu et al., 2026; Jiang et al., 2025; Zhong et al., 2024; Packer et al., 2023; Chhikara et al., 2025). Relatedly, workflow-related memories are often referred to as "skills" and has been extensively studied (Zhang et al., 2025; Li et al., 2026b; Tian et al., 2026; Jiang et al., 2026). For parametric memory, representative approaches include sparse memory fine-tuning (Lin et al., 2025a), which updates trainable memory slots based on their relative activation compared to a background corpus. Memory mechanisms have been studied from a model architecture perspective, with notable works including Hope and Titans (Behrouz et al., 2026a;b).

Agent Evaluation and Reliability. The research community has developed diverse benchmarks for evaluating agent capabilities, in domains such as tool use (Schick et al., 2023; Wang et al., 2024), computer use (Xie et al., 2024), web browsing (Zhou et al., 2024; Koh et al., 2024; Deng et al., 2023). Along these contributions, there is also growing concern regarding the evaluation sciences of agents (Rabanser et al., 2026; Grace et al., 2026; Zhu et al., 2025; Kapoor et al., 2025). In particular, Rabanser et al. (2026) advocate treating reliability as a primary evaluation axis, a perspective we strongly agree with. We extend this viewpoint from single-session agents to multi-session self-improving agents, where reliability becomes even more critical and harder to measure.

6 Conclusion

In this work, we conducted a comprehensive re-evaluation of two established memory-based methods for agent self-improvement, broadening the scope of evaluation along two axes (multiple runs and shuffled task orders) while using a stronger combination of base model and agent harness as the baseline (*i.e.*, the starting point for self-improvement). Overall, our results reveal the fragility of these self-improving systems under more challenging settings: the results of self-improving experiments can exhibit significant variance, meaning that evaluation based on a single run can hinder accurate measurement of progress. Furthermore, agents do not self-improve, but instead degrade, in challenging settings where the baseline is already strong or when tasks arrive in a random order.

Given these surprising findings, we further manually examine the agents' memory and identify environment and task underspecification as a key limitation in existing approaches, supporting this with detailed case studies and analyses. We address this limitation by incorporating additional information that provides better specification to the memory generation module. By doing so, the performance degradation observed in prior experiments is reduced by 31%, demonstrating promising progress. However, the remaining 69% performance gap suggests that there are still fundamental issues to be uncovered and resolved to fully overcome this fragility.

Beyond these empirical observations, our findings offer practical recommendations for researchers and practitioners working with self-improving agentic systems.

Recommendations for Evaluation. Our work advocates for more rigorous evaluation protocols that account for the stability issues discussed in this study. Specifically, results from a single run should be interpreted with caution; instead, reporting outcomes across multiple self-improving runs and randomized task orders should be highly encouraged. Furthermore, we recommend first piloting new methods on benchmarks with well-specified tasks, followed by stress-testing them on underspecified tasks to estimate worst-case performance prior to deployment.

Recommendations for System Development. Our analysis reveals that without proper validation mechanisms, agent memories are merely unverified hypotheses rather than true lessons learned. They can be erroneous and, more importantly, cascade negatively into subsequent tasks over time. Future work should investigate memory validation mechanisms that filter out problematic memories to ensure long-term robustness. Additionally, as demonstrated in §4.1, task and environment specification is a nuanced challenge. Because underspecification cannot be fully foreseen, and real-world deployments unfold as a single, irreversible run, refining specifications through an iterative process represents one viable path forward. Consequently, our findings motivate the development of interfaces that enable effective human oversight and timely intervention before agents fail unexpectedly.

AI USE STATEMENT

In this work, we used AI coding agents to help implement methods (AWM and ReasoningBank) and infrastructure for our experiments. We also used generative AI tools to edit and proofread the manuscript and to aggregate the experimental results reported in its figures and tables. We reviewed all AI-assisted work, including the experimental code, aggregated results, and manuscript edits. We take responsibility for the final content of this work, including any text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or the release of new datasets, and we do not identify foreseeable harmful applications of its findings. We therefore do not identify direct concerns under the ICLR Code of Ethics. Throughout the paper, we investigate unreliable and unintended behaviors of self-improving agents. We discuss the associated concerns and recommendations in §6.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide experimental details omitted from the main text due to space constraints in §C. We include the prompts used by the two compared methods in §F and notable agent trajectories from our experiments in §G. An anonymized version of the source code is included in the supplementary material.

REFERENCES

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJGven05Y7.

Parth Asawa, Christopher M. Glaze, Gabriel Orlanski, Ramya Ramakrishnan, Benji Xu, Asim Biswal, Vincent Sunn Chen, Frederic Sala, Matei Zaharia, and Joseph E. Gonzalez. Continual learning bench: Evaluating frontier ai systems in real-world stateful environments, 2026. URL https://arxiv.org/abs/2606.05661.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation, 2025. URL https://arxiv.org/abs/2503.11926.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026a. URL https://openreview.net/forum?id=nbMeRvNb7A.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026b. URL https://openreview.net/forum?id=8GjSf9Rh7Z.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models, 2024.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv*, 2025. doi: 10.48550/arxiv.2504.19413.

Yutong Dai, Krithika Ramakrishnan, Jing Gu, Matthew Fernandez, Yanqi Luo, Viraj Prabhu, Zhenyu Hu, Silvio Savarese, Caiming Xiong, Zeyuan Chen, and Ran Xu. Scuba: Salesforce computer use benchmark, 2025. URL https://arxiv.org/abs/2509.26506.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=kiYqbO3wqw.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks?, 2025. URL https://arxiv.org/abs/2509.16941.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020. URL https://arxiv.org/abs/2002.06305.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. WebEvolver: Enhancing web agent self-improvement with co-evolving world model. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8959–8975, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.454. URL https://aclanthology.org/2025.emnlp-main.454/.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

Mikaela Grace, Jeremy Hadfield, Rodrigo Olivares, and Jiri De Jonghe. Demystifying evals for AI agents. Anthropic Engineering Blog, Jan 2026. URL https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents. Accessed: 2026-03-24.

Melody Y. Guan, Miles Wang, Micah Carroll, Zehao Dou, Annie Y. Wei, Marcus Williams, Benjamin Arnav, Joost Huizinga, Ian Kivlichan, Mia Glaese, Jakub Pachocki, and Bowen Baker. Monitoring monitorability, 2025. URL https://arxiv.org/abs/2512.18311.

Amine El hattami, Joel Bredeson, Megh Thakkar, Eric LaBouve, Nicolas Chapados, and Christopher Pal. Webarena verified, 2025. URL https://openreview.net/forum?id=CSIo4D7xBG.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters, 2019. URL https://arxiv.org/abs/1709.06560.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.67. URL https://aclanthology.org/2023.emnlp-main.67/.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng Wu. CRMArena: Understanding the capacity of LLM agents to perform professional CRM tasks in realistic environments. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*

(Volume 1: Long Papers), pp. 3830–3850, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.194. URL https://aclanthology.org/2025.naacl-long.194/.

Kung-Hsiang Huang, Akshara Prabhakar, Onkar Thorat, Divyansh Agarwal, Prafulla Kumar Choubey, Yixin Mao, Silvio Savarese, Caiming Xiong, and Chien-Sheng Wu. CRMArena-pro: Holistic assessment of LLM agents across diverse business scenarios and interactions. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=EPlpe3Fx1x.

Bowen Jiang, Zhuoqun Hao, Young Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo Jose Taylor, and Dan Roth. Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=6ox8XZGOqP.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents, 2026. URL https://arxiv.org/abs/2603.12056.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation, 2025. URL https://arxiv.org/abs/2510.11977.

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

Jacob Xiaochen Li, Rick Battle, and Omar Khattab. Machine studying: A system-level reframing of continual adaptation from declarative corpora. In *Continual Adaptation at Scale: Towards Sustainable AI Workshop*, 2026a. URL https://openreview.net/forum?id=e7duOnFgL7.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han chung Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026b. URL https://arxiv.org/abs/2602.12670.

Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. Continual learning via sparse memory finetuning. *arXiv preprint arXiv:2510.15103*, 2025a. URL https://arxiv.org/abs/2510.15103.

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E. Gonzalez. Sleep-time compute: Beyond inference scaling at test-time, 2025b. URL https://arxiv.org/abs/2504.13171.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Shiva Krishna Reddy Malay, Shravan Nayak, Sagar Davasam, Srinivas Sunkara, and Sai Rajeswar. Enterpriseops-gym: Environments and evaluations for stateful agentic planning and tool use in enterprise settings. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=QKy6cKEiEB.

Kevin Meng, Vincent Huang, Jacob Steinhardt, and Sarah Schwettmann. Introducing docent. https://transluce.org/introducing-docent, March 2025.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? a call for multi-prompt llm evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 08 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00681. URL https://doi.org/10.1162/tacl_a_00681.

Shikhar Murty, , Hao Zhu, Dzmitry Bahdanau, and Christopher D Manning. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *arXiv preprint arXiv:2410.02907*, 2025.

NeoCognition. ApprenticeBench. https://neocognition.io/blog/apprentice-bench, 2026. Accessed: 2026-09-25.

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, 2025. URL https://arxiv.org/abs/2509.25140.

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

Vassilis Papadopoulos, McNair Shah, Sam Zimmerman, and Jack Lindsey. Mind viruses: Self-propagating ideas in multi-agent llm systems, 2026. URL https://arxiv.org/abs/2608.10218.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. Gdpval: Evaluating ai model performance on real-world economically valuable tasks, 2025. URL https://arxiv.org/abs/2510.04374.

Viraj Prabhu, Yutong Dai, Matthew Fernandez, Krithika Ramakrishnan, Jing Gu, Yanqi Luo, silvio savarese, Caiming Xiong, Junnan Li, Zeyuan Chen, and Ran Xu. WALT: Web agents that learn tools. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=cgIDqcJcoI.

Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a science of ai agent reliability, 2026. URL https://arxiv.org/abs/2602.16666.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Juergen Schmidhuber. Gödel machines: Self-referential universal problem solvers making provably optimal self-improvements, 2006. URL https://arxiv.org/abs/cs/0309048.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RIu5lyNXjT.

Omar Shaikh, Valentin Teutschbein, Kanishk Gandhi, Yikun Chi, Nick Haber, Thomas Robinson, Nilam Ram, Byron Reeves, Sherry Yang, Michael S. Bernstein, and Diyi Yang. Learning next action predictors from human-computer interaction, 2026. URL https://arxiv.org/abs/2603.05923.

Jiajun Shi, Siyuan Tao, Yuhao Wu, Zexuan Wang, Jingyuan Zhang, Jiaheng Liu, Xinping Lei, Xinrong Zhang, Siyuan Fang, Zhewen Tan, Tianle Cai, Junhao Fang, Jiameng Huang, Yueyang Wang, Jinkai Liu, Yuxuan Zhang, Jian Yang, Zhoujun Li, Shen Yan, Wenhao Huang, and Ge Zhang. S3gym: Can llms turn self-testing and self-judging into self-improvement?, 2026. URL https://arxiv.org/abs/2608.31100.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

Yu Tian, Jiawei Chen, Lifan Zheng, Mingxiang Tao, Xinyi Zeng, Zhaoxia Yin, Hang Su, and Xian Sun. Skills-coach: A self-evolving skill optimizer via training-free grpo, 2026. URL https://arxiv.org/abs/2604.27488.

Anton Voronov, Lena Wolf, and Max Ryabinin. Mind your format: Towards consistent evaluation of in-context learning improvements. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6287–6310, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.375. URL https://aclanthology.org/2024.findings-acl.375/.

Wenyi Wang, Piotr Piękos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-g´odel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T0EiEuhOOL.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=jJ9BoXAfFa.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=lsAY6fWsog.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=NTAhi2JEEE.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=tN61DTr4Ed.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=FiM0M8gcct.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Simon Yu, Gang Li, Weiyan Shi, and Peng Qi. Polyskill: Learning generalizable skills through polymorphic abstraction for continual learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KdEsujyiSV.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with agent skills. Anthropic Engineering Blog, Oct 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Accessed: 2026-05-06.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pUpzQZTvGY.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=cO1IH43yUF.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models, 2021. URL https://arxiv.org/abs/2102.09690.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. URL https://arxiv.org/abs/2504.07079.

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://openreview.net/forum?id=Pc8AU1aF5e.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, pp. 19724–19731, 2024.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Yifei Zhou, Sergey Levine, Jason E Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=9yusqX9DpR.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, Zixin Wen, Zhonglin Xie, Chenhui Gou, Linxuan Ren, Yueyang Wang, Junfeng Zhong, Rui Liu, Tian Gao, Yangguang Lin, Jingyuan Zhang, Maojia Song, Xuan Qi, Jinhong Wu, Chenyang Zhang, Yinzhu Piao, Ziru Niu, Hongbin Lin, Lingxiang Meng, Peng Tang, Chengyao Tang, Shanyu Wu, Huanyu Zheng, Yu Liu, Liya Zhu, He Wang, Ming Ding, Ziyu Wan, Hao Liu, Sibo Wang, Haotian Zhu, Xintian Zhang, Nan Chai, Yipeng Liu, Panhao Lai, Sihang Yuan, Zixin Su, Ge Zhang, Wangchunshu Zhou, Yantao Du, Wenhao Huang, and Guang Shi. Edgebench: Unveiling scaling laws of learning from real-world environments, 2026. URL https://arxiv.org/abs/2607.05155.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellermann, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing best practices for building rigorous agentic benchmarks, 2025. URL https://arxiv.org/abs/2507.02825.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=uTC9AFXIhg.

A Limitations

Our evaluation in the main paper focuses on a specific class of memory-based self-improving agents applied to web-browsing tasks with a single model. To keep the experimental design tractable, we did not extend the core analysis to other types of self-improvement methods, other domains, or a wider range of models, and given the rapid pace of development in this field, we could not evaluate the most recently introduced memory management techniques. To address these limitations to the best of our ability, we conducted additional experiments in the appendix (§E: two additional models, a filesystem-based self-improving method, an enterprise tool-use benchmark, and a larger number of self-improving runs). These results broadly corroborate our main findings and suggest that the reliability issues we identify are not confined to our primary setting.

Secondly, our manual inspection of agent memories was qualitative and covered only a subset of the memories produced. Given the sheer volume of memory entries generated in our experiments, an exhaustive review was infeasible. While our analysis highlights several critical failure modes, other undocumented patterns likely exist. Future work would benefit from analyzing agent memories at scale and developing interfaces for human review of a large volume of agent memories.

Lastly, underspecification only explains part of the performance degradation, and the reasons behind the remaining performance gap are still unknown. It likely arises from multiple interacting factors: the quality of the underlying tasks, the design of the self-improvement and memory mechanisms, and the characteristics of the base LLM. Systematically attributing the degradation to these factors would require the scalable memory analysis pipeline described above, benchmark re-annotation efforts, and extensive experiments to isolate the effect of each factor. We leave this for future work.

B Extended Related Works

Agent Evaluation (Continued). In addition to the agent benchmarks discussed in §5, recent benchmarks increasingly target economically valuable tasks (Patwardhan et al., 2025), including coding (Jimenez et al., 2024; Deng et al., 2025), customer service (Yao et al., 2024; Barres et al., 2025), customer relationship management (Huang et al., 2025; 2026), and enterprise operations (Malay et al., 2026). During the preparation of this work, several new benchmarks were released to assess how agents continously learn and adapt through corpus study, sequences of related tasks, or interactions within an environment, including StudyBench (Li et al., 2026a), EdgeBench (Zhu et al., 2026), CL-Bench (Asawa et al., 2026), S^3Gym (Shi et al., 2026), and ApprenticeBench (NeoCognition, 2026). Notably, ApprenticeBench, CL-Bench, and EdgeBench evaluate multiple agent self-improving runs, while CL-Bench additionally account for instance order. Together, these evaluation choices reflect a growing recognition that run-level variance and order sensitivity are central to rigorously assessing self-improving agents.

Monitorability. Our work is also broadly related to literature on chain-of-thought and agent monitorability (Baker et al., 2025; Guan et al., 2025; Meng et al., 2025). For instance, Baker et al. (2025) observe instances of reward hacking in coding agents via chain-of-thought monitoring. This aligns closely with our findings, as we observe similar cases of "reward hacking" (*e.g.*, the Haversine Formula example in Table 16) occurring during the agent's learning process. Because memory-based continual learning provides an accessible textual interface to inspect an agent's "learning," it introduces opportunities to explore scalable monitoring methods, as well as interfaces to intervene in the

process. A recent work by Papadopoulos et al. (2026) examine safety risks in multi-agent systems where agents share context through coding repositories or exchange information through handoffs. The findings are broadly related to our "memories are contagious" finding in §4.1, although we focus on a single-agent self-improvement setting.

Reliability and Stability Concerns of Emerging Learning Paradigms. Reliability has been a recurring challenge across evolving learning paradigms in AI. Prior works have extensively investigated the stability of deep reinforcement learning (Henderson et al., 2019), gradient-based meta-learning (Finn et al., 2017; Antoniou et al., 2019), language model fine-tuning (Zhang et al., 2021; Dodge et al., 2020), prompting (Sclar et al., 2024; Mizrahi et al., 2024), in-context learning (Zhao et al., 2021; Lu et al., 2022; Voronov et al., 2024), and self-correction (Huang et al., 2024). Self-improving agents represent a nascent paradigm enabled by increasingly capable language models. However, unlike earlier paradigms confined to passive prediction or generation, modern agents are increasingly integrated into real-world systems where they are granted the autonomy to execute irreversible actions. Consequently, deploying them without systematically addressing underlying fragility risks not only compounding computational errors, but tangible real-world harm, making reliability an urgent prerequisite for safe adoption.

C Additional Experiment Details

Costs. When using `GPT-5-mini` as the backbone model, a single run over all 812 WebArena tasks costs around $25, and a single run over all 267 SCUBA tasks costs $29. These estimates are based on the no-memory baselines. For memory-based methods, additional costs arise from (a) generating memories via extra LM calls and (b) including memories in the context. However, we expect the additional cost from both components to be minimal. All experiment code are hosted and run from a standard 64-CPU server.

LLM configurations. When using `GPT-5-mini`, we use the default (medium) reasoning efforts and the default temperature (1.0) during generation.

Dataset and Code License. The WebArena benchmark (Zhou et al., 2024) is released under Apache-2.0 license. The VisualWebArena benchmark (Koh et al., 2024) is released under MIT license. The SCUBA benchmark (Dai et al., 2025) is released under Apache-2.0 license. Our baseline agents were revised from WALT (Prabhu et al., 2026) and SCUBA (Dai et al., 2025), released under MIT license and Apache-2.0 license respectively. In §E, we conducted additional experiments using the EnterpriseOps-Gym benchmark (Malay et al., 2026) and the Letta Code framework (Packer et al., 2023; Lin et al., 2025b). Both are both released under Apache-2.0 license.

D Result Tables

Due to space constraints, the main paper reports only aggregated metrics (*e.g.*, pass@1 averaged over domains; Table 3). In this section, we present results under alternative metrics (pass@k and passˆk) for the no-memory baseline, as well as a per-domain breakdown for all compared methods.

D.1 Pass@3 and Passˆ3 Metrics for the No-Memory Baseline

For the no-memory baseline, each task in the benchmark is performed *independently*, so pass@k and passˆk (Yao et al., 2024) are natural metrics for capturing *task-level* variance across repeated executions of the same task. Note that this differs from the *run-level* variance discussed in the main paper, which measures variance across multiple self-improving runs.

We report both metrics in Table 5-6. Consistent with prior findings, agents are often quite inconsistent across runs: the gap between pass@3 and passˆ3 frequently exceeds 20%.

Table 5: Evaluation of the no-memory baseline agent on WebArena.

Metric	**Shopping** (187)	**Admin** (182)	**GitLab** (180)	**Map** (109)	**Reddit** (106)	**Multisite** (48)	**All** (812)
Avg. Pass	45.8	62.6	53.9	53.2	69.5	34.0	54.8
Pass@3	55.1	72.0	63.3	67.0	82.1	47.9	65.4
Passˆ3	34.8	52.2	45.0	40.4	53.8	20.8	43.3

Table 6: Evaluation of the no-memory baseline agent on VisualWebArena and SCUBA.

Metric	**VisualWebArena**				**SCUBA**			
	Classifieds (234)	**Shopping** (466)	**Reddit** (210)	**All** (910)	**Admin** (158)	**Sales** (64)	**Service** (45)	**All** (267)
Avg. Milestone	-	-	-	-	46.7	61.8	42.3	49.6
Avg. Pass	60.1	57.6	43.3	54.9	26.2	39.6	28.1	29.7
Pass@3	73.9	69.1	51.4	66.3	39.2	57.8	40.0	43.8
Passˆ3	44.9	45.1	33.8	42.4	13.9	20.3	13.3	15.4

D.2 Per-domain Results

For the memory-based methods (AWM and RBank), we conduct multiple self-improving runs, and task execution within a given run *depends* on the memory accumulated in that run. Consequently, pass@3 and passˆ3 are not directly applicable. Instead, we track the pass rate (pass@1) over the sequence of tasks within each self-improving run and report the mean, standard deviation, and best–worst gap across the three runs. For consistency, we report the same metrics for the no-memory baseline, even though it does not self-improve.

Results are presented in Tables 7–10, which also serve as the per-domain breakdown of Table 3 in the main paper.

We note that one could additionally evaluate each task three times within each of the three self-improving runs to measure pass@3/passˆ3. However, this would triple the cost of our evaluation protocol and cost nine times as much as the standard single-run evaluation, which is prohibitive in this work.

Table 7: **Per-domain pass rate (%) on WebArena.** Each cell reports mean/std/best–worst gap, based on three runs of memory construction. The right-most column ("All") reports the overall pass rate.

Method	Shopping (187)	Admin (182)	GitLab (180)	Map (109)	Reddit (106)	Multisite (48)	All (812)
No Memory	45.8/1.53/3.74	62.6/1.19/2.75	53.9/1.98/4.44	53.2/1.30/2.75	69.5/1.78/3.77	34.0/0.98/2.08	54.8
Task Order: Ordinal							
AWM	48.3/1.26/2.67	59.7/2.99/6.59	51.7/1.20/2.78	50.5/1.98/4.59	72.3/2.48/5.66	32.6/0.98/2.08	54.1
RBank	52.8/1.33/3.21	60.6/2.12/4.95	59.3/3.34/7.78	48.6/3.89/8.26	71.1/2.22/4.72	27.8/4.28/10.42	56.3
Task Order: Shuffle-1							
AWM	43.3/1.51/3.21	54.8/2.70/6.59	46.7/1.36/3.33	44.0/2.70/6.42	71.1/2.22/4.72	22.2/3.93/8.33	49.1
RBank	42.4/1.10/2.67	56.0/1.55/3.30	45.7/0.94/2.22	46.8/2.25/5.50	72.6/2.04/4.72	27.1/0.00/0.00	49.8
Task Order: Shuffle-2							
AWM	43.5/1.40/3.21	55.1/1.13/2.75	44.6/2.92/6.67	48.6/0.75/1.83	69.5/1.60/3.77	32.6/3.93/8.33	49.8
RBank	44.4/1.16/2.67	56.8/2.63/6.04	46.3/2.77/6.67	51.7/1.14/2.75	68.2/1.60/3.77	28.5/0.98/2.08	50.7

Table 8: **Per-domain pass rate (%) on VisualWebArena.** Each cell reports mean/std/best–worst gap, based on three runs of memory construction. The right-most column ("All") reports the overall pass rate.

Method	Classifieds (234)	Shopping (466)	Reddit (210)	All (910)
No Memory	60.1/1.01/2.14	57.6/0.83/1.93	43.3/1.03/2.38	54.9
Task Order: Ordinal				
AWM	61.3/1.79/4.27	55.3/1.01/2.15	45.1/2.38/5.71	54.5
ReasoningBank	59.5/1.23/2.99	58.3/1.58/3.86	45.4/1.47/3.33	55.6
Task Order: Shuffle 1				
AWM	57.3/0.92/2.14	53.2/3.04/6.65	45.2/0.67/1.43	52.4
ReasoningBank	60.3/2.09/5.13	58.9/2.33/5.15	45.1/0.22/0.48	56.0
Task Order: Shuffle 2				
AWM	58.4/2.13/5.13	54.9/0.63/1.50	42.1/2.76/6.67	52.9
ReasoningBank	62.5/0.53/1.28	58.7/0.27/0.64	44.8/2.43/5.71	56.4

Table 9: **Per-domain milestone score (%) on SCUBA.** Each cell reports mean/std/best–worst gap, based on three runs of memory construction. The right-most column ("All") reports the overall pass rate.

Method	Admin (158)	Sales (64)	Service (45)	All (267)
Baseline	46.7/0.25/0.59	61.8/1.57/3.36	42.3/1.40/3.02	49.6
Synapse (Zheng et al., 2024)	51.3/1.39/3.40	64.9/2.64/5.94	48.6/5.24/12.51	54.1
AWM	45.8/1.40/3.25	60.8/2.98/7.19	49.6/4.15/9.13	50.1
ReasoningBank	50.7/0.19/0.47	62.4/2.70/6.59	36.3/2.85/6.11	51.1

Table 10: **Per-domain pass rate (%) on SCUBA (binary success: milestone score $= 1.0$).** Each cell reports mean/std/best–worst gap, based on three runs of memory construction. The right-most column ("All") reports the overall pass rate.

Method	Admin (158)	Sales (64)	Service (45)	All (267)
Baseline	26.2/0.30/0.63	39.6/1.95/4.69	28.1/2.77/6.67	29.7
Synapse (Zheng et al., 2024)	30.4/1.37/3.16	43.2/2.66/6.25	28.9/1.81/4.44	33.2
AWM	22.4/2.09/4.43	37.5/1.28/3.12	34.1/2.77/6.67	28.0
ReasoningBank	30.2/1.58/3.80	40.1/1.47/3.12	21.5/3.78/8.89	31.1

E SUPPLEMENTARY EXPERIMENTS

Table 11: **WebArena Results, Three Additional Runs using `GPT-5-mini`.** Each cell reports avg/std/best-worst gap across n runs, and the right-most column ("All") reports the overall pass rate.

Method	Shopping (187)	Admin (182)	GitLab (180)	Map (109)	Reddit (106)	Multisite (48)	All (812)
$n = 3$ runs							
No Memory	45.8/1.5/3.7	62.6/1.2/2.8	53.9/2.0/4.4	53.2/1.3/2.8	69.5/1.8/3.8	34.0/1.0/2.1	54.8
RBank (Ordinal)	52.8/1.3/3.2	60.6/2.1/5.0	59.3/3.3/7.8	48.6/3.9/8.3	71.1/2.2/4.7	27.8/4.3/10.4	56.3
$n = 6$ runs							
No Memory	45.8/1.7/4.3	61.7/1.8/5.5	53.7/1.5/4.4	52.0/2.2/6.4	68.1/2.7/7.6	34.0/1.0/2.1	54.2
RBank (Ordinal)	51.1/2.5/8.0	60.8/1.9/5.0	58.3/2.8/8.9	47.3/3.1/9.2	71.4/2.3/6.6	28.1/3.9/10.4	55.6

Additional Runs (n=3 → n=6). In the main paper, we evaluated each memory-based method with three self-improving runs. To characterize the benefits of further increasing n, we conduct three additional runs on a subset of experiments (WebArena; no-memory baseline and ReasoningBank; ordinal order), for n=6 in total. In Table 11 we report the original (n=3) and extended (n=6) results.

We find that the spread widens from n=3 to n=6: best-worst gaps grow in most domains (*e.g.*, shopping: 3.21→8.02 for ReasoningBank), indicating that n=3 may understate the true variability, and larger n better characterizes the spread and worst-case behavior. The variance-amplification finding on `GPT-5-mini` holds at n=6; ReasoningBank shows a larger standard deviation than the no-memory baseline in 5 of 6 domains.

Finally, we perform Welch's t-tests comparing the no-memory baseline and ReasoningBank using the n=6 runs. On the overall success rate, this yields $p = 0.07$, which is lower than the $p = 0.23$ obtained with only three runs but still not significant at $\alpha = 0.05$; ReasoningBank's overall gain in this setting therefore cannot be distinguished from run-to-run noise. Per-domain results are mixed, with statistically significant differences in both directions for some domains.

Table 12: **WebArena Results using Additional Models.** Each cell reports avg/std/best-worst gap across three runs, and the right-most column ("All") reports the overall pass rate.

Method	Shopping (187)	Admin (182)	GitLab (180)	Map (109)	Reddit (106)	Multisite (48)	All (812)
`GPT-OSS-120B`							
No Memory	35.5/1.1/2.7	46.7/0.4/0.8	45.4/0.9/2.2	33.0/1.5/3.7	45.9/2.5/5.7	24.3/3.9/8.3	40.6
RBank (Ordinal)	38.0/0.9/2.1	49.0/1.7/4.1	42.4/0.3/0.6	30.0/2.2/4.6	48.1/5.4/13.2	18.8/1.7/4.2	40.5
RBank (Shuffle-1)	34.3/1.5/3.2	40.4/2.6/5.8	34.5/1.7/3.9	33.9/2.2/5.5	49.4/1.2/2.8	22.9/3.4/8.3	37.0
`Gemini-2.5-Pro`							
no-memory	40.3/1.1/2.7	59.2/2.4/5.2	48.7/2.1/5.0	33.9/1.3/2.8	70.4/2.4/5.7	33.3/3.4/8.3	49.0
RBank (ordinal)	42.8/1.2/2.7	59.3/1.6/3.8	51.1/4.3/9.4	37.0/1.6/3.7	75.2/1.6/3.8	30.6/2.6/6.3	51.0
RBank (shuffle1)	38.3/2.8/6.4	57.2/0.6/1.4	44.2/1.6/3.9	37.0/3.0/7.3	73.9/1.9/4.7	32.6/1.0/2.1	48.0

Additional Models (`GPT-OSS-120B` and `Gemini-2.5-Pro`). In the main paper, we restrict our scope to `GPT-5-mini` for all compared methods. In this section, we additionally evaluate `Gemini-2.5-Pro`, to more closely match the evaluation setup of ReasoningBank (Ouyang et al., 2025), and `GPT-OSS-120B`, whose open weights could facilitate future white-box analysis. We focus on WebArena with the no-memory baseline and ReasoningBank under the ordinal and Shuffle-1 task orders. We report the results in Table 12.

These results confirm two key findings of the main paper: (1) Variance across runs can be large for self-improving agents (*e.g.*, a 9.4-point best-worst gap on GitLab with `Gemini-2.5-Pro`, and a 13.2-point gap on Reddit with `GPT-OSS-120B`). (2) Task order matters, and shuffled orders can push performance below the no-memory baseline (`GPT-OSS-120B`: 37.0 vs. 40.6 overall; `Gemini-2.5-Pro`: 48.0 vs. 49.0).

Table 13: **Enterprise-Ops Gym Results.** Each cell reports avg/std/best-worst gap across three runs, and the right-most column ("All") reports the overall pass rate.

Method	Calendar	CSM	Drive	Email	HR	Hybrid	ITSM	Teams	All
`GPT-OSS-120B`									
No memory	21.3/2.7/6.6	21.4/0.8/1.9	41.2/2.0/4.7	54.2/2.5/6.0	12.4/0.9/2.0	23.6/1.5/3.7	5.8/0.8/1.9	35.5/4.3/9.8	24.5
With memory	26.2/2.7/6.6	19.7/2.8/6.8	33.9/6.3/14.1	52.7/1.4/3.0	5.9/1.4/2.9	25.6/2.0/4.9	1.3/0.5/1.0	36.1/3.5/8.2	22.3
`GPT-5.4-mini`									
No memory	38.3/0.8/1.6	21.4/1.6/3.9	47.9/2.7/6.3	58.7/2.5/6.0	20.6/1.4/2.9	24.8/4.0/9.8	25.6/3.7/7.8	21.3/2.3/4.9	30.5
With memory	36.1/1.3/3.3	25.9/1.7/3.9	51.6/3.4/7.8	56.7/1.2/3.0	16.7/2.1/4.9	23.6/5.8/13.4	24.6/1.2/2.9	15.3/4.7/11.5	29.7

Additional Benchmark (EnterpriseOps-Gym) and Baseline (Memory Filesystem). The main paper focuses on web-browsing tasks and two memory-based self-improving methods. To assess the generality of our findings, we conduct a new set of experiments with an additional self-improving method and an additional benchmark. The new method is built on the Letta Code framework (Packer et al., 2023; Lin et al., 2025b)[5]. After each task, the binary task reward is revealed. A reflection agent inspects the task trajectory and writes memory to a filesystem, where it can modify system prompts and add skills or references by creating and editing files. This design supports multi-step edits and offers greater flexibility than AWM and ReasoningBank. The new benchmark, EnterpriseOps-Gym (Malay et al., 2026), targets tool use in enterprise settings across 8 domains (*e.g.*, calendar, email, HR). We evaluate two models (`GPT-OSS-120B` and `GPT-5.4-mini`) in this setting and report the results in Table 13.

Like AWM and ReasoningBank, this third self-improving method does not consistently improve performance. It helps in some domains but hurts in others, and the overall success rate drops slightly with memory enabled (24.5→22.3 for `GPT-OSS-120B`; 30.5→29.7 for `GPT-5.4-mini`). Memory also amplifies variance in the majority of cases: in 10 of 16 cases (2 models × 8 domains), the with-memory configuration shows larger variance than the no-memory baseline. Most notably, in the drive domain with `GPT-OSS-120B`, the best–worst gap increases from 4.7 to 14.1 points once memory is enabled.

[5] `letta-ai/letta-code`

F PROMPTS

Listing 1: Memory construction prompt used in AWM. The prompt is obtained from the codebase of ASI (Wang et al., 2025a).

````
You are a proficient software engineer. Your task is to (1)
    summarize reusable workflows from the provided action
    trajectories. Each workflow should be a textual guideline to
    solve a task.
A workflow should contain the task described in natural language,
    and an action trajectory to solve the task. The action
    trajectory should contain a sequence of steps, each with an
    action and a textual description.

An example workflow look like this:
"""
Task: Tell me who has made the most contributions, in terms of
    number of commits, to the primer/design project
Action Trajectory:
```click("505")```  # open the `Primer / design` repository page
```click("642")```  # click on the "Commits" link
```click("307")```  # click on the "Contributors" link
```stop("The top 3 contributors to the 'prime/design' repo are: 1.
     Shawn Allen with 95 commits, 2. Inayaili León with 77 commits,
     3. Aurora Pleguezuelo with 66 commits.")```  # This page
    contains the information required to determine the top 3
    contributors to the `prime/design` repository, based on the
    number of commits. I will now extract and send the top 3
    contributors based on the number of commits from the current
    page.

"""

Do not include the answer or example-specific information in the
    workflows.
Make sure each workflow has no less than 2 steps and no more than
    5 steps; to keep the workflow simple and task-oriented.
You can generate zero or one workflow depending on the provided
    examples.

Particularly for `click` actions, either put exact element ids
    like `click('120')` or use a variable placeholder like `click({
    element_id})`. Do NOT put text strings in such as `click('
    Reports')`.

[Two additional example workflows were included in the experiments
     but skipped here.]

```
````
````

Listing 2: Memory construction prompt used in ReasoningBank (successful tasks).

```
You are an expert in web navigation. You will be given a user
    query, the corresponding trajectory that represents how an
    agent successfully accomplished the task.

## Guidelines

You need to extract and summarize useful insights in the format of
     memory items based on the agent's successful trajectory. The
    goal of summarized memory items is to be helpful and
    generalizable for future similar tasks.

## Important notes
- You must first think why the trajectory is successful, and then
    summarize the insights.
- You can extract at most 3 memory items from the trajectory.
- You must not repeat similar or overlapping items.
- Do not mention specific websites, queries, or string contents,
    but rather focus on the generalizable insights.

## Output Format
Your output must strictly follow the Markdown format shown below:

```
Memory Item i
Title: <the title of the memory item>
Description: <one sentence summary of the memory item>
Content: <1-3 sentences describing the insights learned to
    successfully accomplishing the task>
```
```

Listing 3: Memory construction prompt used in ReasoningBank (failed tasks).

```
You are an expert in web navigation.
You will be given a user query, the corresponding trajectory that
    represents how an agent attempted to resolve the task but
    failed.

## Guidelines

You need to extract and summarize useful insights in the format of
     memory items based on the agent's failed trajectory.
The goal of summarized memory items is to be helpful and
    generalizable for future similar tasks.

## Important notes
- You must first reflect and think why the trajectory failed, and
    then summarize what lessons you have learned or strategies to
    prevent the failure in the future.
- You can extract at most 3 memory items from the trajectory.
- You must not repeat similar or overlapping items.
- Do not mention specific websites, queries, or string contents,
    but rather focus on the generalizable insights.

## Output Format

Your output must strictly follow the Markdown format shown below:
```
Memory Item i
Title: <the title of the memory item>
Description: <one sentence summary of the memory item>
Content: <1-3 sentences describing the insights learned to
    successfully accomplishing the task>
```
```

G FAILURE MODES

Table 14: **Example of Ambiguous Task Query: WebArena Task 118.** The task is intended to be a web navigation task but is interpreted as a question answering task by the agent.

Task Query
I have jaw bruxism problem, show me something that could alleviate the problem.
Ground-Truth Evaluator
"Jaw bruxism" and "mouth guard" must be included in the content of the final page.
Agent's Final Message
(The agent didn't browse the shopping website and ends the task after 1 step.) Here are practical things that can help reduce jaw bruxism (teeth grinding/clenching). This is general information — consult your dentist or doctor for personalized advice: 1) Night guard / oral splint - Best: a custom-fitted occlusal guard from your dentist (protects teeth and reduces muscle stress). [text omitted for brevity...]
Agent's Memory
Title: Always gather targeted patient/context details before giving medical guidance **Description:** Before offering specific treatments for a health problem, ask concise clarifying questions about symptoms and context. **Content:** Ask whether the problem occurs during sleep or wakefulness, severity/frequency, current treatments, dental history, relevant medications, pregnancy, and presence of red-flag symptoms (severe pain, tooth damage, jaw locking, daytime sleepiness). These details let you tailor recommendations (e.g., conservative self-care vs. urgent dental referral) and avoid unsafe suggestions.

Table 15: **Example of Evaluator Bug: WebArena Task 329.** A correct answer may fail strict string-matching rubric, and further lead to the creations of irrelevant memories.

Task Query
How much I spend on 4/19/2023 on shopping at One Stop Market?
Ground-Truth Evaluator
The answer string must include "0".
Agent's Final Message
I could not find any orders dated 4/19/2023 in your One Stop Market account. I scanned the full order history (37 orders shown after setting per-page to 50) and the extraction returned no matches for 4/19/2023. Therefore, according to this account, the amount spent on 4/19/2023 at One Stop Market is $0.00.
Author Comment
"0.00" is processed as one token and is considered different from "0" according to WebArena evaluator.
Agent's Memory
Title: Consider alternate accounts and guest/third-party purchases **Description:** Missing orders in one account can result from purchases made under a different account, guest checkout, or through a different platform. **Content:** If an order date isn't found, check other user accounts, alternate email addresses or phone numbers, loyalty profiles, and third-party marketplaces or apps the user might have used.

Table 16: **Example of Using Haversine Formula: WebArena Task 38.** The agent receives an reward for including "yes" in its final answer, while the strategy to reach the answer is unintended.

Task
Check if Duquesne University in Pittsburgh can be reached in one hour by car from Pittsburgh Airport.
Rubric
Answer string must include "yes".
Agent's Answer
Result and conclusion: *What I could not do:* The OpenStreetMap directions page did not show a numeric route or duration because it displayed messages indicating JavaScript was not available, so a live OSRM route summary could not be extracted. *Fallback method used:* Coordinates embedded in the page URL were used (origin: 40.4918, -80.2370 — Pittsburgh International Airport; destination: 40.4365, -79.9905 — Duquesne University area). Great-circle (haversine) distance was computed, then a conservative routing factor was applied to estimate driving distance and time. *Calculations and assumptions:* • Straight-line distance (haversine): ≈ 21.7 km (≈ 13.5 miles). • Estimated driving distance (routing factor ≈ 1.3): $\approx$ 28–29 km ($\approx$ 17.5–18 miles). • Estimated driving time using typical effective speeds: 60 mph $\rightarrow \sim$ 17–18 min; 40 mph $\rightarrow \sim$ 26–28 min; heavy congestion at ≈ 30 mph $\rightarrow \sim$ 35–40 min. • Conservative upper bound: under most circumstances, the drive remains well under 60 minutes; only extreme incidents may push it beyond one hour. *Final answer:* Yes — Duquesne University can be reached by car from Pittsburgh International Airport in under one hour under normal and moderately heavy traffic conditions (typical estimate: $\sim$ 20–40 minutes). For exact current travel time, enable JavaScript on the directions page or use a live routing API (OSRM/Google Maps/Waze). *Method summary (reproducibility):* Used URL coordinates (40.4918,-80.2370 $\rightarrow$ 40.4365,-79.9905), computed haversine distance (≈ 21.7 km), applied a 1.3 routing multiplier ($\approx$ 28–29 km), and converted distance to time ranges using reasonable average speeds.
Agent's Memory
Title: Use URL or page coordinates as a robust routing fallback. **Description:** If the directions UI or routing engine fails to render, extract coordinates from the page or URL and compute an offline route estimate. **Content:** Parse origin and destination latitudes and longitudes embedded in the page or URL, compute straight-line (haversine) distance, and convert it to plausible driving distance using a routing multiplier (e.g., $\sim$ 1.2–1.4). This provides a reliable fallback when live routing is unavailable.