REASONINGBANK: SCALING AGENT SELF-EVOLVING WITH REASONING MEMORY

Anonymous authors
Paper under double-blind review

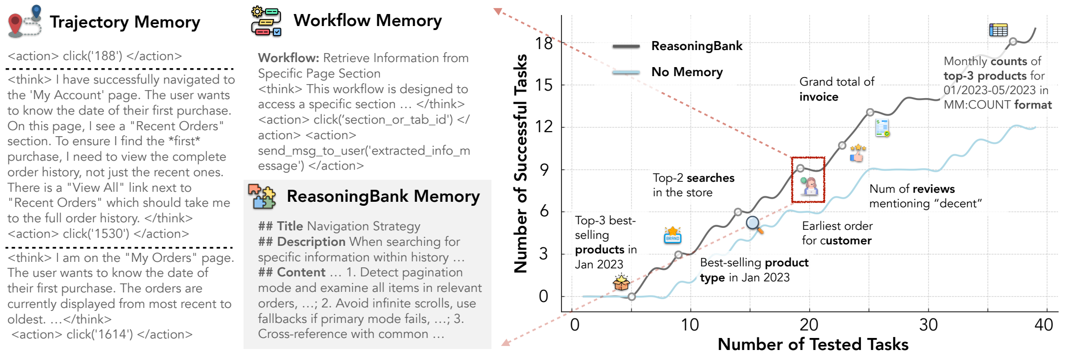
ABSTRACT

With the growing adoption of large language model (LLM) agents in persistent, real-world roles, they naturally encounter continuous streams of tasks and interactions. A key limitation, however, is their failure to learn from the accumulated experiences, forcing them to discard valuable insights and repeat past errors. Unlike prior works that primarily store raw experience or successful routines, we propose REASONINGBANK, a novel memory framework that allows an agent to self-curate generalizable reasoning strategies from both its successful and failed experiences for future use. This mechanism enables agents to generalize across tasks and become more capable over time. To accelerate and diversify this test-time learning process, we further propose memory-aware test-time scaling (MATTS), which leverages a powerful synergy between memory and test-time scaling. On one hand, relevant memory from REASONINGBANK guides the scaling process toward more effective exploration and improved reliability. On the other, scaling — in both parallel and sequential settings — generates abundant, diverse experiences that provide rich contrastive signals for synthesizing higher-quality memory. Experiments on web browsing and software engineering benchmarks show that REASONINGBANK consistently outperforms existing memory mechanisms in both effectiveness and efficiency, with MATTS further amplifying the gains. These findings position memory-driven experience as a new dimension of test-time scaling, where emergent behaviors naturally arise and agents acquire self-evolving capabilities.

1 INTRODUCTION

The rapid advancement of large language models (LLMs) has significantly accelerated the development of interactive LLM agents (Wang et al., 2024; Liu et al., 2025a), which are crucial in tackling complex real-world tasks that require multi-turn interactions with environments. These agents have demonstrated great potential across diverse scenarios, including web browsing (Gur et al., 2024), computer use (Yang et al., 2024; Xie et al., 2024), and scientific discovery (Ghafarollahi & Buehler, 2025). As these agents are increasingly deployed in persistent, long-running roles, they naturally encounter a continuous stream of tasks and interactions. However, a critical limitation persists: they largely fail to learn from this accumulated experience. By approaching each new task in isolation, they are doomed to (i) repeat similar errors observed in the past (Yin et al., 2025), (ii) discard valuable insights gained from related problems, and, most importantly, (iii) lack self-evolving capabilities that make the agent system more capable over time (Gao et al., 2025). This phenomenon highlights the necessity of building memory-aware agent systems that could learn from their past experiences (Zhang et al., 2024b).

Recent efforts on agent memory have primarily focused on storing past interactions for reuse (Zhao et al., 2024; Tang et al., 2025b; Chen et al., 2025; Sun et al., 2025). While useful, these approaches are often limited to leveraging raw trajectories (Zheng et al., 2024; Kagaya et al., 2024; Kong et al., 2025) or common, successful routines (i.e., workflows, procedures) (Wang et al., 2025d; Fang et al., 2025). These approaches suffer from two fundamental drawbacks. First, they lack the ability to distill higher-level, transferable reasoning patterns. Second, by over-emphasizing successful experiences, it leaves the valuable lessons from an agent's own failures largely underexplored (Zhang et al., 2024a). Consequently, existing memory designs often remain limited to passive record-keeping rather than providing actionable, generalizable guidance for future decisions.



Trajectory Memory

<action> click('188') </action>

<think> I have successfully navigated to the 'My Account' page. The user wants to know the date of their first purchase. On this page, I see a "Recent Orders" section. To ensure I find the *first* purchase, I need to view the complete order history, not just the recent ones. There is a "View All" link next to "Recent Orders" which should take me to the full order history. </think>
<action> click('1530') </action>

<think> I am on the "My Orders" page. The user wants to know the date of their first purchase. The orders are currently displayed from most recent to oldest. …</think>
<action> click('1614') </action>

Workflow Memory

Workflow: Retrieve Information from Specific Page Section
<think> This workflow is designed to access a specific section … </think>
<action> click('section_or_tab_id') </action> <action> send_msg_to_user('extracted_info_message') </action>

ReasoningBank Memory

## Title Navigation Strategy
## Description When searching for specific information within history …
## Content … 1. Detect pagination mode and examine all items in relevant orders, …; 2. Avoid infinite scrolls, use fallbacks if primary mode fails, …; 3. Cross-reference with common …

ReasoningBank
No Memory
Number of Successful Tasks
Number of Tested Tasks
0
3
6
9
12
15
18
0
10
20
30
40
Top-3 best-selling products in Jan 2023
Best-selling product type in Jan 2023
Top-2 searches in the store
Earliest order for customer
Grand total of invoice
Num of reviews mentioning "decent"
Monthly counts of top-3 products for 01/2023-05/2023 in MM:COUNT format

Figure 1: REASONINGBANK induces reusable reasoning strategies, making memory items more transferrable for future use. This enables agents to continuously evolve and achieve higher accumulative success rates than the "No Memory" baseline on the WebArena-Admin subset.

Motivated by this, we propose REASONINGBANK, a novel memory framework for agent systems that distills and organizes memory items from *both successful and failed experiences*. REASONINGBANK focuses on capturing high-level reasoning insights, enabling agents to generalize across tasks and adapt to dynamic environments more effectively as shown in Figure 1. When facing a new query, the agent retrieves relevant memory items from REASONINGBANK and integrates them into its inference process; after execution, new experiences are distilled and consolidated back into REASONINGBANK. This closed-loop design continuously enables the agent to evolve and discover new strategies for unseen tasks.

As a strong experience learner, REASONINGBANK naturally benefits more from diverse experiences, making scaling a principled way to unlock greater gains. To this end, we introduce memory-aware test-time scaling (MATTS) in both parallel and sequential settings, where scaling supplies diverse contrastive experiences that enrich the quality and generality of REASONINGBANK's memory. Moreover, we reveal **a synergy between memory and test-time scaling**: while REASONINGBANK provides guidance that steers scaling towards sampling more promising trajectories (termed as "*rollouts*"), and the diverse rollouts generated during scaling in turn provide abundant experience sources with valuable contrastive signals for memory curation. This forms a positive feedback loop and positions memory-driven experience as a new dimension of test-time scaling where agents acquire self-evolving capabilities.

Our contributions are threefold: (1) We propose REASONINGBANK, a novel memory framework that distills generalizable reasoning strategies from both successful and failed experiences, beyond prior work that primarily stores raw trajectories or success-only routines. (2) We introduce MATTS that establishes a powerful, bidirectional synergy between memory and test-time scaling, with memory-driven experience as a new scaling dimension. (3) We conduct extensive experiments on web browsing (WebArena, Mind2Web) and software engineering (SWE-Bench-Verified) tasks. We demonstrate that our approaches not only outperform existing methods in effectiveness (up to 24.1% relative improvement) and efficiency (18.0% less interactions), but also uniquely learn from failures and enable agents to develop increasingly complex, emergent reasoning strategies over time.

2 RELATED WORK

Memory for LLM Agents. Memory has emerged as an essential module in modern agent systems (Zhang et al., 2024b), with prior efforts primarily emphasizing personalization and long-context management (Xu et al., 2022; Maharana et al., 2024; Wu et al., 2025; Hu et al., 2025). Memory representations have been organized in various forms, including latent knowledge embeddings (Wang et al., 2025b) and structured graphs (Xu et al., 2025; Chhikara et al., 2025; Packer et al., 2023). Beyond memory content, reflective mechanisms such as RMM (Tan et al., 2025) develop both backward- and forward-looking strategies, while MemoryBank (Zhong et al., 2024) introduces a novel updating mechanism. More recently, with the growing integration of reinforcement learning (RL) in LLM agents, memory has also been leveraged for managing contextual information across interactions (Yu et al., 2025a; Zhou et al., 2025). This paper falls in the research line of learning from past experiences as memory. Different from previous works that emphasize reusing successful trajectories (Zheng et al., 2024; Tang et al., 2025a) or procedural workflows (Wang et al., 2025d; Qian et al., 2024; Fang et al., 2025; Liu et al., 2025b), ReasoningBank stores high-level strategies and reasoning hints. By abstracting experiences into reusable reasoning units, ReasoningBank enables agents to generalize not only from successful cases but also by learning from failures, thereby providing richer guidance for test-time learning. Additionally, we are the first to explore memory-aware test-time scaling, where ReasoningBank synergistically work with diverse signals from abundant exploration trajectories.

Agent Test-Time Scaling. Test-time scaling (TTS) (Snell et al., 2025) has demonstrated strong effectiveness and has become a widely adopted practice in end-to-end problem-solving such as coding (Li et al., 2025; Yu et al., 2025c) and math reasoning (Muennighoff et al., 2025), where methods including best-of-N (Chow et al., 2025), beam search (Wu et al., 2024b), and leveraging verifiers (Setlur et al., 2025) are commonly employed. However, its application to multi-turn interactive scenarios, particularly agentic tasks, remains underexplored. Existing works mainly adapt the lesson learned from reasoning tasks (Zhu et al., 2025b) and scale different dimensions of agentic systems, including the search space for each action (Yu et al., 2025b), the number of agents in multi-agent systems (Jin et al., 2025), and the number of interactions with the environment (Shen et al., 2025). We found that none of these efforts considers the role of *agent memory* in scaling, where an agent can learn from past experiences to guide future decisions. Our work extends this line of research by introducing memory-aware test-time scaling (MaTTS). As we will show in our empirical results (§4.3 and §4.4), memory offers benefits beyond mere computational scaling, where memory and scaling synergistically work towards better performance.

3 Methodology

In this section, we introduce the problem setup (§3.1), and present our proposed ReasoningBank (§3.2), based on which we further develop memory-aware test-time scaling (MaTTS) (§3.3).

3.1 Problem Formulation

Agent Configuration. The scope of this work focuses on language model (LM)-based agents. The agent policy $\pi_{\mathcal{L}}(\cdot|\mathcal{M}, \mathcal{A})$ is parameterized by the backbone LLM $\mathcal{L}$, conditioned on a memory module $\mathcal{M}$, and the action space $\mathcal{A}$, denoted as $\pi_{\mathcal{L}}$ for short. The agent needs to perform a task via interacting with the environment, which can be viewed as a sequential decision-making process. Formally, the transition function of the environment is defined as $\mathcal{T}(s_{t+1}|s_t, a_t)$ where s_t is the state and a_t is the action selected by $\pi_{\mathcal{L}}$ at time t. We focus on web browsing and software engineering (SWE) tasks. $\mathcal{A}$ is a set of web navigation operations for web browsing and bash commands for SWE, $\mathcal{M}$ is ReasoningBank and initialized as empty. For each given task, the agent generates a trajectory of $(o_{0:t}, a_{0:t})$ for t steps, where observation o_t is from the current state s_t. Observations are text-based accessibility tree of web pages[1] for web browsing tasks and code snippets for SWE. The agent needs to generate an action $a_{t+1} \in \mathcal{A}$ via $\pi_{\mathcal{L}}(o_{0:t}, a_{0:t}; \mathcal{M}, \mathcal{A}) \rightarrow a_{t+1}$. For implementation, the memory module $\mathcal{M}$ contributes relevant memories as additional system instruction for $\pi_{\mathcal{L}}$.

Test-Time Learning. We focus on the test-time learning paradigm (Wu et al., 2024a; Wang et al., 2025c) where a sequence of task queries $\mathcal{Q} = \{q_1, q_2, ..., q_N\}$ arrives in a streaming fashion, i.e., each query is revealed and must be completed sequentially without access to future ones. In this setting, no ground truth is available during test-time, so the agent must continually *evolve* by only leveraging its own past trajectories and any self-verification without relying on external labels. This streaming setting highlights two key challenges: (i) how to extract and preserve useful memory from past trajectories, and (ii) how to effectively leverage such memory for future queries to avoid redundantly re-discovering already successful strategies or repeating past mistakes.

3.2 ReasoningBank

Past raw trajectories (or experiences), while being comprehensive and original, are often too lengthy and noisy to be directly applied to the current user query. As illustrated in Figure 2,

[1]We use the thinking process of $\pi_{\mathcal{L}}$ as the approximation of $o_{0:t}$ due to lengthy observation representations following Wang et al. (2025d).

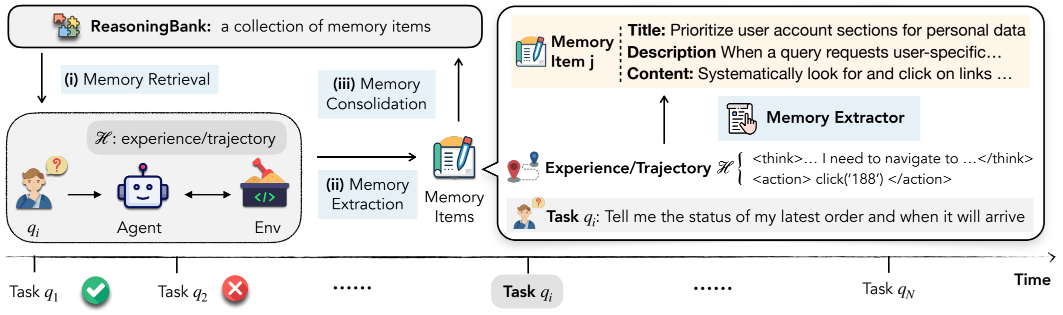

ReasoningBank: a collection of memory items
(i) Memory Retrieval
(iii) Memory Consolidation
$\mathcal{H}$: experience/trajectory
q_i
Agent
Env
(ii) Memory Extraction
Memory Items
Memory Item j
Title: Prioritize user account sections for personal data
Description When a query requests user-specific…
Content: Systematically look for and click on links …
Memory Extractor
Experience/Trajectory $\mathcal{H}$
<think>… I need to navigate to …</think>
<action> click('188') </action>
Task q_i: Tell me the status of my latest order and when it will arrive
Task q_1
Task q_2
Task q_i
Task q_N
Time

Figure 2: Overview of REASONINGBANK. Experiences are distilled into structured memory items with a title, description, and content. For each new task, the agent retrieves relevant items to interact with the environment, and constructs new ones from both successful and failed trajectories. These items are then consolidated into REASONINGBANK, forming a closed-loop memory process.

REASONINGBANK distills useful strategies and reasoning hints from past experiences into structured memory items, which are then stored in the agent's memory for future reuse.

Memory Schema. Memory items in REASONINGBANK are designed and induced from past experiences as structured knowledge units that abstract away low-level execution details while preserving transferrable reasoning patterns and strategies. Each memory item specifies three components: (i) a *title*, which serves as a concise identifier summarizing the core strategy or reasoning pattern; (ii) a *description*, which provides a brief one-sentence summary of the memory item; and (iii) the *content*, which records the distilled reasoning steps, decision rationales, or operational insights extracted from past experiences. Together, memory items extracted are both human-interpretable and machine-usable, facilitating efficient usage and integration with agents.

Integrating REASONINGBANK with Agents. An agent $\pi_{\mathcal{L}}$ equipped with REASONINGBANK can draw upon a curated pool of transferable strategies to guide decision-making. This enables the agent to recall effective insights, avoid previously observed pitfalls, and adapt more robustly to unseen queries. The integration proceeds in three steps: (i) *memory retrieval*, (ii) *memory construction*, and (iii) *memory consolidation*, as shown in Figure 2. During *memory retrieval*, the agent queries REASONINGBANK with the current query context to identify the top-k relevant experiences and their corresponding memory items using embedding-based similarity search. Retrieved items are injected into the agent's system instruction, ensuring that action prediction from $\mathcal{A}$ is grounded with useful past experiences. When the current query task is completed, we will perform *memory construction* to extract new memory items. The first step is to obtain correctness signals for completed trajectories: we adopt an LLM-as-a-judge (Gu et al., 2024) to label outcomes as success or failure given the query and trajectory. Based on these signals, we apply different extraction strategies: successful experiences contribute validated strategies, while failed ones supply counterfactual signals and pitfalls that help sharpen guardrails. In practice, we extract multiple memory items for each trajectory/experience as detailed in Appendix A.1. Finally, *memory consolidation* incorporates these items into REASONINGBANK with a simple addition operation, maintaining an evolving repository of memory items. Details are in Appendix A.2. Together, these steps form a closed-loop process: the agent leverages past experiences, constructs new memory from current tasks, and continually updates its memory, enabling sustained evolvement in test-time learning scenarios.[2]

3.3 MATTS: MEMORY-AWARE TEST-TIME SCALING

REASONINGBANK enables learning from experiences to translate more experiences into greater improvements. As test-time scaling (TTS) (Snell et al., 2025) recently emerged as a powerful strategy for boosting the performance of LLM agents (Zhu et al., 2025a), it shows strong potential by allocating additional inference-time computation to generate abundant exploration histories. A vanilla combination is depicted in Figure 3(a), where more trajectories are independently converted to more memory items. However, this is suboptimal because it does not leverage inherent contrastive signal that arises from abundant explorations on the same problem, which limits the

[2]We deliberately keep the memory usage pipeline simple, avoiding additional complexity in retrieval or consolidation so as to highlight the contribution of REASONINGBANK itself. These components, however, can be further enhanced with more sophisticated techniques, which could provide additional benefits.

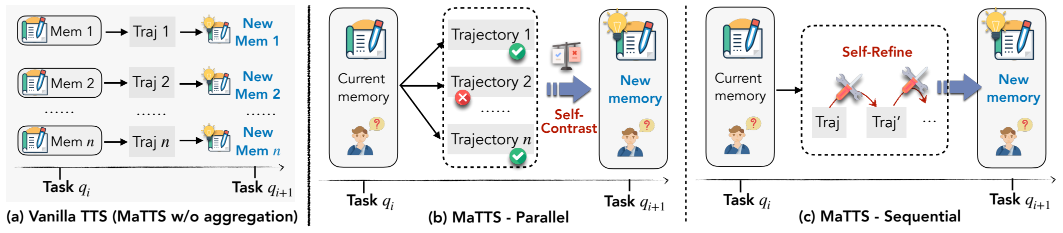


Figure 3: Comparison of *(a) vanilla TTS* that independently run REASONINGBANK for multiple times without aggregating on trajectories, MATTS with *(b) parallel scaling*, where self-contrast across multiple trajectories curates reliable memory, and *(c) sequential scaling*, where self-refinement enriches memory with intermediate reasoning signals.

resulting performance advantage brought by TTS. To address this, we propose *Memory-aware Test-Time Scaling* (MATTS), a novel integration of test-time scaling with our memory framework, REASONINGBANK. Unlike the vanilla approach, MATTS learns by deliberately aggregating from the abundant successful and failure trajectories generated during scaling for more effective memory curation. We design two complementary instantiations for MATTS, parallel scaling and sequential scaling, as illustrated in Figure 3(b) and 3(c) with detailed implementation in Appendix A.3.

Parallel Scaling. In the parallel setting, we generate multiple trajectories for the same query under the guidance of retrieved memory items. By comparing and contrasting (*self-contrast* (Chen et al., 2020)) *across different trajectories*, the agent can identify consistent reasoning patterns while filtering out spurious solutions. This process enables more reliable memory curation from multiple trials of a single query and promotes diverse yet grounded exploration.

Sequential Scaling. We iteratively refine its reasoning *within a single trajectory* after the initial completion, following the principle of *self-refinement* (Madaan et al., 2023). In this process, the intermediate notes generated in self-refinement are also used as valuable signals for memory, since they capture reasoning attempts, corrections, and insights that may not appear in the final solution.

We define the scaling factor k, denoting the number of trajectories for parallel scaling and refinement steps for sequential scaling. Equipped with REASONINGBANK, both parallel and sequential strategies become memory-aware, ensuring that the additional computation allocated at test time translates into more transferable and higher-quality memory for future tasks.

4 EXPERIMENTS

4.1 SETUP

Following existing work (Wang et al., 2025d), we conduct experiments on WebArena (Zhou et al., 2024) which features general web navigation across diverse domains,[3] and Mind2Web (Deng et al., 2023) that tests generalization of agents on versatile operations and environments. We also conduct experiment on SWE-Bench-Verified for repository-level issue-resolving. For comparison, we consider baselines ranging from memory-free agents (No Memory) to trajectory-based memory (Synapse) (Zheng et al., 2024) and workflow-based memory (AWM) (Wang et al., 2025d). Our agents are built on Gemini-2.5 (Comanici et al., 2025) and Claude-3.7 (Anthropic, 2025) models using BrowserGym (de Chezelles et al., 2025) for web browsing and bash-only for SWE, following ReAct (Yao et al., 2023) style with default decoding configurations. We evaluate effectiveness (success rate, SR) and efficiency (average steps, AS), with specific metrics varying for each dataset. Full descriptions for datasets, baselines, implementations, and evaluation are in Appendix B.

4.2 RESULTS OF REASONINGBANK

Tables 1, 2, 3 summarize the main evaluation results of REASONINGBANK on WebArena, Mind2Web, and SWE-Bench-Verified accordingly. We have the following observations.

REASONINGBANK consistently outperforms baselines across LLM backbones on all datasets. Specifically, REASONINGBANK improves the overall success rate on WebArena (Table 1) by +8.3,

[3]We exclude the domain of *Map* due to website issues following Miyai et al. (2025) for a fair comparison.

Table 1: Experiment results of REASONINGBANK on WebArena benchmark. Success rate (SR ↑) and average number of steps (AS ↓) are reported on 5 subsets for 3 different backbone LLMs.

Models	**Shopping**		**Admin**		**Gitlab**		**Reddit**		**Multi**		**Overall**	
	SR	*AS*	*SR*	*AS*	*SR*	*AS*	*SR*	*AS*	*SR*	*AS*	*SR*	*AS*
Gemini-2.5-flash												
No Memory	39.0	8.2	44.5	9.5	33.9	13.3	55.7	6.7	10.3	10.0	40.5	9.7
Synapse	40.6	7.0	45.1	9.1	35.6	13.0	59.4	6.5	10.3	10.5	42.1	9.2
AWM	44.4	7.0	46.7	8.8	37.2	13.2	62.3	6.1	3.4	**7.7**	44.1	9.0
REASONINGBANK	**49.7**	**6.1**	**51.1**	**8.2**	**40.6**	**12.3**	**67.0**	**5.6**	**13.8**	8.8	**48.8**	**8.3**
Gemini-2.5-pro												
No Memory	45.5	7.6	51.1	8.7	35.0	11.6	71.7	6.0	6.9	8.8	46.7	8.8
Synapse	46.5	6.6	52.2	8.9	38.3	11.3	68.9	5.9	6.9	9.0	47.7	8.5
AWM	48.1	6.4	49.3	9.8	40.0	11.2	68.9	6.4	3.4	9.3	47.6	8.7
REASONINGBANK	**51.9**	**6.0**	**56.6**	**7.7**	**44.4**	**9.8**	**80.2**	**5.1**	**13.8**	**8.2**	**53.9**	**7.4**
Claude-3.7-sonnet												
No Memory	38.5	6.1	49.5	8.4	36.7	10.6	53.8	5.5	0.0	11.6	41.7	8.0
Synapse	39.6	5.8	50.5	8.5	38.0	10.0	53.8	6.1	0.0	11.8	42.6	7.9
AWM	39.6	7.2	47.8	9.3	34.6	10.9	52.8	7.0	0.0	12.4	40.8	8.9
REASONINGBANK	**44.9**	**5.6**	**53.3**	**7.6**	**41.1**	**9.5**	**57.5**	**5.2**	**3.4**	**10.5**	**46.3**	**7.3**

+7.2, and +4.6 with three different backbone LLMs compared to memory-free agents. A similar pattern holds on Mind2Web (Table 3), where REASONINGBANK delivers clear gains across cross-task, cross-website, and cross-domain settings, underscoring both the consistency and scalability of its benefits across datasets and model sizes. Results on SWE-Bench-Verified (Table 2) further confirm its robustness. Crucially, unlike baselines such as Synapse and AWM that rely on a narrow, homogeneous memory source derived exclusively from successful trajectories, REASONINGBANK employs a superior extraction strategy that is key to its consistent outperformance.

REASONINGBANK enhances generalization with better transferrable memory across tasks. We also evaluate in challenging generalization settings. On WebArena (Table 1), the *Multi* subset requires transferring memory across multiple websites, where REASONINGBANK achieves a notable gain of +4.6 averaged SR over the strongest baseline. In contrast, strong baselines such as AWM fail to provide gains and even degrade in this setting. On Mind2Web (Table 3), which includes cross-task, cross-website, and cross-domain evaluations that impose progressively higher demands, REASONINGBANK consistently improves success rates. The gains are especially pronounced in the cross-domain setting, which requires the highest level of generalization. These results demonstrate that memory curated by REASONINGBANK is more robust and transferable, enabling agents to generalize effectively across diverse scenarios.

Table 2: Experiment results of REASONINGBANK on SWE-Bench-Verified dataset for issue-resolving in a given repository.

Methods	Resolve Rate	AS
Gemini-2.5-flash		
No Memory	34.2	30.3
Synapse	35.4	30.7
REASONINGBANK	**38.8**	**27.5**
Gemini-2.5-pro		
No Memory	54.0	21.1
Synapse	53.4	21.0
REASONINGBANK	**57.4**	**19.8**

REASONINGBANK achieves superior efficiency by leveraging past experiences as memory. In addition to higher success rates, REASONINGBANK also reduces the number of interaction steps needed to complete tasks, as shown in the *Step* metric of Table 1 and 2. On WebArena, across almost all subsets and backbones, REASONINGBANK lowers the average step count by up to 1.4 compared with "No Memory", and 1.6 compared with other memory baselines. The average step on SWE-Bench-Verified is also smaller by saving 2.8 and 1.3 steps respectively. This indicates that REASONINGBANK enables agents to solve tasks more efficiently by reusing and refining reasoning knowledge, thus avoiding unnecessary or redundant exploration.

4.3 RESULTS OF MATTS

We experimented MATTS with Gemini-2.5-flash on Webarena-Shopping subset. By default, MATTS integrates REASONINGBANK, but it could also use other memory mechanisms. To

Table 3: Results on Mind2Web benchmark for cross-task, cross-website, and cross-domain generalization test. EA (↑) is short for element accuracy, AF$_1$ (↑) is short for action F$_1$, and SSR (↑) is short for step success rate. SR (↑) is the task-level success rate measuring if all steps are correct.

Models	*Cross-Task*				*Cross-Website*				*Cross-Domain*			
	EA	*AF$_1$*	*SSR*	*SR*	*EA*	*AF$_1$*	*SSR*	*SR*	*EA*	*AF$_1$*	*SSR*	*SR*
Gemini-2.5-flash												
No Memory	46.0	59.1	40.3	3.3	39.8	45.1	31.7	1.7	35.8	37.9	31.9	1.0
Synapse	47.0	59.5	41.2	3.5	40.3	46.0	32.1	1.9	36.3	38.5	32.4	1.1
AWM	46.3	56.1	41.0	3.5	39.1	42.2	31.7	2.1	33.3	36.5	30.1	0.7
ReasoningBank	**52.1**	**60.4**	**44.9**	**4.8**	**44.3**	**52.6**	**33.9**	**2.3**	**40.6**	**41.3**	**36.6**	**1.6**
Gemini-2.5-pro												
No Memory	49.3	60.2	44.4	3.5	41.2	49.8	34.8	3.4	37.9	37.7	35.0	1.4
Synapse	50.1	61.0	44.7	3.6	41.8	51.2	35.0	3.2	38.5	39.8	35.6	1.5
AWM	48.6	61.2	44.4	3.7	41.9	47.9	34.8	2.3	37.3	38.1	34.4	1.2
ReasoningBank	**53.6**	**62.7**	**45.6**	**5.1**	**46.1**	**54.8**	**36.9**	**3.8**	**42.8**	**45.2**	**38.1**	**1.7**

investigate the overall scaling effect, we benchmark with **(i) MaTTS w/o memory**, which represents the scaling setting without memory mechanism, **(ii) MaTTS w/o aggregation**, which is equal to Vanilla TTS in Figure 3(a) and **(iii) MaTTS** to demonstrate the effect with respect to scaling factor k. Notably, $k = 1$ is the setting without scaling. For parallel scaling, we compute Best-of-N (BoN) as the final metric detailed in Appendix A.3. Results are shown in Figure 4.

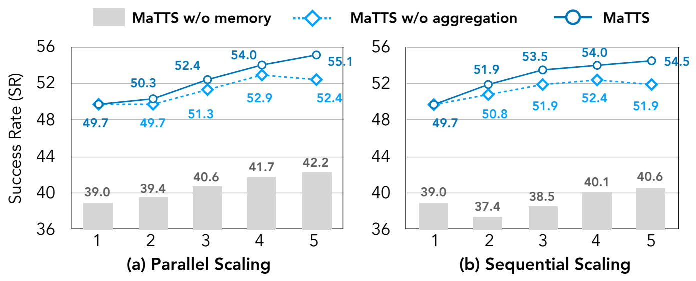


Figure 4: Effect of scaling factor k for MaTTS under with ReasoningBank on WebArena-Shopping subset. We compare (a) parallel and (b) sequential test-time scaling.

Both parallel scaling and sequential scaling boost performance. Increasing k generally improves success rate, confirming the benefit of allocating more inference-time computation. With MaTTS, parallel scaling grows from 49.7 ($k = 1$) to 55.1 ($k = 5$), while sequential scaling rises from 49.7 to 54.5. For the baseline of MaTTS w/o memory, the gains are smaller and less consistent (e.g., parallel scaling fluctuates between 39.0 and 42.2, sequential between 37.4 and 40.6). In contrast, MaTTS enables stronger and more stable improvements across both scaling strategies, highlighting its role in making scaling more effective.

MaTTS is consistently better than vanilla TTS. With ReasoningBank, MaTTS consistently surpasses MaTTS w/o aggregation (vanilla TTS), showing that memory-aware coordination and aggregation is important. Specifically, at $k = 5$, MaTTS achieves 55.1 in parallel scaling compared with 52.4 for vanilla TTS, and 54.5 versus 51.9 in sequential scaling. These improvements highlight that memory-aware scaling effectively directs the agent toward more promising solutions by synthesizing insights from multiple trajectories or interaction steps to leverage contrastive signals.

Sequential scaling shows short-term advantage, but parallel dominates at larger scales for ReasoningBank. With stronger memory mechanisms such as ReasoningBank, sequential refinement brings higher gains at small k, but its benefit quickly saturates—once the model either succeeds or fails decisively, further refinements add little new insight. In contrast, parallel scaling continues to provide diverse rollouts that allow the model to critique and improve upon its own generations, leading it to surpass sequential at larger k (e.g., 55.1 vs. 54.5 at $k = 5$). In contrast, for vanilla TTS which is not equipped with memory module, sequential scaling yields little or even no benefit as scaling goes on, and parallel scaling consistently dominates.

4.4 Synergy of Memory and Test-Time Scaling

While the previous section establishes the overall effectiveness of MaTTS, we highlight the synergy between memory and TTS in this section. Figure 5 presents a snapshot of MaTTS on the WebArena-

Shopping subset with parallel scaling factor $k = 3$, where we report both Pass@1 (randomly selected trajectory) and Best-of-3 (BoN). This setting allows us to examine the bidirectional interaction between memory quality and scaling effectiveness.

Better memory enables stronger test-time scaling performance. To see how memory improves the effectiveness of scaling, we focus on the BoN results, which directly measures an agent's ability to surface the best outcome among multiple rollouts. As shown by blue bars in Figure 5, the benefit of scaling depends critically on the underlying memory. Without memory, scaling yields slight improvement, with BoN rises only from 39.0 to 40.6. Weaker memory mechanisms such as Synapse and AWM provide moderate gains, reaching 42.8 and 45.5, respectively. In contrast, MaTTS with ReasoningBank delivers the strongest benefit, with BoN climbing from 49.7 to 52.4. These results show that high-quality memory directs scaling toward more promising rollouts, ensuring that the additional trajectories are not wasted but converted into higher success rates.

Scaling yields better memory curation. To fairly evaluate how scaling feeds back into memory, we report Pass@1, which measures the average quality of trajectories after memory curation and allows direct comparison with the no-scaling case. The trend is depicted in pick bars and is striking: scaling actually reduces performance for weaker memories, where Synapse drops from 40.6 to 40.1, and AWM from 44.4 to 41.2. These declines suggest that without strong guidance, the extra rollouts generated by scaling introduce noise rather than useful signals. In contrast, ReasoningBank is the only method that benefits: Pass@1 rises from 49.7 to 50.8, showing that high-quality memory can harness the diversity of scaling to extract constructive contrastive signals. This asymmetry highlights that scaling alone is insufficient; only when paired with good memory mechanism does it contribute to curation of more effective memory, thereby closing the virtuous cycle.

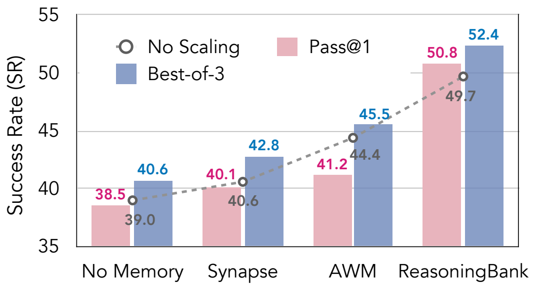


Figure 5: Snapshot of MaTTS on WebArena-Shopping subset with different memory mechanisms with $k = 3$. We compute BoN for all 3 trajectories and Pass@1 with one randomly selected trajectory.

5 Analysis

We analyze ReasoningBank beyond overall benchmark performance through three aspects: incorporating failure trajectories, examining emergent strategies, and evaluating efficiency across both successful and failed cases. Additional analyses are presented in Appendix C, including but not limited to number of retrieved experiences, calibration of LLM-as-a-judge, additional results on smaller open-source model, and inference cost study.

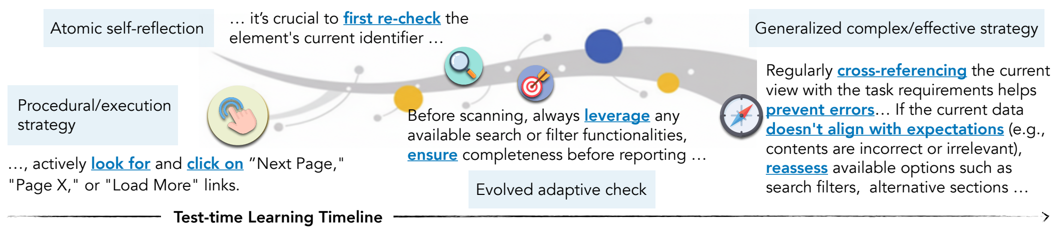

Atomic self-reflection
... it's crucial to first re-check the element's current identifier ...
Generalized complex/effective strategy
Procedural/execution strategy
Regularly cross-referencing the current view with the task requirements helps prevent errors... If the current data doesn't align with expectations (e.g., contents are incorrect or irrelevant), reassess available options such as search filters, alternative sections ...
Before scanning, always leverage any available search or filter functionalities, ensure completeness before reporting ...
..., actively look for and click on "Next Page," "Page X," or "Load More" links.
Evolved adaptive check
Test-time Learning Timeline

Figure 6: A case study illustrating emergent behaviors in ReasoningBank through memory items.

Emergent behaviors with ReasoningBank. We find that the strategies in ReasoningBank are not flat or monolithic, but instead evolve over time, exhibiting emergent behaviors that resemble the learning dynamics of RL (Wang et al., 2025a). As illustrated in a human case study in Figure 6, memory items describing a specific strategy "*User-Specific Information Navigation*" in ReasoningBank could gradually evolve during test-time learning process. It starts from execution-oriented or procedural strategies (e.g., find navigation links), where the agent follows straightforward action rules. It then progresses to adaptive self-reflections such as re-verifying

Table 4: Average number of steps on successful and failed test instances across four WebArena domains. REASONINGBANK consistently reduces the number of steps compared to the vanilla baseline, with notably larger reductions on successful instances.

Models	**Shopping**		**Admin**		**Gitlab**		**Reddit**	
	Successful	*Failed*	*Successful*	*Failed*	*Successful*	*Failed*	*Successful*	*Failed*
No Memory	6.8	8.7	8.4	10.4	8.6	15.7	6.1	7.6
REASONINGBANK	4.7↓2.1	7.3↓1.4	7.0↓1.4	9.5↓0.9	7.6↓1.0	15.5↓0.2	5.0↓1.1	6.8↓0.8

identifiers to reduce simple mistakes. With more experiences, the same memory item evolves into adaptive checks, where the agent systematically leverages available search or filters to ensure completeness before results. Finally, it eventually matures into compositional strategies such as cross-referencing task requirements and reassessing options. This evolution highlights how REASONINGBANK enables agents to refine strategies from low-level actions to high-level reasoning during test-time learning.

REASONINGBANK makes good use of failure trajectories. Figure 7 compares different memory designs on WebArena-Shopping with Gemini-2.5-flash under two settings: using only successful trajectories versus leveraging both successes and failures. Baseline methods such as Synapse and AWM build memory solely from successful trajectories, and thus are not equipped to benefit from failures. As a result, when failures are added, their performance is limited or even degraded: Synapse increases only from 40.6 (success only) to 41.7 (with failures), while AWM drops from 44.4 to 42.2. In contrast, the design of REASONINGBANK enables distillation of reasoning patterns from *both* successes and failures, achieving 46.5 on success-only traces and further improving to 49.7 when failures are included. This highlights that, unlike baselines, REASONINGBANK can transform failures into constructive signals rather than noise, enabling more robust generalization.

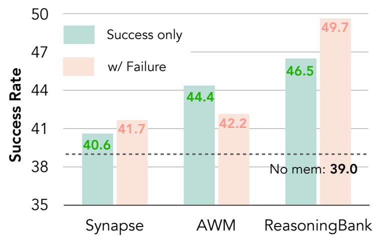


Figure 7: Ablation results of incorporating failure trajectories for memory induction.

REASONINGBANK delivers targeted efficient gains. While the overall number of steps in Table 1 provides a general view of model efficiency, it does not distinguish whether reductions come from successful or failed trajectories. To gain deeper insight, we further separate the analysis into successful and failed test cases, which allows us to understand the source of step reduction: a desirable system should reduce unnecessary exploration when it is on the right track, rather than merely cutting short failed attempts. The results are shown in Table 4. We find that REASONINGBANK consistently reduces the number of steps across all domains compared to the baseline. More importantly, the reduction is particularly pronounced on successful cases, reaching up to 2.1 fewer steps (a 26.9% relative reduction) than on failed ones. This indicates that REASONINGBANK primarily helps the agent reach solutions with fewer interactions by strengthening its ability to follow effective reasoning paths rather than simply truncating failed trajectories, which highlight the role of memory in guiding purposeful decision-making and improving efficiency in practice.

6 CONCLUSION

We introduce REASONINGBANK, a memory framework that distills strategy-level reasoning signals from both successes and failures and integrates them into test-time scaling (MATTS). Extensive experiments show that REASONINGBANK consistently improves performance while reducing redundant exploration. Further results reveal a strong synergy between memory and scaling: REASONINGBANK guides scaling toward more promising rollouts, while diverse rollouts enrich memory with valuable contrastive signals. We also provide analyses of individual components and emergent behaviors. Our findings suggest a practical pathway toward building adaptive and lifelong-learning agents, with additional future directions and limitations in Appendix D and E.

Ethics Statement

This work does not involve human subjects, sensitive personal data, or any tasks requiring Institutional Review Board (IRB) approval. All datasets used in our experiments (WebArena, Mind2Web, SWE-Bench-Verified) are publicly available and widely used in the community; detailed descriptions and preprocessing steps are provided in Appendix B to ensure transparency and compliance. Our methods, ReasoningBank and MaTTS, focus on improving the effectiveness, efficiency and generalization of LLM-based agents in web navigation and software engineering tasks. While these advances aim to benefit research on adaptive and lifelong-learning agents, we are mindful of potential concerns regarding misuse (e.g., automation of harmful tasks). We therefore restrict our experiments to established academic benchmarks and commit to open-sourcing our code under a research-only license to encourage responsible use. We have carefully followed the ICLR Code of Ethics throughout this research and submission process.

Reproducibility Statement

We have made extensive efforts to ensure the reproducibility of our work. The full methodology of ReasoningBank and Memory-Aware Test-Time Scaling (MaTTS) is detailed in Section 3, with additional implementation details provided in Appendix A. Experimental setups, including datasets (WebArena, Mind2Web, SWE-Bench-Verified), baselines, evaluation protocols, and environment configurations, are described in Section 4.1 and Appendix B. For clarity, we provide ablation studies (Section 5), efficiency analyses (Section 5), and emergent behavior case studies (Section 5) to validate the robustness of our findings. To further facilitate replication, we include precise descriptions of the memory schema, retrieval, construction, and consolidation pipeline in Section 3.2 and Appendix A.1-A.2, and we outline the scaling procedures for both parallel and sequential variants in Section 3.3 and Appendix A.3. We will release our full codebase to the open-source community to foster transparency, reproducibility, and future research.

References

Anthropic. Claude 3.7 sonnet and claude code, 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

Silin Chen, Shaoxin Lin, Xiaodong Gu, Yuling Shi, Heng Lian, Longfei Yun, Dong Chen, Weiguo Sun, Lin Cao, and Qianxiang Wang. Swe-exp: Experience-driven software issue resolution. *ArXiv preprint*, abs/2507.23361, 2025. URL https://arxiv.org/abs/2507.23361.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *ArXiv preprint*, abs/2504.19413, 2025. URL https://arxiv.org/abs/2504.19413.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Aviral Kumar, Rishabh Agarwal, Sridhar Thiagarajan, Craig Boutilier, and Aleksandra Faust. Inference-aware fine-tuning for best-of-n sampling in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=77gQUdQhE7.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *ArXiv preprint*, abs/2507.06261, 2025. URL https://arxiv.org/abs/2507.06261.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samual Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *ArXiv preprint*, abs/2508.06433, 2025. URL https://arxiv.org/abs/2508.06433.

Zafeirios Fountas, Martin Benfeghoul, Adnan Oomerjee, Fenia Christopoulou, Gerasimos Lampouras, Haitham Bou Ammar, and Jun Wang. Human-inspired episodic memory for infinite context LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BI2int5SAC.

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

Alireza Ghafarollahi and Markus J Buehler. Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523, 2025.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *ArXiv preprint*, abs/2411.15594, 2024. URL https://arxiv.org/abs/2411.15594.

Izzeddin Gur, Hiroki Furuta, Austin V. Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=9JQtrumvg8.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *ICML 2025 Workshop on Long-Context Foundation Models*, 2025. URL https://openreview.net/forum?id=ZgQ0t3zYTQ.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Can Jin, Hongwu Peng, Qixin Zhang, Yujin Tang, Dimitris N Metaxas, and Tong Che. Two heads are better than one: Test-time scaling of multi-agent collaborative reasoning. *ArXiv preprint*, abs/2504.09772, 2025. URL https://arxiv.org/abs/2504.09772.

Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents. *ArXiv preprint*, abs/2402.03610, 2024. URL https://arxiv.org/abs/2402.03610.

Yi Kong, Dianxi Shi, Guoli Yang, Chenlin Huang, Xiaopeng Li, Songchang Jin, et al. Mapagent: Trajectory-constructed memory-augmented planning for mobile task automation. *ArXiv preprint*, abs/2507.21953, 2025. URL https://arxiv.org/abs/2507.21953.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, et al. Gemini embedding: Generalizable embeddings from gemini. *ArXiv preprint*, abs/2503.07891, 2025. URL https://arxiv.org/abs/2503.07891.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E Gonzalez, and Ion Stoica. S*: Test time scaling for code generation. *ArXiv preprint*, abs/2502.14382, 2025. URL https://arxiv.org/abs/2502.14382.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *ArXiv preprint*, abs/2504.01990, 2025a. URL https://arxiv.org/abs/2504.01990.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14179–14198, Vienna, Austria, 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.694. URL https://aclanthology.org/2025.acl-long.694/.

Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A Burke. Memtool: Optimizing short-term memory management for dynamic tool calling in llm agent multi-turn conversations. *ArXiv preprint*, abs/2507.21428, 2025. URL https://arxiv.org/abs/2507.21428.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

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

Atsuyuki Miyai, Zaiying Zhao, Kazuki Egashira, Atsuki Sato, Tatsumi Sunada, Shota Onohara, Hiromasa Yamanishi, Mashiro Toyooka, Kunato Nishina, Ryoma Maeda, et al. Webchorearena: Evaluating web browsing agents on realistic tedious web tasks. *ArXiv preprint*, abs/2506.01952, 2025. URL https://arxiv.org/abs/2506.01952.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *ArXiv preprint*, abs/2501.19393, 2025. URL https://arxiv.org/abs/2501.19393.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. Memgpt: Towards llms as operating systems. 2023.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NPAQ6FKSmK.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. *ArXiv preprint*, abs/2401.13996, 2024. URL https://arxiv.org/abs/2401.13996.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without verification or RL is suboptimal. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=beeNgQEfe2.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. ReasonIR: Training retrievers for reasoning tasks. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=kkBCNLMbGj.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, et al. Thinking vs. doing: Agents that reason by scaling test-time interaction. *ArXiv preprint*, abs/2506.07976, 2025. URL https://arxiv.org/abs/2506.07976.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience. *ArXiv preprint*, abs/2508.04700, 2025. URL https://arxiv.org/abs/2508.04700.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.413. URL https://aclanthology.org/2025.acl-long.413/.

Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, Arman Cohan, and Mark Gerstein. Chemagent: Self-updating memories in large language models improves chemical reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=kuhIqeVg0e.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *ArXiv preprint*, abs/2507.06229, 2025b. URL https://arxiv.org/abs/2507.06229.

Haozhe Wang, Qixin Xu, Che Liu, Junhong Wu, Fangzhen Lin, and Wenhu Chen. Emergent hierarchical reasoning in llms through reinforcement learning. *ArXiv preprint*, abs/2509.03646, 2025a. URL https://arxiv.org/abs/2509.03646.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryLLM with scalable long-term memory. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=OcqbkROe8J.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. *ArXiv preprint*, abs/2504.06821, 2025c. URL https://arxiv.org/abs/2504.06821.

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

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. Streambench: Towards benchmarking continuous improvement of language agents. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/c189915371c4474fe9789be3728113fc-Abstract-Datasets_and_Benchmarks_Track.html.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *ArXiv preprint*, abs/2408.00724, 2024b. URL https://arxiv.org/abs/2408.00724.

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 Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. 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. 5180–5197, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.356. URL https://aclanthology.org/2022.acl-long.356.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *ArXiv preprint*, abs/2502.12110, 2025. URL https://arxiv.org/abs/2502.12110.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=WE_vluYUL-X.

Shaozhe Yin, Jinyu Guo, Kai Shuang, Xia Liu, and Ruize Ou. Learning wisdom from errors: Promoting llm's continual relation learning through exploiting error cases. *ArXiv preprint*, abs/2508.12031, 2025. URL https://arxiv.org/abs/2508.12031.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *ArXiv preprint*, abs/2507.02259, 2025a. URL https://arxiv.org/abs/2507.02259.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. ExACT: Teaching AI agents to explore with reflective-MCTS and exploratory learning. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=GBIUbwW9D8.

Zhaojian Yu, Yinghao Wu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. Z1: Efficient test-time scaling with code. *ArXiv preprint*, abs/2504.00810, 2025c. URL https://arxiv.org/abs/2504.00810.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *ArXiv preprint*, abs/2504.13837, 2025. URL https://arxiv.org/abs/2504.13837.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=PAPY0cAB3C.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *ACM Transactions on Information Systems*, 2024b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Pc8AU1aF5e.

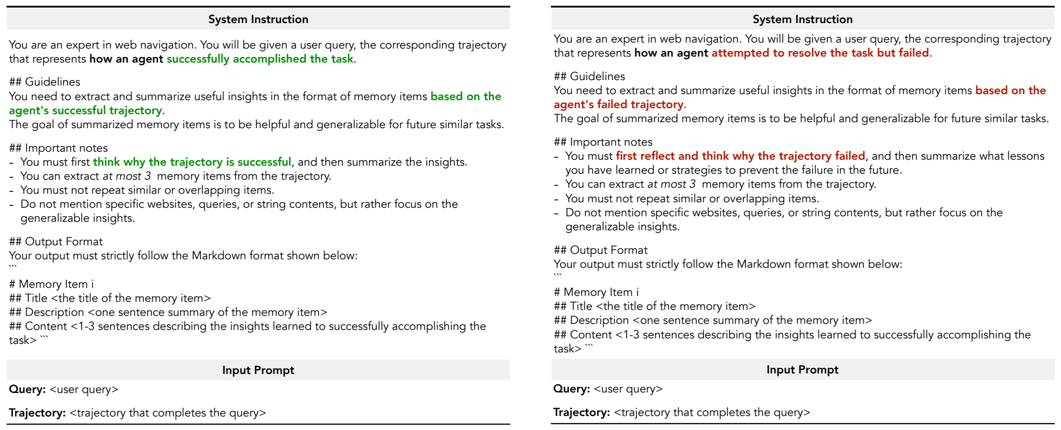
Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *ArXiv preprint*, abs/2506.15841, 2025. URL https://arxiv.org/abs/2506.15841.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for LLM agents. *ArXiv preprint*, abs/2506.12928, 2025a. URL https://arxiv.org/abs/2506.12928.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, et al. Scaling test-time compute for llm agents. *ArXiv preprint*, abs/2506.12928, 2025b. URL https://arxiv.org/abs/2506.12928.



~~~
System Instruction

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> ```

Input Prompt

Query: <user query>

Trajectory: <trajectory that completes the query>
~~~

~~~
System Instruction

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> ```

Input Prompt

Query: <user query>

Trajectory: <trajectory that completes the query>
~~~

Figure 8: System instructions for extracting memory items from agent trajectories: the left panel targets successful trajectories (summarizing why they succeed), while the right targets failed trajectories (reflecting on failure and deriving lessons).

A Experiment Details

This section details the implementation of ReasoningBank with agent systems mentioned in Section 4.1 for web browsing tasks including WebArena and Mind2Web. We first present all the prompts used for memory extraction in Appendix A.1, and then we provide the technical details for memory extraction, retrieval, and consolidation in Appendix A.2.

A.1 Prompts Used for ReasoningBank

Memory Extraction. Figure 8 illustrates the system instructions we used to guide the extraction of memory items from agent trajectories mentioned in Section 3.2. We will first obtain correctness signals from LLM-as-a-Judge (Gu et al., 2024) using the same backbone LLMs. When the trajectory corresponds to a successful case (left panel), the instruction emphasizes analyzing why the trajectory led to success and summarizing transferable reasoning strategies. Conversely, when the trajectory represents a failed case (right panel), the instruction requires reflecting on the causes of failure and articulating lessons or preventive strategies. In both settings, the output format is constrained to at most three memory items expressed in a structured Markdown format, ensuring that the resulting insights are concise, non-redundant, and generalizable across tasks rather than tied to specific websites or queries.

LLM-as-a-Judge for Correctness Signals. Figure 9 displays the instruction used for self-evaluation used to get binary signals for both successes and failures. Given the current user query, trajectory in resolving the query, final state of the website, and model output, the LLM is required to output the state of "Success" or "Failure" of whether the trajectory given successfully resolved the query or not.

A.2 Implementation Details

Memory Extraction. We use an LLM-based extraction pipeline to convert raw trajectories into structured memory items. Specifically, we design a prompt template that asks the model to distill reasoning patterns into three components: *title*, *description*, and *content* as previously mentioned in Appendix A.1. The backbone LLM of the extractor is set to the same as the agent system with temperature 1.0. For each trajectory, at most 3 memory items could be extracted. Crucially, we

System Instruction

You are an expert in evaluating the performance of a web navigation agent. The agent is designed to help a human user navigate a website to complete a task. Given the user's intent, the agent's action history, the final state of the webpage, and the agent's response to the user, your goal is to decide whether the agent's execution is successful or not.

There are three types of tasks:

1. Information seeking: The user wants to obtain certain information from the webpage, such as the information of a product, reviews, map info, comparison of map routes, etc. The bot's response must contain the information the user wants, or explicitly state that the information is not available. Otherwise, e.g. the bot encounters an exception and respond with the error content, the task is considered a failure. Besides, be careful about the sufficiency of the agent's actions. For example, when asked to list the top-searched items in a shop, the agent should order the items by the number of searches, and then return the top items. If the ordering action is missing, the task is likely to fail.
2. Site navigation: The user wants to navigate to a specific page. Carefully examine the bot's action history and the final state of the webpage to determine whether the bot successfully completes the task. No need to consider the bot's response.
3. Content modification: The user wants to modify the content of a webpage or configuration. Carefully examine the bot's action history and the final state of the webpage to determine whether the bot successfully completes the task. No need to consider the bot's response.

IMPORTANT
Format your response into two lines as shown below:
Thoughts: <your thoughts and reasoning process>"
Status: "success" or "failure"

Input Prompt

User Intent: {intent}

Trajectory: {trajectory}

The detailed final state of the webpage: ```md {cap} ```

Bot response to the user: {response if response else "N/A"}

Figure 9: System instructions for obtaining binary signals indicating success or failures of the current trajectory.

System Instruction

You are an expert in web navigation. You will be given a user query and **multiple trajectories showing how an agent attempted the task**. Some trajectories may be successful, and others may have failed.

## Guidelines
Your goal is to **compare and contrast** these trajectories to identify the most useful and generalizable strategies as memory items.
Use **self-contrast reasoning**:
- Identify patterns and strategies that consistently led to success.
- Identify mistakes or inefficiencies from failed trajectories and formulate preventative strategies.
- Prefer strategies that generalize beyond specific pages or exact wording.

## Important notes
- Think first: Why did some trajectories succeed while others failed?
- You can extract *at most 5* memory items from all trajectories combined.
- Do not repeat similar or overlapping items.
- Do not mention specific websites, queries, or string contents — focus on generalizable behaviors and reasoning patterns.
- Make sure each memory item captures **actionable** and **transferable** 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 <1-5 sentences describing the insights learned to successfully accomplishing the task> ```

**Input Prompt**

**Query:** <user query>

**Trajectories:** <trajectory 1>\n<trajectory 2>\n...<trajectory k>

**First-time Check Instruction**

Important: Let's carefully re-examine the previous trajectory, including your reasoning steps and actions taken.

Pay special attention to whether you used the correct elements on the page, and whether your response addresses the user query. If you find inconsistencies, correct them. If everything seems correct, confirm your final answer.

Output must stay in the same "<think>...</think><action></action>" format as previous trajectories.

**Follow-up Check Instruction**

Let's check again.

Output must stay in the same "<think>...</think><action></action>" format as previous trajectories.

Figure 10: System instructions for memory-aware test-time scaling: the left panel shows parallel scaling (comparing multiple trajectories to extract generalizable insights), while the right panel shows sequential scaling (iteratively re-checking a trajectory to refine the final answer).

induce items from *both* successful and failed trajectories. Successes provide validated strategies, while failures supply counterfactual pitfalls that act as negative signals. To determine success or failure, we adopt an LLM-based binary classifier following (Pan et al., 2024; Wang et al., 2025d). The classifier is prompted with the trajectory and the given user query, and asked to output a categorical judgment (`Success` or `Failure`) as shown in Figure 9. Similarly, the backbone of the classifier is set to the same as the agent system, with decoding temperature setting to 0.0 for determinism.

**Memory Retrieval and Response Generation.** For retrieval, we embed each task query using `gemini-embedding-001` (Lee et al., 2025), accessed via Vertex AI.[4] Similarity search is conducted

[4] https://ai.google.dev/gemini-api/docs/embeddings
```

over the memory pool using cosine distance. We select memory items of the top-k most similar experiences (default $k = 1$; ablation study in §5). The retrieved items are concatenated into the agent's system prompt with a simple formatting template (each item represented by its title and content) and instruction:

> Below are some memory items that I accumulated from past interaction from the environment that may be helpful to solve the task. You can use it when you feel it's relevant. In each step, please first explicitly discuss if you want to use each memory item or not, and then take action.

Memory Consolidation. After finishing each new query, the trajectory is processed by the extraction pipeline to produce new memory items, which are appended into the memory pool. We adopt a minimal consolidation strategy: newly generated items are directly added without additional pruning. This choice highlights the contribution of REASONINGBANK itself without introducing confounding factors from complex consolidation algorithms. Nevertheless, more advanced consolidation mechanisms (e.g., merging, forgetting) can be incorporated in future work.

REASONINGBANK Storage We maintain REASONINGBANK in a JSON format, and each entry of REASONINGBANK consists of a task query, the original trajectory, and the corresponding memory items. All memory items are stored with the schema {`title`, `description`, `content`}. The embedding is pre-computed for each given query and stored in another JSON file for efficient similarity search. We persist the memory pool for each independent run, enabling continual accumulation of experiences throughout test-time learning.

A.3 MATTS DETAILS

Prompt Used for MATTS Figure 10 illustrates the system instructions used in our MATTS framework mentioned in Section 3.3. In the parallel scaling setting (left), multiple trajectories for the same query—both successful and failed—are provided, and the model is instructed to perform self-contrast reasoning. Instead of relying on the LLM to act as an external judge of quality, the model is guided to directly compare and contrast trajectories, identifying patterns that lead to success and mistakes that cause failure. This provides a contrastive signal that grounds the memory extraction process in observable differences between outcomes, yielding more reliable and transferable insights. In the sequential scaling setting (right), the model repeatedly re-examines its own trajectory with check instructions, ensuring consistency and correction over iterations without appealing to external judgment.

Best-of-N Calculation Details. Given the task query and N trajectories from the agent system, we leverage an LLM and selects the best answer from the N trajectories. The LLM is initiated as the same backbone LLM as the agent system (e.g., if the agent system uses Gemini-2.5-flash, then the model also uses Gemini-2.5-flash). We feed all the N trajectories to the model at once and use a carefully curated prompt shown in Figure 11, asking the model to select the best answer.

B DETAILS FOR EXPERIMENT SETTINGS

B.1 WEB BROWSING

In this section, we detail the experiment settings used for web browsing agents mentioned in Section 4.1.

Datasets. We test REASONINGBANK on three agentic datasets for benchmarking web browsing and coding agents. Specifically, we conduct experiments on WebArena (Zhou et al., 2024) which features general web navigation across diverse domains, spaning shopping, administration, coding (Gitlab), and forums (Reddit). Another benchmark we used is Mind2Web (Deng et al., 2023), which provides playground to test the generalization of agents on versatile operations and environments, including cross-task, cross-website, and cross-domain settings. There are 684 and 1341 test instances in total for WebArena and Mind2Web, respectively. For WebArena, the number of

System Instruction

You are an expert in evaluating web navigation agent trajectories. You will be given the user query, and {N} candidate trajectories, each representing a sequence of steps for solving the same task. Your job is to select the single best trajectory that most effectively and efficiently solves the task, and explain your reasoning.

## Input Format:
Each trajectory consists of multiple steps. For each step, you will be provided:
- step_num: Step index in the trajectory.
- action_output: The action the agent takes (click, type, scroll, etc.).
- think_output: The agent's reasoning or plan before taking the action.

## Evaluation Criteria:

### Progress Toward Goal 1. How well the trajectory advances toward completing the user's task. 2. Reward tangible, meaningful progress; penalize minimal or no advancement. 3. Consider both individual step contributions and overall progress.

### Trajectory Efficiency 1. How efficiently the trajectory achieves progress given the number and complexity of steps. 2. Reward significant progress in fewer steps. 3. Favor better value-to-depth ratios. 4. Reward efficient search space exploration.

### Loop Detection: Identify loops or redundant actions. 1. Real Loops: Repeating identical observations and actions with no added value. 2. Benign Repetitions: Slight variations that still yield new information. 3. Penalize real loops heavily; penalize benign repetitions only if they waste effort.

### Error Severity and Stability: Assess severity of errors: 1. Fatal/Blocking: Major penalty. 2. Significant: Moderate penalty. 3. Minor/Recoverable: Minor penalty. 4. Penalize unstable or incoherent model reasoning. 5. Consider whether errors prevent goal completion.

### Overall Trajectory Quality 1. Logical flow of steps, clarity of strategy, and coherence. 2. Balanced exploration vs. exploitation. 3. Closeness to final goal. 4. Reward consistent progress and coherent planning.

## Output Format:
Return the evaluation as a JSON object: ``` { "index": [best_trajectory_index], "analysis": "Detailed reasoning explaining why this trajectory is the best, referencing progress, efficiency, loop detection, error severity, and overall quality." } ```

Input Prompt

Query: {query}

Trajectory 1: {trajectory_1}\n**Trajectory 2:** {trajectory_2}\n……\n**Trajectory N:** {trajectory_N}

Figure 11: System instructions for obtaining the best answer from N candidate trajectories for BoN calculation.

instances for different domains are Shopping (187), Admin (182), Gitlab (180), Reddit (106), and Multi (29). For Mind2Web, the number of different settings are Cross-Task (252), Cross-Website (177), and Cross-Domain (912).

Baselines. We compare REASONINGBANK against several representative memory-augmented approaches: (i) *Vanilla*, the backbone LLM agent without any memory module, serving as a reference point; (ii) *Synapse* (Zheng et al., 2024), a representative work that organizes past trajectories as in-context memory; and (iii) *AWM* (Wang et al., 2025d), which further abstracts common patterns from trajectories into reusable workflows. Together, these baselines span a progression from agents without memory, to those that directly reuse past trajectories, and finally to methods that distill higher-level structures, providing a comprehensive comparison for evaluating REASONINGBANK. To ensure a fair comparison, the baselines are implemented with the same "Memory Retrieval" and "Memory Consolidation" mechanisms. The only difference is about "Memory Extraction", which is exactly how REASONINGBANK different from baselines in terms of memory formulations.

Implementation Details. We build our agents upon several state-of-the-art LLMs accessed via the Vertex AI API,[5] including Gemini-2.5-Flash, Gemini-2.5-Pro (Comanici et al., 2025), and Claude-3.7-Sonnet (Anthropic, 2025). These choices allow us to investigate both cross-family (Gemini, Claude) and intra-family (Flash, Pro) variations. BrowserGym (de Chezelles et al., 2025) is used as the execution environment for WebArena, where we set a maximum step limit of 30 per query. The agent is implemented in ReAct (Yao et al., 2023) style, and iterates until the model predicts the stop action or reaches a task termination condition. We use the decoding temperature of 0.7 for model generations for both WebArena and Mind2Web.

Evaluation Metrics. For WebArena benchmark, we evaluate all methods across two key dimensions: *effectiveness* and *efficiency*. For effectiveness, we report the *success rate (SR)*. A task is marked as "successful" only if the agent's final output or state precisely matches the pre-defined ground-truth goal, which is measured by the evaluation protocol from the corresponding benchmarks. SR is the total number of successful tasks divided by the total number of tasks evaluated, formally, it is calculated as $SR = \frac{1}{N}\sum_{i=1}^{N} \text{isSuccess}(q_i)$, where $\text{isSuccess}(q_i)$ is the binary function that returns 1 if task q_i is successful and 0 otherwise. For efficiency, we measure

[5] https://cloud.google.com/vertex-ai

the *average number of steps (AS)* taken by the agent to complete each query, which reflects the computational and interaction cost incurred during task completion. A single step is defined as one complete agent-env interaction cycle following the ReAct loop, which typically involves observing the current state, generating a thought, and a subsequent action. AS is calculated as the total number of steps taken in the trajectory when solving task q_i divided by the total number of tasks, specifically, $AS = \frac{1}{N}\sum_{i=1}^{N} \text{Steps}(q_i)$ For Mind2Web dataset, each task in has a predefined fixed number of steps; at each step, the agent needs to predict an action, which is evaluated by: *element accuracy*: to check if the correct page element is selected, *action F1* to check if the action taken on the element is correct. Aggregating element accuracy and action F1 yields *step success rate* which checks that both element and action selection are correct at the current step. Lastly, after completing every step in the given task, the last metric *task-level success rate* measures if all intermediate steps are successfully conducted for this task, i.e., all steps for this task score 1.0 under metric step success rate.

B.2 Software Engineering

Datasets. To benchmark agentic coding tasks, we evaluate on SWE-Bench-Verified (Jimenez et al., 2024), a repository-level issue resolution benchmark. The dataset consists of 500 high-quality test instances that have been manually verified. Each instance requires generating a patch to address the underlying bug described in the input issue. The objective is to modify the relevant portions of the codebase such that all provided test scripts execute successfully.

Metrics. We report the issue resolution rate on SWE-Bench-Verified as the primary evaluation metric. The resolution rate measures the percentage of issues successfully fixed across all data points, where an issue is deemed resolved if the submitted patch passes all test scripts. To evaluate the patch application rate, we attempt to apply the generated patches to the repository using the standard `patch` program, counting only successful applications. Our implementation follows the official evaluation scripts.[6] For efficiency, we additionally report the average number of steps performed by the agent per instance, following web-browsing tasks.

Implementation. We implement ReasoningBank for SWE-Bench following the setting of mini-SWE-Agent (Yang et al., 2024), which enforces the Bash-Only environment with no tools and no special scaffold structure. It assumes a simple ReAct agent loop (Yao et al., 2023). Similar to previous experiments, we compare ReasoningBank against (i) No memory and (ii) Synapse. [7]

C Additional Analyses

C.1 Number of Retrieved Experiences

We conduct another ablation study on different number of retrieved experiences using Gemini-2.5-flash on WebArena-Shopping subset. As shown in Figure 12, we found that incorporating relevant memory significantly boosts performance (from 39.0 without memory to 49.7 with one experience). However, as the number of experiences increases, the success rate gradually declines (46.0 with 2, 45.5 with 3, and 44.4 with 4). This suggests that while memory provides valuable guidance, excessive experiences may introduce conflicts or noise. Hence, the relevance and quality of memory are more crucial than sheer quantity for effective performance.

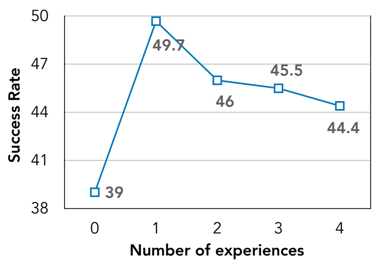


Figure 12: Ablation results for using various number of experiences.

[6]https://www.swebench.com/SWE-bench/api/harness/

[7]We exclude AWM here because the action space in mini-SWE-Agent is open-ended (arbitrary Bash commands), making it difficult to extract the common routines or fixed workflows that AWM requires for cross-task generalization.

C.2 Calibration of LLM-as-a-Judge

A critical step in our method is sourcing correctness signals for agent trajectories via an LLM-as-a-Judge. In this section, we quantitatively calibrate this judge and analyze its robustness to verification noise. We conduct our analysis on the WebArena-Shopping subset, using Gemini-2.5-flash as the judge. We first establish the baseline accuracy by comparing the judge's predictions against ground-truth labels, which we find to be 72.7%.

To systematically study the judge's robustness, we simulate different levels of verification accuracy. This is achieved by probabilistically correlating the judge's labels with the ground-truth. For example, a 100% accurate verifier uses the ground-truth labels directly. A 90% accurate verifier is simulated by using the correct (ground-truth) label 90% of the time and the incorrect (flipped) label 10% of the time. We extend this simulation down to 50% accuracy, which represents a random-guess baseline for this binary (success/failure) classification task. The results are presented in Figure 13.

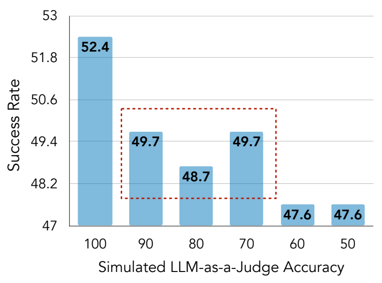


Figure 13: Results for success rate with respect to simulated LLM-as-a-judge accuracy.

We observe that ReasoningBank maintains a substantial performance improvement over baselines across all simulated judge accuracies. Furthermore, the judge's accuracy does not significantly impact the performance of ReasoningBank, as all variants achieve similar success rates within reasonable accuracy range (70%-90%). Intuitively, the 100% (ground-truth) accuracy setting yields the best performance. These findings confirm that ReasoningBank is robust to noise in the verification step.

C.3 Inference Cost Study

In this section, we provide a comprehensive view on inference cost for ReasoningBank and baselines to facilitate real-world deployment. We report a breakdown of the averaged total token consumption for each trajectory in addition to number of interaction steps mentioned in Section 4.1. The results are shown in Table 5.

Table 5: Breakdown results of total token consumption required for each task.

Methods	Action Generation	LLM-as -a-Judge	Memory Extraction	Total
No Memory	50847.4	-	-	50847.4
Synapse	55920.5	2594.2	-	58514.7
AWM	53819.6	2479.1	3074.1	59372.8
ReasoningBank	49306.1	2186.3	1562.1	53054.5

From the table we can see that compared with "No memory", while the total token consumption is increased only by around 4.3%, the overall performance is boosted by 20.5%. Other memory baselines such as Synapse and AWM will greatly increase the computation overhead while achieving less performance gains compared with ReasoningBank, demonstrating the cost-effectiveness of ReasoningBank.

C.4 Pass@k Analysis

Memory-aware scaling improves sample efficiency and sustains stronger performance gains. Pass@k analysis under parallel scaling on WebArena-Shopping subset with Gemini-2.5-flash (Figure 14) reveals two distinct effects. First, MaTTS w/o aggregation (Vanilla TTS) already makes test-time learning behave similarly to RL training: instead of inflating pass@k at large k, it improves sample efficiency by guiding exploration. For example, at $k = 2$, MaTTS w/o aggregation achieves 50.8 compared to 47.6 from MaTTS w/o memory, extracting more value from each rollout as noted in (Yue et al., 2025). Second, equipping TTS with memory-aware scaling pushes performance further. MaTTS not only preserves efficiency at small k (51.3 at $k = 2$) but also sustains strong growth with scaling, reaching 62.1 at $k = 5$, compared to only 52.4 for MaTTS w/o memory. Overall, we see that MaTTS unlocks more potential of agent systems and encourages diversified generation for better pass@k performance.

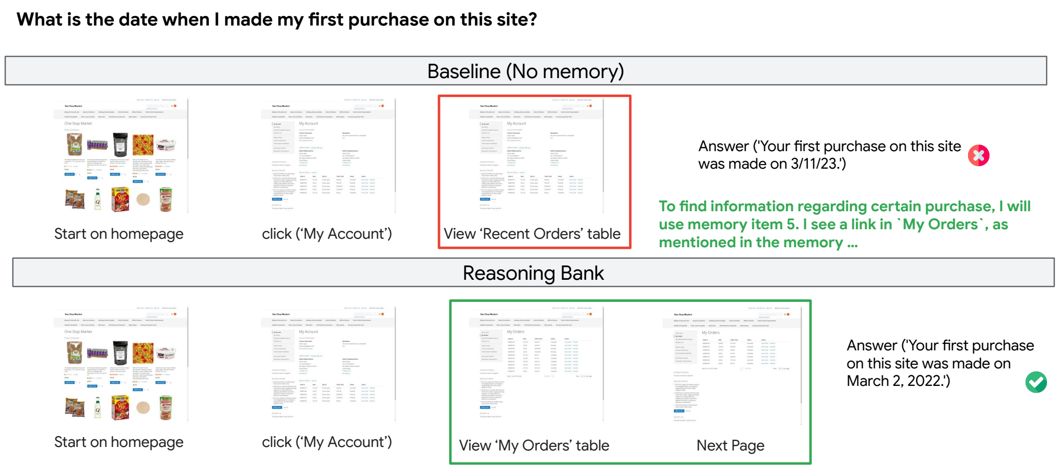

What is the date when I made my first purchase on this site?
Baseline (No memory)
Start on homepage
click ('My Account')
View 'Recent Orders' table
Answer ('Your first purchase on this site was made on 3/11/23.')
To find information regarding certain purchase, I will use memory item 5. I see a link in `My Orders`, as mentioned in the memory ...
Reasoning Bank
Start on homepage
click ('My Account')
View 'My Orders' table
Next Page
Answer ('Your first purchase on this site was made on March 2, 2022.')

Figure 15: REASONINGBANK enables the agent to recall and apply past reasoning hints, guiding it to the full order history and yielding the correct first purchase date, unlike the baseline that fails with only recent orders.

C.5 CASE STUDY

To better illustrate the benefits of our approach, we present three representative case studies.

Effectiveness. Figure 15 highlights the effectiveness of REASONINGBANK in leveraging related previous experiences as memory items. While the baseline agent (without memory) only checks the "Recent Orders" table and mistakenly outputs the most recent purchase date, REASONINGBANK recalls from past reasoning hints to explore the full purchase history and correctly identifies the earliest order.

Efficiency. Figure 16 demonstrates the efficiency gains. In a navigation-heavy shopping task, the baseline requires 29 steps due to repeated inefficient browsing. It stucks and struggles to find the correct place of filter for "Men". In contrast, REASONINGBANK leverages stored reasoning about category filtering, enabling the agent to directly reach the relevant items and complete the task in only 10 steps.

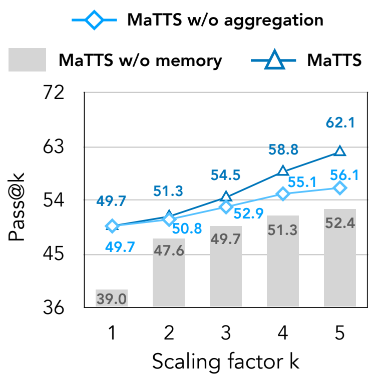


Figure 14: Pass@k under parallel scaling with REASONINGBANK.

Emergent Capabilities. Figure 17 shows how memory items induced by REASONINGBANK through reflecting on past trajectories helps to prevent similar errors from happening again, which enables emergent improvement. In this case, the original trajectory actually fails because of the imprecise search query that leads to numerous returned items, and irrelevant objects. REASONINGBANK is able to first reflect on the trajectory, pinpoint the key reason of failure, and extract valuable strategies that would avoid similar errors such as search query optimization and using the functionality filters.

D FUTURE DIRECTIONS

In this section, we briefly discuss the potential future directions following REASONINGBANK and MATTS.

Modular and Compositional Memory. Our current framework distills each experience into multiple memory items, and when a new query arrives, we retrieve similar experiences and reuse

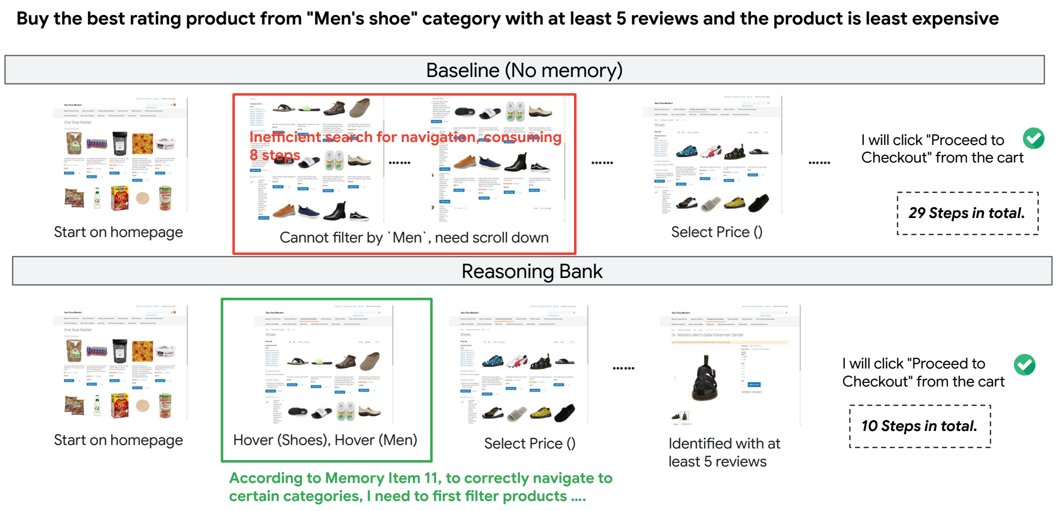

Buy the best rating product from "Men's shoe" category with at least 5 reviews and the product is least expensive
Baseline (No memory)
Start on homepage
Inefficient search for navigation, consuming 8 steps
Cannot filter by `Men`, need scroll down
Select Price ()
I will click "Proceed to Checkout" from the cart
29 Steps in total.
Reasoning Bank
Start on homepage
Hover (Shoes), Hover (Men)
According to Memory Item 11, to correctly navigate to certain categories, I need to first filter products ....
Select Price ()
Identified with at least 5 reviews
I will click "Proceed to Checkout" from the cart
10 Steps in total.

Figure 16: REASONINGBANK improves efficiency by leveraging past reasoning hints, reducing the navigation from 29 steps to 10 steps compared to the baseline without memory.

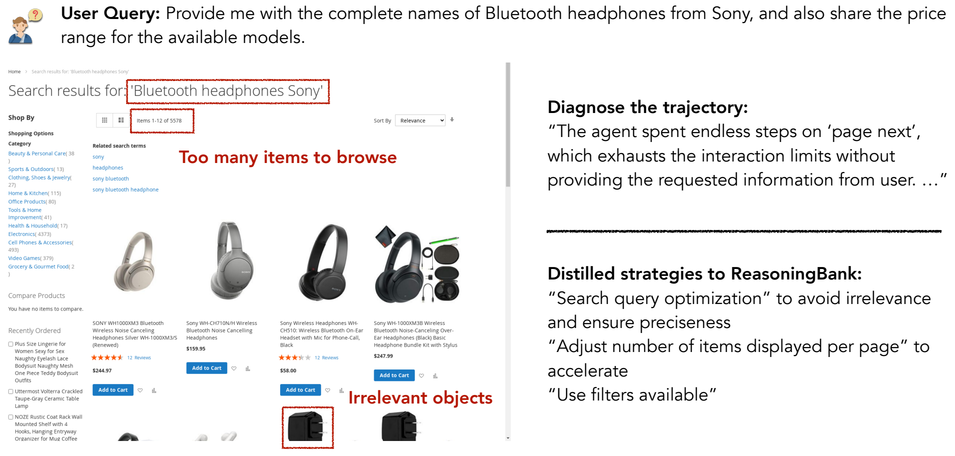

User Query: Provide me with the complete names of Bluetooth headphones from Sony, and also share the price range for the available models.
Search results for: 'Bluetooth headphones Sony'
Items 1-12 of 5578
Too many items to browse
Irrelevant objects
Diagnose the trajectory:
"The agent spent endless steps on 'page next', which exhausts the interaction limits without providing the requested information from user. …"
Distilled strategies to ReasoningBank:
"Search query optimization" to avoid irrelevance and ensure preciseness
"Adjust number of items displayed per page" to accelerate
"Use filters available"

Figure 17: Memory items induced by REASONINGBANK unlocks emergent improvement of REASONINGBANK, which help avoid similar errors from happening again.

all associated items independently. This design highlights the effect of memory content but does not consider how items could be composed into higher-level strategies. Future work could explore composition-aware retrieval and consolidation with based on modular memory extraction, enabling the agent to combine complementary items or form reusable macros, thereby yielding richer strategies and stronger generalization in long-horizon tasks. For example, memory could be extracted with respect to "planning memory", "tool-use memory", "operational memory", "user-centric memory", etc. In this way, memory extracted would be more fine-grained and memory retrieval could unlock compositional and complementary power, not just task similarity.

Advanced Memory Architectures. Our current system design is intentionally minimal; a natural next step is to build a layered, product-level memory stack that integrates mature paradigms — e.g., episodic traces (Fountas et al., 2025) for per-task context, short-term "working" memory (Lumer et al., 2025) for within-session state, and long-term (Wang et al., 2025b) consolidated knowledge with decay/refresh policies. The philosophy of REASONINGBANK are compatible with the above different memory angularities. Additionally, the current memory retrieval could also move beyond embedding-based similarities to reasoning-intensive controllers (Shao et al., 2025) that decompose queries, plan multi-hop lookups across tiers, and condition selection on uncertainty, recency, and cost. Learning-based routers and consolidation policies could also automate this process. This integration would turn REASONINGBANK with MATTS into a deployable memory service that scales across domains and teams.

E LIMITATIONS

While REASONINGBANK demonstrates strong empirical performance and introduces a practical paradigm for memory as a scaling dimension, it also comes with several limitations that suggest directions for future research.

Focus on memory content. Our study emphasizes how to curate and utilize memory content (e.g., integrating failure trajectories, constructing distilled reasoning cues). For this reason, we did not extensively compare with other memory architectures such as episodic or hierarchical memory. These designs address orthogonal concerns (memory form/structure), while our contribution targets what should be stored and reused. Exploring their combination would be an interesting future direction.

Simplicity in memory retrieval and consolidation. We intentionally adopt simple embedding-based retrieval and straightforward consolidation to better isolate the effect of content quality. More sophisticated strategies (e.g., adaptive retrieval, hierarchical consolidation) are compatible with our framework and could further enhance performance, but are not the focus of this work. This choice ensures that the observed gains can be attributed directly to the design of reasoning-oriented memory content.

Dependence on LLM-as-a-judge for correctness signals. In our implementation, success and failure signals for trajectories are determined by an LLM-as-a-judge. While this automatic labeling enables scalable evaluation without ground-truth feedback, it may introduce noise when tasks are ambiguous or when the judge model itself errs. While our results suggest the framework remains robust under such noise, future work could incorporate stronger verifiers, human-in-the-loop feedback, or ensemble judgment to enhance the reliability of memory induction.

F USE OF LLMS

We used LLMs as a general-purpose writing assist tool during the preparation of this submission. Specifically, LLMs were employed for polishing the clarity and readability of text (e.g., refining sentence structure, improving grammar, and shortening overly verbose phrasing). All research ideas, methodology design, experiments, analyses, and final writing decisions were conceived, implemented, and validated solely by the authors.