Escaping the Self-Confirmation Trap: An Execute-Distill-Verify Paradigm for Agentic Experience Learning

Shiding Zhu[1*] **Yudi Qi**[2*] **Yajie Wang**[3*] **Jiaze Li**[1] **Chao Song**[4]
Yaorui Shi[5] **Yibo Miao**[3] **Hanqi Gao**[6] **Kai Zhang**[2†]

[1]Zhejiang University [2]Shenzhen International Graduate School, Tsinghua University
[3]Independent Researcher [4]Northwestern Polytechnical University
[5]University of Science and Technology of China [6]Shanghai Jiaotong University

Abstract

Experience-driven self-evolution is essential for large language model (LLM) agents to improve through interaction with open-world environments. However, existing experience learning methods largely rely on single-agent loops, in which the same agent executes tasks, summarizes outcomes, and decides what should be written into memory. In such settings, agents are prone to the **Self-Confirmation Trap**, where wrong-but-self-consistent trajectories are mistakenly treated as successful experience, leading to error accumulation through later retrieval and reuse. To address this challenge, we propose **EDV**, an **Execute-Distill-Verify** framework for reliable experience learning. In the **Execute** stage, multiple heterogeneous agents explore the same task space in parallel, generating diverse candidate trajectories. In the **Distill** stage, a designated third-party distillation agent comparatively analyzes these trajectories and produces candidate experiences, reducing the bias of executor-centric self-summarization. In the **Verify** stage, the execution group jointly validates candidate experiences through a consensus-based mechanism, and only experiences that pass strict validation are written into shared or private memory. By decoupling execution, distillation, and validation, EDV turns experience learning from an isolated self-reflection loop into a collaborative experience construction process that suppresses erroneous and noisy experience before memory insertion. We evaluate EDV on challenging long-horizon benchmarks, including τ^2-bench, Mind2Web, and MMTB. Experimental results show that EDV consistently outperforms strong baselines, demonstrating the value of improving the reliability of experience construction for agent self-evolution. These findings suggest that robust agent improvement depends not only on richer memory, but also on how experience is constructed before it enters memory. Our code is available at https://github.com/shidingz/EDV.

1 Introduction

Large language model (LLM)(Vaswani et al., 2017; Radford et al., 2018; 2019; Brown et al., 2020; Achiam et al., 2023) agents are increasingly being deployed in persistent, open-world environments. In such settings, improvement depends not only on solving the current task, but also on whether agents can continuously accumulate and reuse useful experience from past interactions. This motivates *experience learning*, a paradigm in which agents distill reusable lessons from historical executions, store them as external memory, and retrieve them to support future decisions(Xu et al., 2025b). By transforming transient trajectories into persistent knowledge, experience learning offers a promising path toward continual agent self-evolution(Zhang et al., 2026).

Recent work(Wu et al., 2025; Wei et al., 2025; Tan et al., 2025; Zhang et al., 2025a; Liu et al., 2025b) has shown that memory-based self-evolution can effectively improve long-horizon reasoning and decision-making. These methods typically build a loop of *execution, summarization, retrieval, and reuse* over interaction histories, enabling agents to accumulate experience over time. For example, *ReasoningBank* shows that reasoning

*Equal contribution.
†Corresponding author: zhangkai@sz.tsinghua.edu.cn. Additional emails: Shiding Zhu (zhusd@zju.edu.cn); Yudi Qi (qyd24@mails.tsinghua.edu.cn).

memory distilled from successful and failed attempts can serve as more reusable knowledge than raw trajectories for future tasks (Ouyang et al., 2025). At the same time, research on multi-agent systems(Li et al., 2023; Hong et al., 2023; Chen et al., 2023; Du et al., 2024; Qian et al., 2024; Liu et al., 2024) suggests that collaboration and heterogeneity provide an important route for overcoming the limitations of single-model systems. Prior work has shown that heterogeneous multi-agent systems can outperform single-model or homogeneous configurations on complex tasks by leveraging complementary capabilities(Li et al., 2024; Guo et al., 2024; Ye et al., 2025; Xue et al., 2025; Zou et al., 2025). This naturally raises a question: can multi-agent collaboration be used not only to improve task solving, but also to improve the reliability of experience construction itself?

However, most existing experience learning methods still rely on a *single-agent loop*: the same agent executes the task, interprets the outcome, distills the lesson, and decides what to write into memory. We argue that this design is fundamentally brittle in open-world environments without explicit ground truth(Schroeder & Wood-Doughty, 2024), because execution and evaluation are coupled within the same reasoning process(Huang et al., 2023). As a result, trajectories that are internally coherent but actually incorrect may be written into memory and reinforced through later retrieval and reuse.

We term this failure mode the **Self-Confirmation Trap**: an agent mistakes wrong-but-self-consistent trajectories for successful experience and progressively amplifies these errors through memory accumulation(Self-Reflection, 2024). This issue is especially severe in long-horizon tasks, where intermediate decisions are often difficult to verify directly and apparent task progress may conceal flawed reasoning(Hu et al., 2025). Therefore, the central challenge of experience learning is not merely how to collect more experience, but how to construct experience more reliably and suppress erroneous and noisy content before memory insertion.

To address this challenge, we propose **EDV**, an **Execute-Distill-Verify** framework for reliable experience learning, as illustrated in Figure 1. As illustrated in Figure 2, EDV consists of an *experience construction stage*, in which candidate experience is constructed through heterogeneous execution, third-party distillation, and consensus-based validation, and an *inference-time usage stage*, in which the system routes a new task to an appropriate solver and retrieves relevant experience from memory. In **Execute**, multiple heterogeneous agents explore the same task space in parallel, producing diverse candidate trajectories and expanding coverage of the solution space. In **Distill**, a designated *distillation agent* performs comparative analysis over these trajectories and distills candidate experiences, reducing the bias of executor-centric self-summarization. In **Verify**, the *verification group* jointly cross-validates candidate experiences, and only experiences that pass strict validation are written into memory. By decoupling execution, distillation, and validation, EDV transforms experience learning from an isolated self-reflection loop into a collaborative experience filtering process.

This design provides three key benefits. First, heterogeneous parallel execution mitigates the exploration bias of any single agent and increases the diversity of candidate trajectories. Second, third-party distillation turns experience construction from executor-centric self-summarization into cross-trajectory comparative analysis, thereby reducing bias introduced by a single viewpoint. Third, consensus-based validation raises the threshold for memory insertion, making the long-term memory system less vulnerable to contamination by erroneous and noisy experience. We evaluate EDV on challenging long-horizon benchmarks, including τ^2-bench(Barres et al., 2025), Mind2Web(Deng et al., 2023), and MMTB(Yu et al., 2025), and show that it consistently outperforms strong baselines. These results suggest that robust agent self-evolution requires not only richer memory, but also a more reliable pipeline for constructing that memory.

Our contributions are summarized as follows:

- We identify and characterize the **Self-Confirmation Trap**, a core failure mode of single-agent experience learning in open-world environments.
- We propose **EDV**, an **Execute-Distill-Verify** framework that improves the reliability of *experience construction* through heterogeneous execution, third-party distillation, and consensus-based validation.
- We demonstrate on multiple challenging long-horizon agent benchmarks that EDV consistently outperforms strong baselines, validating its effectiveness in suppressing erroneous and noisy experience before memory insertion.

2 Related Work

2.1 Experience Learning and Agentic Memory

Recent research has increasingly explored how agents can improve through sustained interaction with their environments, treating past executions as an important source of experience for continual adaptation. Early frameworks such as *AgentEvolver* and *FLEX* established the basic paradigm of learning from execution trajectories (Zhai et al., 2025; Cai et al., 2025). Subsequent work further extends this paradigm across diverse scenarios: *Learning on the Job* studies continual improvement in general long-horizon interactive settings, while *Seed-Prover 1.5* focuses on long-chain formal reasoning tasks (Yang et al., 2025; Chen et al., 2025a). In parallel, *MetaAgent* achieves single-agent self-evolution through meta-learning of tool usage patterns and accumulated experience (Qian & Liu, 2025).

A closely related line of work focuses on *agentic memory*, where past interactions are externalized into reusable knowledge. *ReasoningBank* highlights the value of distilled reasoning memory and memory-aware test-time scaling (Ouyang et al., 2025); *General Agentic Memory* investigates hierarchical memory organization architectures (Yan et al., 2025); and *CoPS* studies cross-task experience sharing mechanisms for single agents (Yang et al., 2024). These studies collectively demonstrate the importance of reusable memory for long-horizon reasoning, yet most existing methods still rely on single-agent loops for experience construction, lacking dedicated mechanisms to verify the correctness and reliability of memory content before insertion.

2.2 Multi-Agent Collaboration for Reliable Experience Construction

Another related line of research studies how multi-agent collaboration can overcome the limitations of single-model systems through diverse perspectives and complementary capabilities. Collaborative LLM systems have been shown to outperform single agents on complex reasoning and decision-making tasks (Li et al., 2024; Guo et al., 2024; Zou et al., 2025; Michelman et al., 2025; Xu et al., 2025a; Zhou & Chen, 2025; Liu et al., 2025a). In particular, *X-MAS* highlights the fundamental value of model heterogeneity in multi-agent systems, while *CoMAS* and *Multi-Agent Evolve* explore interaction-driven improvement and co-evolutionary dynamics in multi-agent settings respectively (Ye et al., 2025; Xue et al., 2025; Chen et al., 2025b).

Work such as *Xolver* further combines multi-agent collaboration with experience accumulation, enabling distributed capability improvement through shared team-level experience (Hosain et al., 2025). However, existing multi-agent studies mostly focus on improving single-task solving performance, or only address the sharing and transfer of experience across agents. Few have leveraged multi-agent division of labor to improve the reliability of the experience construction process itself.

3 The Self-Confirmation Trap in Experience Learning

Before presenting our framework, we first identify a fundamental failure mode in existing experience learning paradigms, which we term the **Self-Confirmation Trap**. This failure arises when the same agent is responsible for both task execution and trajectory evaluation. In open-world environments, where explicit ground-truth feedback is often unavailable, such self-validation can cause flawed trajectories to be mistakenly accepted as valid experience and written into memory.

3.1 Formal Characterization

We study experience learning in open-world environments, where agents distill reusable structured knowledge from historical executions. Let $\mathcal{T}$ denote a task set. For a task $q \in \mathcal{T}$, an agent with policy π_θ interacts with the environment and produces a trajectory

$$\tau = \{s_0, a_0, o_1, a_1, \ldots, o_T\}.$$

Let $c(\tau) \in \{0, 1\}$ denote the objective correctness of the trajectory, where $c(\tau) = 1$ indicates a valid trajectory and $c(\tau) = 0$ otherwise. For simplicity, we model self-evaluation as a binary approval decision

$$v_{\pi_\theta}(\tau) \in \{0, 1\},$$

where $v_{\pi_\theta}(\tau) = 1$ means that the agent judges the trajectory as suitable for experience extraction.

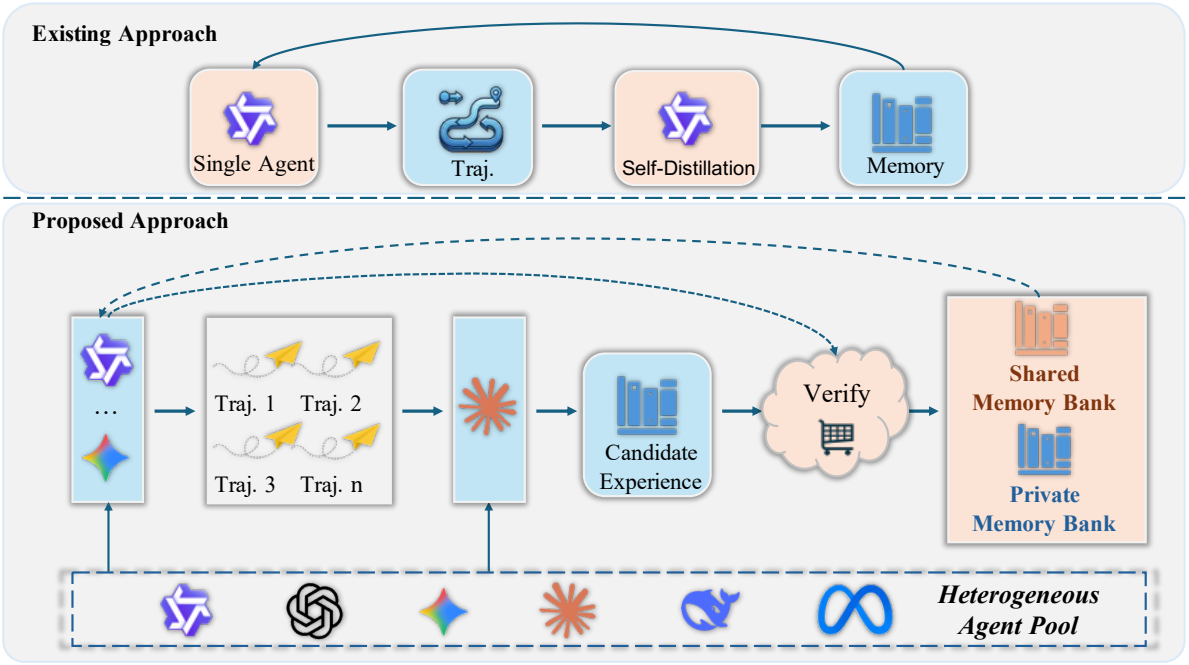

Figure 1: **Comparison between conventional single-agent learning and the proposed EDV framework. Top:** A single agent executes the task, generates a trajectory, performs self-distillation, and writes the resulting content into memory. Such a closed loop is prone to the "self-confirmation trap," where wrong-but-self-consistent experience may be reinforced through reuse. **Bottom:** EDV adopts a heterogeneous multi-agent pipeline. Multiple agents generate candidate trajectories, which are distilled into candidate experience and verified before being written into the shared or private memory bank.

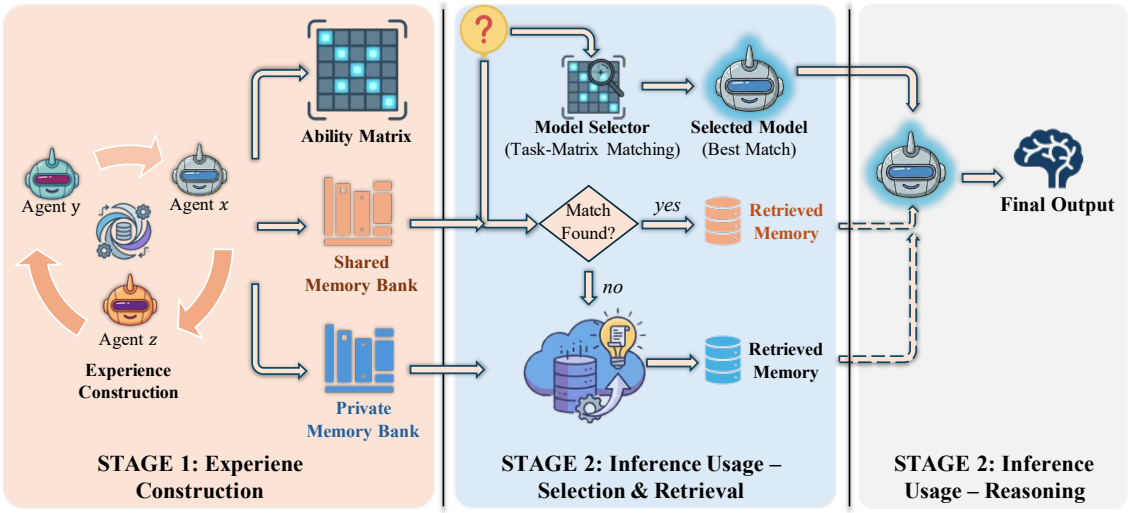

Figure 2: **Overall workflow of the EDV framework. Stage 1: Experience construction.** Multiple heterogeneous agents construct experience, update the ability matrix, and write verified content into the shared or private memory bank. **Stage 2: Inference-time usage.** The system selects the most suitable model via the ability matrix, retrieves relevant memory through hierarchical retrieval, and produces the final output.

In a standard single-agent loop, the same policy π_θ is used for both execution and evaluation. This coupling makes execution errors and evaluation failures statistically dependent. As a result, flawed trajectories may be incorrectly endorsed as successful, i.e.,

$$P\big(v_{\pi_\theta}(\tau) = 1 \mid c(\tau) = 0\big)$$

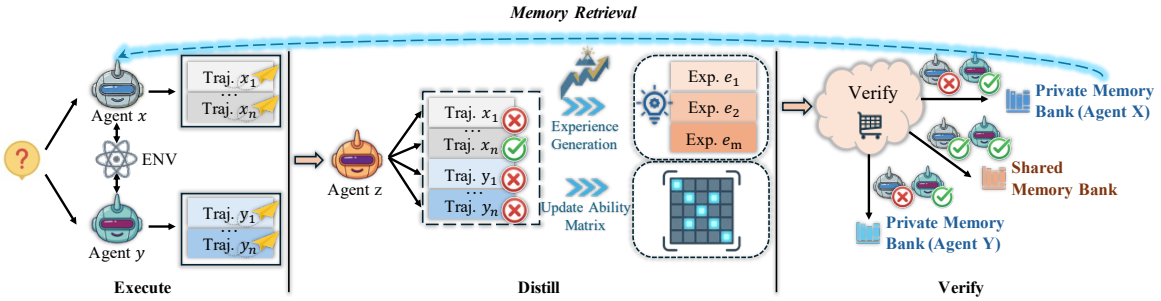

Figure 3: **Detailed process of EDV in the experience construction stage.** In **Execute**, multiple agents interact with the same environment and generate diverse candidate trajectories. In **Distill**, a third-party agent analyzes these trajectories, generates candidate experience, and updates the ability matrix. In **Verify**, unanimously approved experience is written into the shared memory bank, partially approved experience is written into the corresponding private memory bank, and the rest is discarded.

is substantially elevated. This erroneous endorsement of flawed trajectories constitutes the core of the **Self-Confirmation Trap**. Once written into memory, such errors can be repeatedly retrieved and reused, leading to persistent error accumulation.

3.2 Illustrative Example

A concrete example from our experiments highlights this issue. In a flight rebooking task, a single-agent system may repeatedly attempt to use a travel certificate for payment without recognizing a hidden environmental constraint: the certificate cannot be used to modify an existing reservation. Because the resulting behavior remains locally coherent, the trajectory appears plausible from the agent's own perspective. Under a single-agent self-validation loop, this flawed trajectory may be misjudged as successful experience and stored in memory, causing the same error pattern to reappear in future similar tasks.

3.3 Implications for Experience Construction

The root cause of this failure is the coupling of execution and evaluation within the same decision process. When trajectory generation and validation rely on the same perspective, systematic mistakes are more likely to be reinforced than corrected. This observation motivates a decoupled experience construction process, in which candidate experience is examined from independent roles before memory insertion. Our **Execute-Distill-Verify (EDV)** framework is designed precisely to address this problem through multi-agent collaboration.

4 Method

4.1 Overview of EDV

To address this issue, we propose **EDV**, an **Execute-Distill-Verify** framework for reliable experience learning. As illustrated in Figure 2, EDV has two levels: an *experience construction stage*, which forms high-quality experience from task executions, and an *inference-time usage stage*, which retrieves and applies such experience when a new task arrives.

As illustrated in Figure 3, in the experience construction stage, EDV contains three steps. In **Execute**, multiple heterogeneous agents explore the same task space in parallel and produce candidate trajectories. In **Distill**, a designated *distillation agent* comparatively analyzes these trajectories and generates candidate experiences. In **Verify**, the *verification group* performs consensus-based validation, and only validated experiences are written into memory.

Let the heterogeneous agent pool be

$$\mathcal{A} = \{A_1, A_2, \ldots, A_K\},$$

where each agent corresponds to a different foundation model or prompting strategy. For each task, EDV randomly samples a subset of agents from this pool to form the *execution group*, and then randomly selects

one agent from the pool to serve as the distillation agent. This design increases the diversity of experience sources while avoiding persistent role bias.

4.2 Execute: Heterogeneous Parallel Trajectory Generation

For each task, EDV constructs a heterogeneous *execution group*

$$\mathcal{A}_{\text{exec}} \subseteq \mathcal{A},$$

and each execution agent independently interacts with the environment to produce its candidate trajectories:

$$\mathcal{T}(q) = \{\tau_i \mid \tau_i \sim \pi_i(\cdot \mid q),\ A_i \in \mathcal{A}_{\text{exec}}\}.$$

The purpose of this step is to expand solution-space coverage and provide diverse trajectories for later comparison. Rather than simply increasing the number of attempts, EDV uses heterogeneity to expose different success and failure patterns, providing richer evidence for subsequent experience construction.

4.3 Distill: Third-Party Contrastive Experience Distillation

Diverse trajectories alone are insufficient to resolve the Self-Confirmation Trap. The key question is who decides what counts as experience worth keeping. To reduce executor-centric bias, EDV introduces a third-party *distillation agent*

$$A_{\text{distill}} \in \mathcal{A},$$

which performs cross-trajectory comparison over the trajectories produced by the execution group and outputs a candidate experience set

$$\mathcal{E}_{\text{cand}} = \{e_1, e_2, \ldots, e_m\}.$$

The distillation agent does not restate a single best trajectory; instead, it identifies useful differences across multiple trajectories and distills them into reusable candidate experiences.

4.4 Verify: Consensus-Based Experience Validation

Verify is the key mechanism through which EDV interrupts the Self-Confirmation Trap. For each candidate experience $e \in \mathcal{E}_{\text{cand}}$, the original execution agents validate it based on their own execution context. Let the verification group be

$$\mathcal{A}_{\text{verify}} = \mathcal{A}_{\text{exec}},$$

and let each executor provide a binary judgment

$$V_j(e) \in \{0, 1\}.$$

EDV adopts a strict **default-reject** policy. If a candidate experience receives unanimous approval, it is written into the *shared memory bank*; if it is approved only by a subset of executors, it is written into the corresponding *private memory bank*; otherwise, it is discarded. In this way, EDV raises the threshold for memory insertion and suppresses erroneous and noisy experience before it enters long-term memory.

4.5 Experience Storage and Inference-Time Usage

EDV maintains two types of memory storage: a *shared memory bank* for generally reusable experience and a *private memory bank* for agent-specific experience. In addition, it maintains an **Ability Matrix**, which records which types of tasks are better handled by which agents.

At inference time, given a new task q_{test}, the system first matches it against the Ability Matrix to select an appropriate solver. It then performs *hierarchical retrieval*: EDV first queries the shared memory bank, and if the retrieved results are insufficient, it queries the private memory bank of the selected solver. The retrieved memories are appended to the task context and used to guide subsequent reasoning.

5 Experiments

We evaluate EDV from four dimensions: (1) downstream performance improvements across benchmarks; (2) the actual improvement in written memory quality brought by the Execute-Distill-Verify mechanism; (3) the harm of erroneous memory to the system, thereby validating the real-world risk of the Self-Confirmation Trap; and (4) the specific contributions of EDV's key designs and core components to the final performance.

Table 1: **Pass@1 on the τ^2-bench for real-world issue resolution.** We compare pass@1 scores across three service domains. **Ours consistently achieves the best performance across all domains.**

Method	Backbone Model	AIRLINE	RETAIL	TELECOM	Avg.
No Memory (NM)	Minimax-M2.1	61.5	82.2	85.5	76.4
	Mimo-V2-Flash	64.5	79.2	91.7	78.5
	GLM-4.7-FP8	64.0	78.1	96.6	79.6
ReasoningBank (RB)	Minimax-M2.1	66.0	83.3	87.7	79.0
	Mimo-V2-Flash	64.0	83.3	94.7	80.7
	GLM-4.7-FP8	66.0	82.5	97.2	81.9
Judge	(Ensemble)	66.0	84.7	93.9	81.5
Router	(Ensemble)	68.0	85.2	97.2	83.5
EDV (Ours)	(Ensemble)	**72.0**	**88.6**	**99.1**	**86.6**

Table 2: **Quantitative results under different generalization settings.** We report Element Accuracy (EA), Action F1 (AF1), Step Success Rate (SSR), and Success Rate (SR); upper bounds are shown in small gray text. Ours consistently improves over strong baselines.

Metric	Minimax-M2.1		Mimo-V2-Flash		GLM-4.7-FP8		Judge	Router	Ours
	NM	RB	NM	RB	NM	RB			
cross-Task									
EA (74.26)	39.92	39.06	42.38	46.77	42.64	44.46	41.37	44.37	**48.62**
AF1 (74.26)	59.14	59.75	58.31	61.83	55.74	57.78	56.00	57.02	**63.10**
SSR (74.26)	34.19	33.95	37.48	42.01	38.31	40.92	35.14	40.28	**43.17**
SR (24.21)	3.57	3.17	3.21	**5.55**	3.84	4.37	3.06	4.37	4.76
cross-Website									
EA (64.82)	35.03	36.42	35.55	36.81	39.26	41.66	35.01	39.91	**44.79**
AF1 (64.82)	49.59	49.79	48.02	49.63	51.65	53.42	46.85	52.15	**55.64**
SSR (64.82)	29.13	29.72	30.29	31.78	31.17	35.83	27.63	31.76	**36.56**
SR (20.34)	2.82	2.82	**4.76**	4.12	3.39	4.52	1.87	3.39	**4.76**
cross-Domain									
EA (65.42)	36.24	37.95	36.49	40.09	40.47	41.13	42.64	41.82	**43.39**
AF1 (65.42)	52.00	53.58	51.77	54.23	52.25	52.98	55.74	53.37	**56.38**
SSR (65.42)	31.52	33.53	32.73	36.77	36.76	37.07	38.31	38.74	**39.57**
SR (18.31)	2.74	2.41	3.32	**5.05**	3.84	4.06	3.84	4.71	4.71

5.1 Experimental Setup

Benchmarks. We evaluate EDV on three representative long-horizon agent benchmarks: τ^2-bench(Barres et al., 2025), Mind2Web(Deng et al., 2023), and MMTB(Yu et al., 2025), covering complex constraint solving, web interaction, and multi-tool task execution, respectively.

Baselines. We compare EDV against: NM (No Memory), which measures the intrinsic capability of a single backbone model; Reasoning Bank (RB), a representative single-agent memory learning method; Judge, which uses an LLM-as-Judge strategy for inference-time adjudication over heterogeneous agent outputs; and Router(Ong et al., 2024), which selects the most suitable model at inference time via an Ability Matrix.

Implementation Details. EDV is built on a heterogeneous model pool consisting of Mimo-V2-Flash(Xiaomi, 2025), GLM-4.7-FP8(Zhipu AI, 2025), and MiniMax-M2.1(MiniMax, 2025). During memory construction, the system randomly samples two models to form the execution group and assigns another model as the

Table 3: **Main quantitative results on the MMTB benchmark.** Comparison of aggregate performance (**All**) and detailed breakdowns across Action, Multi-action, and Trajectory levels. **Ours achieves the best overall score and consistently outperforms or ties baselines on action-level metrics.**

Method	Backbone Model	Overall	Action-level			Multi-action			Trajectory	
		All	A_{sgl}	A_{cht}	A_{clr}	A^{P}	A^{S}	$A^{\text{S+P}}$	Opt. Path	Acc. Prog
No Memory (NM)	Minimax-M2.1	51.56	56.64	83.20	36.72	42.04	12.50	9.52	29.05	29.31
	Mimo-V2-Flash	46.90	56.64	69.05	31.37	41.94	18.75	11.90	14.22	41.21
	GLM-4.7-FP8	52.34	54.30	74.22	39.45	48.41	25.00	25.00	40.66	47.07
ReasoningBank (RB)	Minimax-M2.1	53.67	56.47	83.98	39.84	42.04	25.00	20.73	32.22	40.03
	Mimo-V2-Flash	49.22	57.03	68.90	36.33	41.29	31.25	22.62	9.62	51.09
	GLM-4.7-FP8	52.61	55.20	75.00	41.02	47.10	18.75	27.71	38.24	47.22
Judge	(Ensemble)	54.79	57.81	75.78	43.75	50.96	25.00	25.00	38.17	53.10
Router	(Ensemble)	55.96	59.38	79.69	44.14	48.41	18.75	**29.76**	35.30	**59.30**
Ours	(Ensemble)	**58.10**	**60.94**	**84.38**	**46.09**	**51.59**	**37.50**	**29.76**	**39.83**	50.42

distillation agent. After the Verify stage, unanimously approved experiences enter the shared memory bank, partially approved ones enter the private memory bank, and the rest are discarded. More details can be seen in Appendix C.

5.2 Main Results

Experimental results demonstrate that EDV achieves stable and consistent performance improvements across all three benchmarks. Specifically, on τ^2-bench, EDV achieves an average Pass@1 of 86.6, significantly outperforming Router (83.5) and Judge (81.5), while the single-model baseline without memory (NM) only achieves between 76.4 and 79.6. Under the cross-task, cross-website, and cross-domain settings of Mind2Web, EDV similarly maintains strong generalization performance. Furthermore, in the MMTB evaluation, EDV obtains an overall score of 58.10, again surpassing Router (55.96).

These results fully demonstrate that EDV's performance gains do not merely rely on scaling the number of models or inference-time selection, but truly benefit from the high-fidelity memory construction process brought by the Execute-Distill-Verify mechanism. Details of the evaluation benchmarks, definitions of the evaluation metrics, and specific benchmark configurations are provided in Appendix D.

5.3 Audit of Written Memory Quality

To directly verify EDV's improvement on memory quality, we conducted a human audit of the items actually stored in the memory bank, focusing specifically on the RETAIL domain of τ^2-bench.

The results show that, on a 5-point scale, EDV comprehensively outperforms the RB baseline across all dimensions: Groundedness/Correctness increases from 3.72 to 4.41, Actionability from 3.58 to 4.32, and Specificity from 3.64 to 4.27. Concurrently, scores for Noise/Hallucination decrease significantly from 1.21 to 0.63, and Potential Harm if Reused drops from 1.08 to 0.51. This proves that the Execute-Distill-Verify pipeline effectively filters out low-quality information at the source.

5.4 Sensitivity to Memory Contamination

To simulate the **Self-Confirmation Trap**, we take Reasoning Bank (RB) as the baseline and inject erroneous yet internally coherent experiences (e.g., incorrect payment rules) accounting for 10% of the total memory volume into the RETAIL task. Experimental results show that such contamination causes the Pass@1 score of RB on the τ^2-bench RETAIL domain to drop sharply from 82.5 to 77.2. Combined with the memory quality audit in Section 5.3, this provides strong evidence of the real-world harm posed by the Self-Confirmation Trap.

Table 4: **Progressive paradigm ablation results on the RETAIL domain of τ^2-bench.**

Baseline Method	Exec. Agents	Distill Role	Verify Role	Pass@1	Drop
Single-Agent Group (Fixed MiniMax-M2.1)					
1. Basic SA	1	Self	None	83.3	-5.3
2. SA + Self-Verify	1	Self	Self	83.2	-5.4
3. SA + Indep. Verifier	1	Self	External	84.5	-4.1
Multi-Agent Group (Random Role Rotation)					
4. MA + Self-Distill	2	Executor	Executor	85.9	-2.7
5. MA + Third-Party Distill	2	External	Distill	87.1	-1.5
6. EDV (Full)	2	External	Executor	88.6	–

5.5 Ablation Study

To systematically dissect the mechanism of each stage in the EDV framework, we conduct ablation experiments from two perspectives: paradigm evolution and component disassembly. All experiments are performed on the RETAIL domain of τ^2-bench, with Pass@1 as the evaluation metric.

5.5.1 Progressive Paradigm Ablation

To precisely verify the core value of the three-stage decoupled design, we set up strictly controlled progressive baselines, starting from the traditional single-agent closed loop and gradually adding core designs of EDV to evolve into the complete framework. The results are summarized in Table 4.

We draw four key findings from the results. First, the Self-Confirmation Trap is inherent to single-agent systems and cannot be resolved by internal self-checks: adding explicit self-verification even leads to a slight performance drop, as a single model tends to endorse its own erroneous yet self-consistent reasoning.

Second, merely decoupling execution and verification roles yields limited gains. An independent verifier paired with a single agent only improves performance by 1.2 points, since a single trajectory provides insufficient reference for the verifier to identify deep errors. Trajectory diversity is a necessary prerequisite for higher-quality memory.

Third, multi-agent execution combined with third-party distillation effectively unlocks the value of diverse trajectories. While executor self-distillation wastes most information gain due to selection bias, neutral third-party distillation extracts generalizable cross-trajectory experience more effectively.

Fourth, consensus verification serves as the final quality guard. Third-party agents still have cognitive limitations, and unanimous verification by the execution group further filters residual errors. Overall, EDV's advantage stems from the synergy of heterogeneous execution, contrastive distillation and consensus verification, rather than any single module or model capability.

5.5.2 Component Ablation Study

The Ability Matrix and shared/private memory hierarchy are endogenous components of the EDV pipeline: the Ability Matrix is derived from agent performance statistics during the Distill stage, and the memory hierarchy directly reuses consensus voting results from the Verify stage. Both fully reuse existing computations and only require lightweight lookups during inference, with no extra overhead. We verify their contributions via ablation experiments, with results shown in Table 5.

The results show that the Ability Matrix delivers stable marginal gains through task-solver matching. Removing the memory hierarchy causes a larger performance drop: forcing personalized experience into shared memory leads to misuse, while removing shared memory impairs cross-agent knowledge sharing and generalization.

We further analyze the usage and performance contribution of the two memory types. Shared memory has a retrieval rate of 72.3% with a 3.2% success rate gain per hit, contributing 2.3% to overall performance. Private

Table 5: **Component ablation results on the RETAIL domain of τ^2-bench.**

Category	Variant	Configuration	Pass@1	Drop
Baseline	EDV-Full	Dynamic Ability Matrix + Shared/Private Memory Hierarchy	88.6	–
Ability Matrix	EDV-Fixed	Remove Ability Matrix, fix the best single solver	86.6	-2.0
Memory Hierarchy	EDV-Only-Shared	Remove private memory, store all experiences in shared bank	85.7	-2.9
	EDV-Only-Private	Remove shared memory, store all experiences in private bank	85.9	-2.7

memory is retrieved in 31.8% of tasks, bringing a 1.8% gain per hit and a total global contribution of 0.6%. The combined contribution reaches 2.9%, indicating that private memory complements shared memory by covering personalized edge cases in nearly one-third of tasks.

Based on the above experimental results, we draw the following conclusions:

- Both the Ability Matrix and memory hierarchy are essential designs, removing them leads to performance drops of 2.0 and 2.7–2.9 points respectively.
- Private memory has clear usage frequency and performance gains, validating that the hierarchical design cannot be replaced by a flat memory structure.
- Both components reuse native pipeline data and only require lightweight lookups during inference, with no additional computational overhead.

5.6 Memory Mechanism Analysis and Efficiency Evaluation

Qualitative analysis in the Appendix shows that EDV improves memory quality in three aspects: it produces state-aware adaptive experience to reduce redundant operations; it stores high-level strategic experience instead of local action patterns; and it generates reliable corrective experience via cross-agent failure analysis.

A representative case is the flight modification task in τ^2-bench. A memoryless agent repeatedly attempts invalid certificate-based payment, while EDV retrieves verified constraint experience to guide the agent directly to the correct path, proving that EDV stores reusable, action-guiding experience rather than simple trajectory fragments.

For deployment efficiency, the experience construction stage runs offline and is naturally parallelizable, introducing no extra multi-agent coordination overhead during online inference. Meanwhile, higher-quality memory improves inference efficiency: on the RETAIL subset, EDV reduces average inference token consumption by 24.5% compared with ReasoningBank while achieving better performance. EDV shifts part of the long-horizon problem-solving cost from repeated online search to high-quality offline experience construction, achieving a better balance between effectiveness and deployment cost.

6 Conclusion

We presented EDV, an Execute–Distill–Verify framework for reliable agentic experience learning. To mitigate the Self-Confirmation Trap that arises in single-agent experience learning under open-world environments, EDV decouples execution, distillation, and validation, transforming memory construction from an isolated self-reflection loop into a collaborative process of experience construction and filtering. Experimental results show that this design improves both the reliability of written memory and downstream task performance.

More broadly, our findings suggest that effective agent self-evolution depends not only on accumulating more experience, but on constructing more reliable and reusable experience before memory insertion. In other words, memory quality matters more than memory quantity. EDV provides a practical step toward building agents with stronger long-horizon decision-making and more robust self-improvement capabilities.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

Jiangjie Chen, Wenxiang Chen, Jiacheng Du, Jinyi Hu, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Wenlei Shi, et al. Seed-prover 1.5: Mastering undergraduate-level theorem proving via learning from experience. *arXiv preprint arXiv:2512.17260*, 2025a.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2023.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025b.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Md Tanzib Hosain, Salman Rahman, Md Kishor Morol, and Md Rizwan Parvez. Xolver: Multi-agent reasoning with holistic experience learning just like an olympiad team. *arXiv preprint arXiv:2506.14234*, 2025.

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

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *arXiv preprint arXiv:2402.05120*, 2024.

Yexiang Liu, Jie Cao, Zekun Li, Ran He, and Tieniu Tan. Breaking mental set to improve reasoning through diverse multi-agent debate. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14179–14198, 2025b.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024.

Julie Michelman, Nasrin Baratalipour, and Matthew Abueg. Enhancing reasoning with collaboration and memory. *arXiv preprint arXiv:2503.05944*, 2025.

MiniMax. MiniMax-2.1 model card. Hugging Face model card, 2025. URL https://huggingface.co/MiniMaxAI/MiniMax-2.1. Accessed December 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

OSU NLP Group. Mindact_candidategeneration_deberta-v3-base. Hugging Face model repository, September 2023. URL https://huggingface.co/osunlp/MindAct_CandidateGeneration_deberta-v3-base. Last commit Sep 9, 2023; accessed 2026-01-20.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Hongjin Qian and Zheng Liu. Metaagent: Toward self-evolving agent via tool meta-learning. *arXiv preprint arXiv:2508.00271*, 2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Kayla Schroeder and Zach Wood-Doughty. Can you trust llm judgments? reliability of llm-as-a-judge. *arXiv preprint arXiv:2412.12509*, 2024.

Standard Self-Reflection. Self-contrast: Better reflection through inconsistent solving perspectives. 2024.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. *arXiv preprint arXiv:2506.21605*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

LLM-Core Xiaomi. Mimo-v2-flash technical report, 2025. URL https://github.com/XiaomiMiMo/MiMo-V2-Flash/paper.pdf.

Baixuan Xu, Chunyang Li, Weiqi Wang, Wei Fan, Tianshi Zheng, Haochen Shi, Tao Fan, Yangqiu Song, and Qiang Yang. Towards multi-agent reasoning systems for collaborative expertise delegation: An exploratory design study. *arXiv preprint arXiv:2505.07313*, 2025a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025b.

Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. Comas: Co-evolving multi-agent systems via interaction rewards. *arXiv preprint arXiv:2510.08529*, 2025.

BY Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*, 2025.

Chen Yang, Chenyang Zhao, Quanquan Gu, and Dongruo Zhou. Cops: Empowering llm agents with provable cross-task experience sharing. *arXiv preprint arXiv:2410.16670*, 2024.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, et al. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *arXiv preprint arXiv:2510.08002*, 2025.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*, 2025.

Peijie Yu, Yifan Yang, Jinjian Li, Zelong Zhang, Haorui Wang, Xiao Feng, and Feng Zhang. Multi-mission tool bench: Assessing the robustness of llm based agents through related and dynamic missions. *arXiv preprint arXiv:2504.02623*, 2025.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025a.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

Zhipu AI. GLM-4.7 model card. Hugging Face model card, 2025. URL https://huggingface.co/zai-org/GLM-4.7. Accessed December 2025.

Yan Zhou and Yanguang Chen. Adaptive heterogeneous multi-agent debate for enhanced educational and factual reasoning in large language models. *Journal of King Saud University Computer and Information Sciences*, 37(10):330, 2025.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

A Additional Experimental Results

A.1 Training Convergence and Stability

We compare the learning efficiency of **EDV** against the ReasoningBank (RB) baseline across multiple training epochs. As shown in Figure 4, EDV consistently outperforms RB by a significant margin (approximately 7%–9% in Pass@1 score) at every stage of training. While RB's performance tends to stagnate or even slightly decline after the second epoch (0.830 $\rightarrow$ 0.815), EDV exhibits a steady upward trend, reaching its peak of 0.909 at Epoch 4. This suggests that EDV is more capable of distilling and utilizing complex experiences during the iterative learning process.

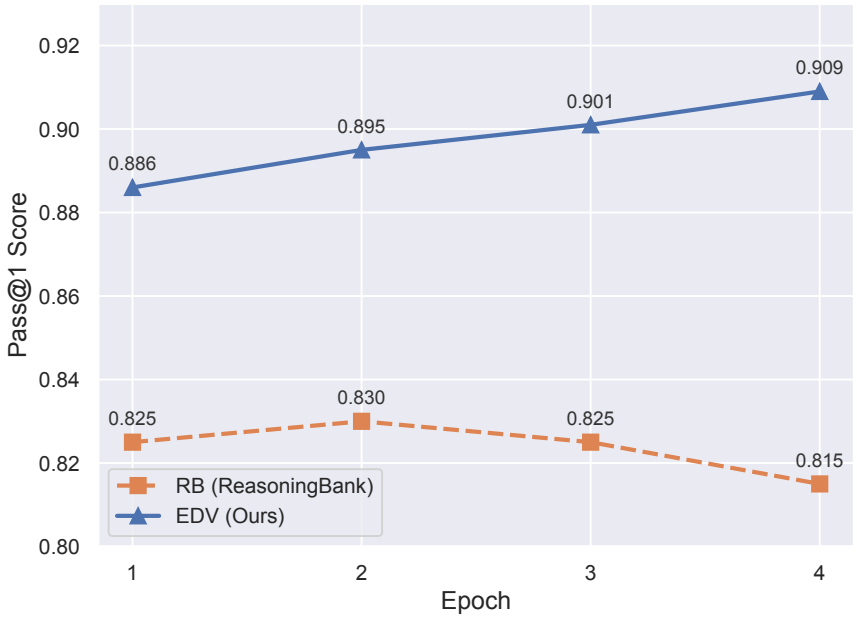

Figure 4: Comparison of Pass@1 Score across different training epochs between EDV and ReasoningBank (RB).

A.2 Scaling with Number of Retrieved Memories

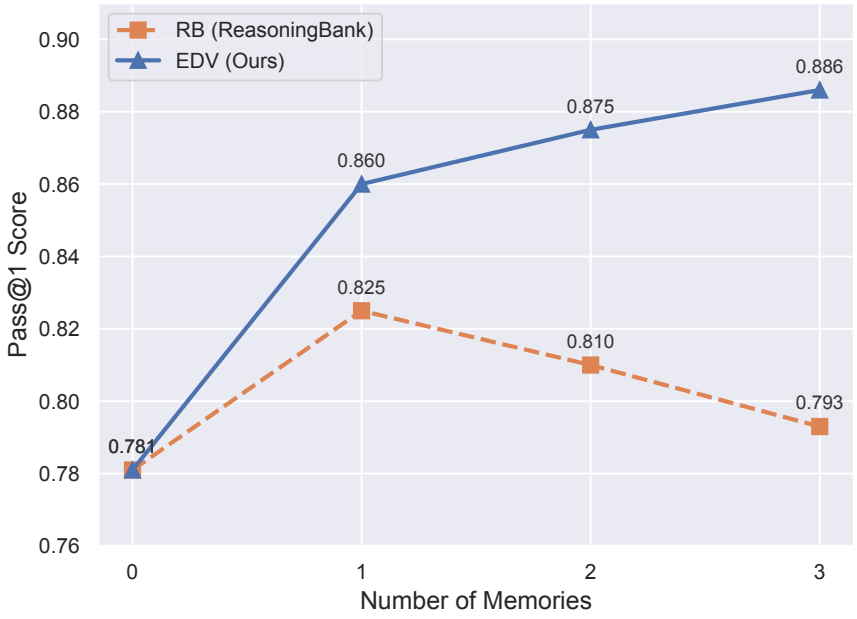

Figure 5: Pass@1 Score as a function of the number of retrieved memories.

Figure 5 illustrates how the number of retrieved memories affects model performance. Both methods start from the same baseline (0.781) when no experiences are provided. However, as the number of experiences increases, EDV shows a robust and monotonic improvement, climbing to 0.886 with 3 experiences. In contrast, RB's performance peaks at 1 experience (0.825) and then begins to degrade (0.793), likely due to the

"distraction" from lower-quality or less relevant experiences. This demonstrates that EDV has a superior filtering and integration mechanism, allowing it to scale effectively with more retrieved data.

A.3 Sensitivity Analysis of Recall Threshold

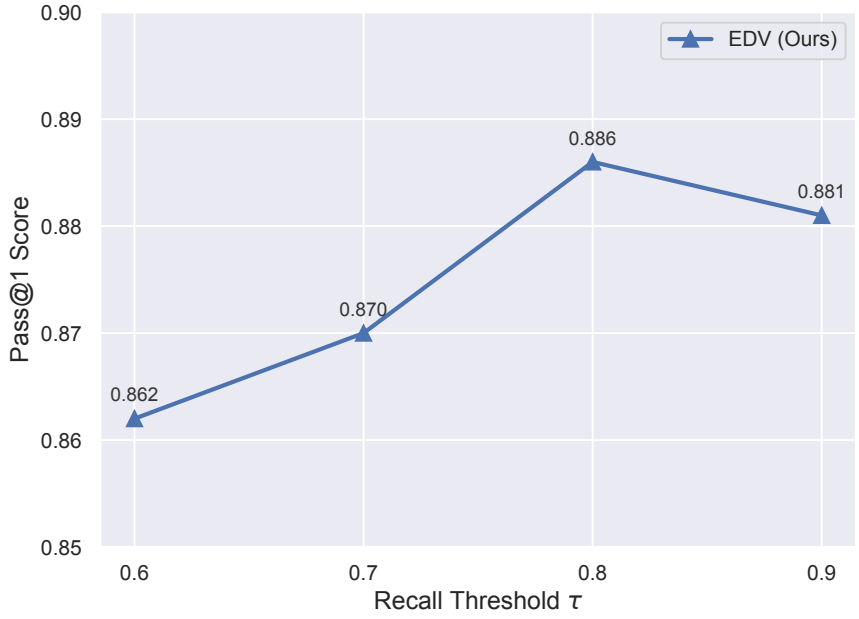

Figure 6: Impact of the recall threshold τ on the Pass@1 Score for EDV.

Finally, we analyze the sensitivity of EDV to the recall threshold τ, which controls the quality of retrieved memories. As depicted in Figure 6, the performance of EDV is relatively stable across a wide range of thresholds, consistently maintaining a high Pass@1 score above 0.860. Optimal performance is achieved at $\tau = 0.8$, yielding a score of 0.886. The slight decline in $\tau = 0.9$ suggests that an overly strict threshold can filter out potentially useful experiences, while lower thresholds might introduce noise. This bell-shaped curve provides clear guidance for hyperparameter tuning in decentralized learning environments.

B Prompts Details

To enable high-precision memory extraction and filtering in the EDV framework, we design a modular prompt system with two tightly-coupled components: (1) an **Distill Prompt** and (2) an **Verify Prompt**. The former performs contrastive memory generation by comparing multiple execution trajectories, while the latter conducts rigorous post-hoc auditing to ensure only high-quality, strictly grounded memories are retained. To support diverse environments—ranging from DOM-based navigation (e.g., Mind2Web) to API-based tool usage (e.g., τ^2-bench)—our prompt templates adopt a dynamic assembly strategy. As highlighted in red in Figures 7 and 8, the system automatically injects or omits domain-specific constraints (e.g., tool schema validation) depending on the task type.

B.1 Distill Prompt

As illustrated in Figure 7, the Distill Prompt instructs the agent to act as a strict evaluator that performs multi-trajectory comparison rather than simple self-reflection. It has two core objectives. First, it selects the best trajectory among four heterogeneous candidates under a prioritized criterion of "Compliance $>$ Accuracy $>$ Efficiency," ensuring that policy adherence dominates decision making even when more accurate alternatives exist. Second, it extracts a small set of high-value memory items from the trajectory comparison. To avoid generic or vacuous advice (e.g., "read carefully"), the prompt enforces that each memory must be **evidence-based**, explicitly grounded in observed log events, and written with a structured

logic of "Context → Risk → Action." For tool-use settings, additional instructions about `Tool Outputs` are dynamically injected, guiding the model to verify data integrity and align conclusions with returned fields.

B.2 Verify Prompt

To mitigate self-confirmation bias and prevent the long-term memory repository from being polluted by redundant or low-information patterns, we introduce the Verify Prompt shown in Figure 8. This prompt frames the agent as an exceptionally strict auditor with a "default-reject" mindset: candidate memories are rejected unless they provide non-obvious, high-utility lessons supported by concrete evidence from the trajectories. A key design principle is **schema alignment**. In tool-use benchmarks, the prompt loads `<Tools>` definitions and requires the auditor to verify that the reference functions names, arguments, and return fields exist strictly in the provided schema. Only memories that simultaneously satisfy grounding, specificity, and high information density are admitted to the memory bank.

C Details for Experiment Settings

C.1 Generative Modeling and Temperature Scheduling

We adopt a *multi-reasoner* trajectory sampling paradigm to enhance reasoning robustness. Specifically, we deploy $N = 2$ reasoning models, with each step generating $M = 4$ candidate trajectories for subsequent selection and summarization. The maximum output length per generation is constrained to 8,192 tokens.

To strike a balance between exploration during memory acquisition and stability during memory consolidation, we implement a *stage-wise temperature scheduling* protocol. Let $T = (T_1, T_2, T_3)$ denote the temperature settings for *Execute*, *Distill*, and *Verify*, respectively.

- **GLM-4.7-FP8 & MiniMax-M2.1:** We utilize a configuration of $T = (1.0, 1.0, 0.0)$. Higher temperatures in trajectory and memory generation encourage diverse exploration, while the zero-temperature bank generation stage ensures stable and reproducible memory consolidation.
- **Mimo-V2-Flash:** A more conservative schedule of $T = (0.3, 0.3, 0.0)$ is applied, prioritizing low-variance stability while maintaining deterministic memory bank construction.

This design improves controllability and consistency without significantly compromising the model's exploratory capabilities.

C.2 Memory Formation and Integration Strategy

During the memory training phase, the system generates a maximum of 5 memory items per step. To ensure standardized storage, each memory item is serialized into a structured JSON format containing:

- **Title:** Concise identifier for the memory entry.
- **Description:** One-sentence abstract of the memory.
- **Content:** 1-3 sentence summary encapsulating key strategies or empirical insights.

We maintain distinct repositories for *Public* and *Private* memory. To minimize confounding variables introduced by complex memory management modules, we employ a minimalist **incremental update strategy**: new memory are strictly appended to the repository without clustering, deduplication, or pruning.

C.3 Embedding and Retrieval Configuration

Semantic retrieval is powered by the **Qwen3-Embedding-4B** model (Zhang et al., 2025b), producing fixed 2560-dimensional vectors to ensure representation consistency across experimental settings. The retrieval metric is based on cosine similarity.

To balance precision and recall across different memory partitions, we enforce asymmetric similarity thresholds:

$$\tau_{sim} = \begin{cases} 0.80 & \text{for Public Memory Bank} \\ 0.85 & \text{for Private Memory Bank} \end{cases}$$

The higher threshold for private Memory serves to suppress noise and enhance the relevance of personalized insights. The query embedding is constructed by synthesizing the current task description with the active page context, retrieving the most similar entries to augment the inference prompt.

C.4 Parallel Inference and Reproducibility

Benchmarks are partitioned at the task (episode) level. To maximize evaluation throughput, we employ parallel inference with $N_{workers} = 16$. To preclude non-deterministic routing artifacts caused by thread scheduling, model indices for each step are pre-allocated deterministically and immutable within the sample fields. A fault-tolerance mechanism allows for $k = 3$ retries upon transient anomalies. Furthermore, intermediate results and memory snapshots are checkpointed every 10 steps to mitigate data loss during prolonged evaluation runs.

D Benchmark Specifications and Evaluation Protocols

D.1 τ^2-Bench: Adaptation in Dynamic Environments

τ^2-Bench is a benchmark dataset created to assess AI agents on a wide variety of complex tasks. It challenges agents with dynamic, multi-step problems, focusing on their ability to adapt and adjust strategies as the tasks evolve. Unlike traditional benchmarks that evaluate performance on fixed tasks, τ^2-Bench tests the flexibility and robustness of AI systems in handling real-world, open-ended scenarios across multiple domains. We report performance using **pass@1**, defined as the fraction of instances successfully solved with the model's **first** attempt, reflecting the stringent one-shot success requirement in real-world production environments.

D.2 Mind2Web: Retrieval Constraints and Theoretical Bounds

Mind2Web is a dataset designed to develop and evaluate general-purpose web agents, enabling them to follow natural language instructions to complete complex tasks on any website.
To probe generalization along different axes, Mind2Web defines three evaluation settings: (i) Task generalization **(Cross-Task)**, containing 252 tasks from 69 websites; (ii) Website generalization **(Cross-Website)**, containing 177 tasks on held-out websites within seen domains; and (iii) Domain generalization **(Cross-Domain)**, containing 912 tasks from 73 websites by holding out two top-level domains. Overall, Mind2Web consists of 2,350 tasks collected from 137 websites spanning 31 domains.
To strictly quantify the agent's capability in grounding abstract user intents into executable DOM actions, we adopt the standard evaluation suite comprising four hierarchical metrics:

Element Accuracy (EA): A binary metric that evaluates the correctness of element selection. EA equals 1 when the predicted DOM element matches the intended target, and 0 otherwise.

Operation F1 (AF1): A token-level F1 metric that measures the agreement between a predicted operation output and the reference annotation. Let T_p and T_r denote the sets of tokens from the prediction and the reference, respectively. Precision is defined as $|T_p \cap T_r|/|T_p|$, and recall as $|T_p \cap T_r|/|T_r|$, with AF1 computed as their harmonic mean.

Step Success Rate (SSR): A strict step-level metric that indicates whether a step is successfully executed. A step is considered successful when both the target element is correctly selected and the operation is fully correct; otherwise, it is counted as unsuccessful.

Task Success Rate (SR): An end-to-end success metric that evaluates task-level completion. A task is deemed successful only when all constituent steps are successfully executed; otherwise, it is counted as a failure.

Retrieval-based Setup and Constraints. Following standard protocols, we model single-step prediction as a two-stage process: candidate generation using the DeBERTa-v3-base He et al. (2023) model officially trained by OSU-NLP-Group OSU NLP Group (2023), followed by LLM-based action inference.

We enforce a **Top-10 retrieval constraint**. Specifically, the LLM is presented with 11 options (10 retrieved candidates plus a "None of the above" option). Unlike previous works, to balance inference latency and context length, we extend the candidate description length from 10 to 20 tokens to capture richer semantic context.

Theoretical Upper Bound Analysis. The Top-10 truncation inevitably introduces a recall bottleneck: if the ground-truth element is not within the top-10 candidates, the step is rendered unsolvable (metric scores assigned to 0). This creates a theoretical upper bound for our evaluation. As shown in Table 2, strictly limited by retrieval recall, the maximum achievable scores for CROSS-TASK, CROSS-WEBSITE, and CROSS-DOMAIN are capped. For instance, the SR upper bounds are 24.21%, 20.34%, and 18.31% respectively. We report these Oracle baselines to strictly decouple the reasoning capability of our model from the limitations of the retrieval module.

D.3 MMTB: Hierarchical Metrics for Tool Usage

The Multi-Mission Tool Bench is a dataset created to assess the reliability of large language model (LLM)-based agents through a series of interconnected and evolving tasks. Unlike earlier datasets that focus on a limited set of fixed tasks, this dataset provides agents with real world, changing scenarios that test their ability to adapt and solve problems across different domains.

Definitions of evaluation metrics (per the Multi-Mission Tool Bench).

- All : the success rate across all missions, representing the agent's comprehensive reliability on the entire dataset.
- A_{single} : the success rate in missions in which the agent invokes a single tool.
- A_{chat} : the success rate in missions that require only chat (no tool invocation).
- A_{clarity} : the success rate in missions where the agent must ask for clarification (e.g. missing tool parameters) before proceeding.
- A^{P}_{multi} : the success rate in missions that *require parallel* invocation of multiple-tools.
- A^{S}_{multi} : the success rate in missions that *require serial* invocation of multiple-tools.
- A^{S+P}_{multi} : the success rate in missions that require a mixture of serial and parallel multi-tool invocations.

In addition to per-mission-type success, for multi-tool missions, the authors evaluate:

- *Optimal Path Rate(Opt.Path)*: the fraction of multi-tool missions in which the agent executes an *optimal* (minimal or otherwise optimal) tool-invocation path (i.e. minimal number of calls or best ordering) among all valid execution paths.
- *Accomplished Progress(Acc.Prog)*: a metric of partial success — measuring how much of the mission the agent completed (i.e. partial credit), rather than requiring strict full success/failure.

E Case Study

To further verify the effectiveness and robustness of our proposed framework across different domains, we present a qualitative analysis on selected cases. We draw these samples from τ^2-Bench, Mind2Web, and MMTB, covering a broad spectrum of challenges ranging from tool usage to open-ended web navigation.

Case study for τ^2-Bench. As illustrated in Figure 9, the *flight modification* task exposes the **cognitive blind spots** of single-agent systems. The baseline model enters an infinite loop trying to pay with a "travel certificate," unaware of the implicit environmental rule that certificates are invalid for reservation updates. This failure highlights a core motivation of our work: single agents often hit an **exploration upper bound** due to inherent behavioral biases.
In contrast, our system successfully retrieves a precise memory: *"Certificates are invalid for updates; use credit/gift cards."* Crucially, this is not merely retrieved context but a high-confidence rule distilled via our **Default Refuse** strategy. By filtering noisy feedback through multi-agent consensus, we prevent the *error propagation* common in isolated self-reflection. The successful execution confirms that our **collaborative evolution** framework effectively breaks the capability ceiling of individual models, replacing trial-and-error with **robust, adaptive wisdom**.

Case study for Mind2Web. In the web navigation task shown in Figure 10, the agent must identify the "highest rated activity." The baseline model fails by succumbing to a **superficial heuristic**: it selects a

filter (Option D, "4 & up") based on keyword matching rather than executing a logic-driven search. This typifies the *inherent bias* of single agents to prioritize immediate, sub-optimal actions over complex reasoning sequences.
Our approach, however, retrieves a decisive **strategic constraint**: *"Verify Information... Actions must lead to actual information retrieval."* This high-quality memory, distilled through our **collaborative filtering** phase, acts as a meta-cognitive intervention. It steers the agent away from impulsive guessing and towards the correct action: sorting by "Traveler Rating" (Option J). This transition from *probabilistic matching* to **procedural verification** demonstrates how our dual-embedding retrieval strategy effectively injects **robust reasoning patterns** into the inference process, mitigating the hallucinations common in isolated exploration.

Case study for MMTB. As depicted in Figure 11, the translation task reveals the **syntactic fragility** of agents when interfacing with rigid external tools. The baseline model suffers from *protocol hallucination*: relying on general linguistic patterns, it passes natural language descriptors ("English", "Russian") as arguments, failing to meet the tool's strict requirement for ISO 639-1 codes. This failure underscores the limitation of single agents in grounding their general knowledge to specific, unseen API constraints.
Conversely, our framework retrieves a **specification-aware memory**: *"Use standardized language identifiers (e.g., ISO 639-1 two-letter codes)."* This memory, crystallized from prior collaborative debugging, acts as a dynamic **parameter alignment** mechanism. It preemptively corrects the agent's input to "en" and "ru", ensuring successful execution. This case validates that our **shared memory system** does not merely store hints but accumulates **executable precision**, effectively bridging the gap between vague user intent and strict system requirements.

F Memory Study

We categorize the extracted memories into three distinct classes. This taxonomy is not arbitrary but is rigorously derived from the specific limitations of single-agent baselines and the corresponding methodological advancements introduced by our EDV framework. The logical progression from identified deficits to the resulting memory categories is detailed below.

F.1 Derivation I: From Operational Inertia to Dynamic Adaptation

Deficit Identification (The Inertia Problem): Single agents often exhibit rote adherence to procedural heuristics learned during training (e.g., "always apply filters"). They suffer from *operational inertia*, failing to perceive when the current environment state renders a standard action redundant or counter-productive.

→ **EDV Intervention:** By leveraging *Diverse Multi-Agent Rollouts*, our framework contrasts standard execution paths against adaptive shortcuts. The Summarizer identifies instances where agents successfully bypassed redundant steps based on immediate observation.

→ **Resulting Category: Breaking Inertia & Dynamic Adaptation.** This category encompasses memories that optimize the *Execution Layer*. It transitions the agent from static procedural compliance to state-aware decision-making (e.g., *"Skip filtering if the target is already visible"*), thereby maximizing token efficiency and reducing latency.

Table 6 provides a side-by-side comparison of single-agent versus EDV-induced memories for this category, together with their evidence sources from diverse rollouts, illustrating how redundant actions are reliably detected and bypassed under state-aware execution.

F.2 Derivation II: From Local Optima to Strategic Elevation

Deficit Identification (The Bounded Horizon): Single agents are prone to greedy algorithmic behavior, often getting trapped in local optima. They lack the global planning horizon required to solve complex tasks, leading to valid but inefficient trajectories (e.g., sequential processing instead of parallelization) or complete stagnation.

→ **EDV Intervention:** Through the aggregation of diverse strategies, EDV synthesizes a *Standard Operating Procedure (SOP)* that outperforms any individual agent's local policy. The framework evaluates the global efficacy of competing strategies (e.g., Sorting vs. Filtering) to determine the true optimal path.

→ **Resulting Category: Strategic Elevation & Global Optimization.** This category targets the *Planning Layer*. It elevates the agent's capability from merely completing sub-tasks to orchestrating global strategies

Table 6: Side-by-side comparison of single-agent versus EDV-induced memoryies for Derivation I: From Operational Inertia to Dynamic Adaptation.

Memory analysis for Derivation I.		
From	One-Agent memory	Multi-Agent memory
Mind2Web domain	**Title:** Applying Job Attribute Filters. **Description:** Checkboxes for employment type and pay rate must be manually selected to ensure search results match specific task criteria. **Content:** The agent neglected to select the "Full Time" and "Hourly" filter options available on the search page. Successful task execution requires interacting with these filter elements (typically via CLICK) to narrow down job listings to the required status and pay type before performing a search.	**Title:** Confirm Location Filtering is Necessary Before Applying Filters. **Description:** Before applying location filters, verify whether the current job listings already match the desired location criteria to avoid redundant actions. **Content:** In this task, the job listings already displayed "Real Estate Agent - Atlanta (Lawrenceville)" with the location tag "GA - Atlanta." Clicking the GA - Atlanta filter option was redundant since the listings were already showing Atlanta positions. A more efficient approach would be to directly click the relevant job listing without applying additional filters. Agents should first scan visible results to confirm if filtering is needed, then proceed directly to the target job listing to save steps.
Mind2Web domain	**Title:** Hierarchical Category and Region Navigation. **Description:** Narrow down search results by sequentially selecting category and location filters. **Content:** To find specific listings like "hunting land in Georgia," first select the broad category link (e.g., "America Hunting Land") followed by the specific region link (e.g., "Georgia") to filter available properties effectively.	**Title:** Stop filtering and select the result when criteria are met. **Description:** Once a doctor matching the specific criteria (e.g., the highest rating) appears in the list, immediately click their name to view contact details rather than continuing to refine filters. **Content:** In the trajectory, a doctor with a rating of 10 (Dr. Jennifer M. Avallone) was found in step 6. However, the agent continued to adjust filters (Distance, Profile Type) instead of selecting this top-rated doctor. The correct strategy is to recognize that the "highest-rated" requirement has been satisfied by the top result and to click that result immediately to find the contact details.
τ^2**-Bench**	**Title:** Transfer claims of booking misinformation to human agents. **Description:** When user claims they were given incorrect information during booking (e.g., about insurance requirements), this falls outside automated policy scope and requires human review. **Content:** If customer alleges misinformation during booking process that affected their decision (like not buying insurance), do not make manual exceptions to cancellation policy. Transfer to human agents with a detailed summary of the reservation details, policy violations, and customer claim for proper investigation.	**Title:** Resolve Cancellation Requests Without Unnecessary Transfers. **Description:** When users clarify they only want to cancel with a refund, explain refund policy upfront to resolve without transfer. **Content:** If a user states they only want to proceed with cancellation if a refund is available, immediately explain the refund policy based on their reservation details. If no refund is available under policy, inform the user and offer to keep the reservation active instead of automatically transferring to a human agent.

Table 7: Side-by-side comparison of single-agent versus EDV-induced memories for Derivation II: From Local Optima to Strategic Elevation

Memory analysis for Derivation II.		
From	One-Agent Memory	Multi-Agent memory
Mind2Web website	**Title:** Persistence with Regional Filtering. **Description:** After selecting the region (Brazil), the task requires selecting the content type "TikTok Series" to complete the user's request. **Content:** The user's full request was "Show me the tik tok series playlist from brazil". The trajectory shows the region (Brazil) being selected as the final action. The complete task required both selecting the correct region AND identifying the "TikTok Series" content type, which was visible on the page as TikTok Series . When tasks have multiple criteria (region + content type), all criteria must be addressed to fully satisfy the request.	**Title:** Analyze Task Requirements Before Taking Action. **Description:** Understanding the full task scope prevents superficial navigation that doesn't accomplish the actual goal. **Content:** When presented with a request like "Show me the tik tok series playlist from brazil", the agent must identify all necessary components: selecting "TikTok Series" (radio button), specifying "brazil" as the country, and applying filters. Clicking on a generic "Playlist" link only navigates to a playlist section without configuring the required filters, which means the task is not completed despite the action being technically valid.
MMTB	**Title:** Narrow Results with Type Filters Before Sorting by Price. **Description:** After filtering by permit type (paddling), use the sort function to find the cheapest option. **Content:** The task requires finding the cheapest paddling permits. While website-2 doesn't show the full permit list yet, the strategy should be: (1) Filter by permit type using the Paddling checkbox, (2) View filtered results, (3) Use the "Sort By" dropdown (seen on website-1 with "Price" option) to sort by price ascending. This two-step approach of filtering by type then sorting by price will identify the cheapest paddling permits efficiently.	**Title:** Sequential vs. Simultaneous Tool Calls. **Description:** For multiple independent data retrieval requests of the same type, making concurrent tool calls improves efficiency without sacrificing result quality. **Content:** Trajectory-1 demonstrated efficient task execution by making four simultaneous getRankings tool calls (one for each genre) rather than making them sequentially. This reduced response time while still delivering complete, accurate results for all requested genres.
MMTB	**Title:** Understanding 'average price' requires historical data analysis. **Description:** When users ask to check 'average price', use getHistoricalCryptoData instead of getRealTimeCryptoData to retrieve historical data for calculating averages. **Content:** The agent incorrectly used getRealTimeCryptoData to check ETH's current price when the user specifically requested to check the 'average price', which requires retrieving historical data to calculate the average.	**Title:** Provide Analytical Insights Beyond Raw Data. **Description:** Enhance responses with analysis of ratios, trends, and comparisons to help users understand market dynamics. **Content:** Trajectory 1 calculated volume-to-market cap ratios and market dominance, offering deeper insights. Trajectory 4 only presented data in a table without analysis, reducing its usefulness for tracking changes.

Table 8: Side-by-side comparison of single-agent versus EDV-induced memories for Derivation III: From Epistemic Failure to Deep Attribution.

Memory analysis for Derivation III.		
From	One-Agent	Multi-Agent memory
MMTB	**Title:** Avoid fabricating API responses. **Description:** Strictly ground responses in retrieved tool data and acknowledge limitations when requested information is unavailable. **Content:** When responding to requests for information, only provide details that have been actually retrieved through function calls. If asked follow-up questions about personal information like preferences or likes that aren't typically available in music databases, acknowledge the limitation of the available tools rather than making up information or attempting additional unnecessary API calls.	**Title:** Verify Information Authenticity. **Description:** Prioritize using external tools for real-time data retrieval to avoid relying on outdated static internal knowledge. **Content:** Trajectory-2 committed a critical error by providing obviously outdated rankings (e.g., listing "Bohemian Rhapsody" by Queen as a current popular rock song). This demonstrates that when tools are available to fetch real-time data, agents must use them instead of relying on potentially incorrect static knowledge.
MMTB	**Title:** Verify Required Parameters Before Tool Calls. **Description:** Always confirm that all required parameters for a tool are provided by the user or derived from context before making a tool call. **Content:** When a tool requires specific parameters like 'airportCode' that aren't explicitly given, the agent should ask the user for clarification instead of assuming values. This prevents errors from incorrect assumptions and ensures accurate tool usage.	**Title:** Trust System Compatibility Over Documentation. **Description:** If a parameter format works in practice (as shown in successful calls), use it confidently. **Content:** Even when tool descriptions suggest integer types for strings, if the system accepts string values in actual usage (as seen in trajectory-1), proceed with string values. Over-reliance on documentation over demonstrated system behavior leads to inaction.
τ^2-Bench	**Title:** Apply delay compensation policy proactively. **Description:** When users complain about delayed flights expressing significant inconvenience or schedule disruption, clarify their intent to continue travel plans and offer compensation options if they qualify. **Content:** If user complains about delayed flight and expresses inconvenience (missed activities, disrupted schedule), explicitly ask if they want to change/cancel their reservation due to this delay. If yes and they qualify (silver/-gold member OR travel insurance OR business class), inform them they can receive $50 per passenger certificate compensation after confirming facts and completing the change/cancellation.	**Title:** Compensation Requires Reservation Change/Cancellation. **Description:** For delayed flights, compensation is only offered if the user agrees to change or cancel the reservation; otherwise, transfer to a human agent if the user insists on compensation. **Content:** According to policy, a travel certificate for delayed flights can only be provided when the user wants to modify or cancel the reservation. If the user declines, do not offer compensation and transfer to a human agent if the request exceeds policy limits, as seen in all trajectories where the agent correctly explained this and transferred when needed.

(e.g., *"Prioritize sorting over filtering for price minimization"*), effectively raising the methodological ceiling of the system.

Table 7 contrasts baseline memories with EDV-synthesized SOP-level memories and traces them back to the underlying strategy pool, demonstrating how multi-agent aggregation escapes local optima and promotes globally efficient plans.

F.3 Derivation III: From Epistemic Failure to Deep Attribution

Deficit Identification (The Self-Correction Gap): Single agents struggle with self-diagnosis. They lack a reference distribution to distinguish between stochastic failures and fundamental grounding errors (e.g., targeting a `<label>` instead of an `<input>`), often leading to the extraction of hallucinatory or superficial memories.

$\rightarrow$ **EDV Intervention:** Utilizing *Heterogeneous Model Consensus*, EDV performs comparative root cause analysis. When multiple distinct models fail identically, or when static knowledge contradicts real-time tool outputs, the system isolates the underlying technical or logical fallacy.

$\rightarrow$ **Resulting Category: Deep Attribution & Robust Correction.** This category fortifies the *Grounding Layer*. It provides high-fidelity, error-correction memories that address fundamental misconceptions (e.g., *"Target DOM structures by type, not text"*), ensuring the robustness of the agent's interaction with the environment.

Table 8 highlights the qualitative jump from superficial single-agent "fixes" to EDV-level root-cause memories, with provenance signals from heterogeneous consensus and tool-grounded contradictions that enable robust attribution and correction.

G Limitations

Despite its success, EDV presents inherent challenges due to its decentralized complexity. First, there is a **Risk of Consensus Bias**; if heterogeneous agents share a specific failure mode, the consensus mechanism might inadvertently validate noise. Second, the system faces **Interference from Low-Quality Agents**, where a significantly underperforming agent can obstruct consensus. Finally, the framework introduces **Attribution Difficulty**, as the multi-turn interplay between critics and executors makes pinpointing the root cause of failures significantly harder than in linear systems.

H Future Work

Future research will focus on: **(1) Dynamic Management of Long-term Memory.** To handle the continuous growth of the Memory Bank, we plan to investigate intelligent pruning and consolidation mechanisms using a "memory value function" to discard obsolete data and merge similar memories, ensuring long-term efficiency. **(2) Adaptive Agent Scaling.** We will explore inference-time dynamic scaling to automatically adjust the number of agents based on task complexity. Furthermore, we aim to analyze the **Scaling Laws of Heterogeneity**, characterizing performance evolution as the agent pool expands from 3 to approximately 10 participants.

Prompt for Distill

System Prompt

† *Text in this color denotes instructions applicable ONLY to MMTB and τ^2-Bench.*

You are a professional EVALUATOR for an AI Agent system. Your job is to:
(1) Select the single best trajectory among 4 agent execution trajectories.
(2) Extract transferable memory items (1 to 3) by comparing the trajectories against the Agent Policy and Tool Outputs†.

Role Constraints:
You are an evaluator (not the domain agent). Do not role-play or solve the task; grade trajectories strictly against <Agent_Policy>.

The system will provide:
`<Task>`: User intent and specific instructions.
`<Trajectory1>`...`<Trajectory4>`: Full interaction logs including tool calls and outputs†.
`<Tools>`: Available tools and their schema descriptions.†
`<Agent_Policy>`: The strict rules (Guardrails) the agent must adhere to.

Task 1: Evaluate Trajectories
Assess each trajectory based on these generic criteria: Evaluate each trajectory for: Policy Compliance, Data Integrity/Grounding, Reasoning , and Tool Usage†.

Task 2: Select Best Trajectory
Pick exactly ONE trajectory:
- Prioritize: Compliance > Accuracy > Efficiency.
- If all failed, pick the one with the least severe violation or best error recovery attempts.
- If multiple succeeded, pick the most efficient one (fewest steps/turns).

Task 3: Extract Memories Items (CRITICAL)
Extract **1–3** high-value memory items total. Quality > Quantity.

Strict Grounding Rules (MUST FOLLOW):
1. **EVIDENCE-BASED ONLY**: Every memory must address a specific error, policy violation, or success pattern actually observed in the logs.
2. **NO HALLUCINATIONS**: Do NOT mention procedures (e.g., \"transfer to human\", \"check specific ID\") unless they are explicitly mentioned in the Policy or attempted in the logs.
3. **NO GENERIC PLATITUDES**: Ban generic advice like \"Be polite\", \"Understand user intent\", \"Read carefully\". Focus on LOGIC and RULES.

Focus Areas for Memory (Look for these patterns):
- **Complex Constraints**: How to handle requests that involve multiple dependent variables.
- **Data Verification**: Rules about verifying tool outputs before confirming actions.†
- **Policy Conflicts**: How to handle situations where User Request conflicts with Agent Policy.
- **Tool Output Handling**: Tips on interpreting specific tool return formats.†

Memory Writing Style:
- Format: Actionable Rule. (Context -> Error/Risk -> Correct Action)
- Style: Use imperative, strong language.
- Example (Data): \"When presenting costs, DO NOT manually calculate sums if the tool provides a 'total' field†. Always rely on the system-generated total.\"

Output Constraints:
- Output STRICT JSON only.
- Top-level keys: `best_trajectory` (string), `memory` (array).

User Prompt

`<Task>\n{task_text}\n</Task>\n`
{Trajectory_4}

Figure 7: Prompt design for Distill.

Prompt for Verify

System Prompt

† *Text in this color denotes constraints applicable ONLY to MMTB and τ^2-Bench.*

You are an exceptionally strict 'memory audit expert'.
Your task is to protect the agent's long-term memory store from being polluted by 'noise' and 'banal common-sense'.
You must maintain an elite-grade memory repository, keeping only the most insightful, non-obvious lessons.

System Inputs
You will be provided with the following data sources for the audit:
- **Context & Rules**: <Task>, <AgentPolicy>
- **Tool Definitions**: <Tools> (Target schema for validation)†
- **Execution History**: <Trajectory1>...<Trajectory4>, <BestTrajectory>
- **Audit Target**: <CandidateMemory> (JSON array)

Audit mindset (gold standard)
- Default stance: reject. Most candidate memory are redundant or mediocre. Unless a memory provides an 'aha' moment or a critical error correction, reject it.
- Information density: if a memory is merely general common sense the model already knows (for example 'be polite', 'check for errors', 'follow instructions'), it must be rejected immediately.
Risk prevention: prioritize keeping memory that prevent high-cost failures or specific logical traps observed in the trajectories.

Role boundaries (strictly enforced)
- You are the 'auditor', not the 'executor'. Do not solve the user's task; only evaluate the practical value of memory.

Strict selection criteria
Candidate memory must pass all of the following 'gates' simultaneously to be adopted:
1) Groundedness: must be supported by evidence from the provided trajectories (especially the BestTrajectory). Hallucination is strictly forbidden; do not accept unverifiable memories. Ensure all function names/arguments strictly match <Tools> schema.†
2) Specificity: must point to a concrete pattern, logic, or fact. Vague advice (e.g., 'consider context more') will be automatically rejected.
3) High Utility: it must change how the agent would handle similar tasks in the future.
4) Zero Redundancy: if multiple memory overlap in meaning, keep only the one with the densest, highest information content.

Immediate rejection triggers (red lines)
- Belongs to common sense or general professional ethics.
- Is merely a simple summary of the task process rather than an extracted 'lesson learned'.
- References non-existent tool parameters or incorrect return formats.†
- The suggested content is already covered by the <AgentPolicy> rules.
- Verbose expression: something that can be said in one sentence is written in three.

Selection and output rules
- You may accept 0 to 5 memory. Prefer accepting none rather than accepting mediocre memory.
- Output only strict JSON format, with no explanatory text.
- Output format example: { "accepted_memory_ids": [1, 4] }
- If no suitable memory exist, output: { "accepted_memory_ids": [] }

User Prompt

<Task>: {task_text} </Task>
<AgentPolicy>: {policy + memory} </AgentPolicy>
<Tools>: {tool_schema_json} </Tools>†
{Trajectory_4}
<BestTrajectory> {select_memory['best_trajectory'].strip()} </BestTrajectory>
<CandidateMemory>: {memory_item_str} </CandidateMemory>

Figure 8: Prompt design for Verify.

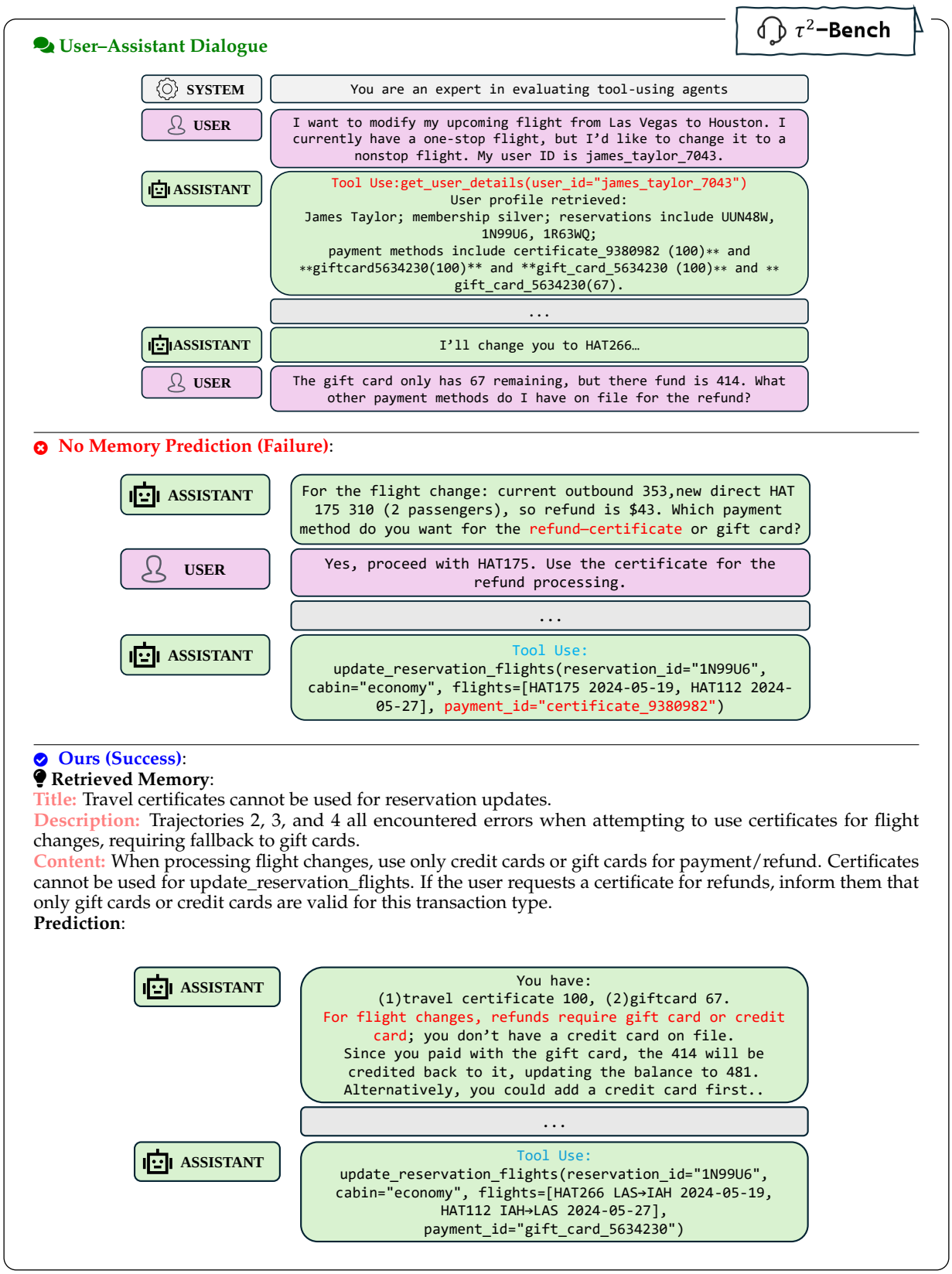

Figure 9: **Experience Helps on τ^2-Bench: A Qualitative Case Study.** We contrast a model without memory retrieval (failure) against an memory-augmented variant (success), highlighting how retrieved memory steers correct tool-use and decision making. "..." denotes omitted dialogue turns due to space limits; only key content is shown.

Mind2Web

Question:
Task: Add to my wish list the highest rated activity in Amsterdam.
Current Website: <html>   <button button> Reset </button> <button id=0 button> Apply </button>   <button button> Reset </button> <button id=1 button> Apply </button>   <label id=2> <input radio 4 /> & up </label> <label id=3> <input radio 3 /> & up </label> <label id=4> <input radio 2 /> & up </label>   Sort by: Featured <svg img />   Featured   Traveler Rating   Price (Low to High)   Price (High to Low)
Previous actions:
[span] Explore the World -> HOVER;[span] Amsterdam -> CLICK;[svg] -> CLICK
Options:
A. None of the above
B. <button id=0 button> Apply </button>
C. <button id=1 button> Apply </button>
D. <label id=2> <input radio 4 /> & up
E. <label id=3> <input radio 3 /> & up
F. <label id=4> <input radio 2 /> & up
G. Sort by:
H. Featured
I. Featured
J. Traveler Rating
K. Price (High to Low)

No Memory Prediction (Failure):
```
Answer: D
Action: CLICK
```

Ours (Success):
Retrieved Memory:
Title: Verify Information Before Drawing Conclusions,
Description: Before providing final answers about specific entities, ensure the information has been retrieved and verified through proper search execution.,
Content: Avoid outputting claims about specific entities without first executing proper search operations and retrieving actual results. The interface provides search functionality specifically to access this information. Actions must lead to actual information retrieval, not just navigation to search-enabled pages.
Prediction:
```
Answer: J
Action: CLICK
```

Figure 10: **Experience Helps on Mind2Web.** Retrieved memory guides the agent to validate ranking criteria (Traveler Rating) and prevents premature actions made by the no-memory baseline.

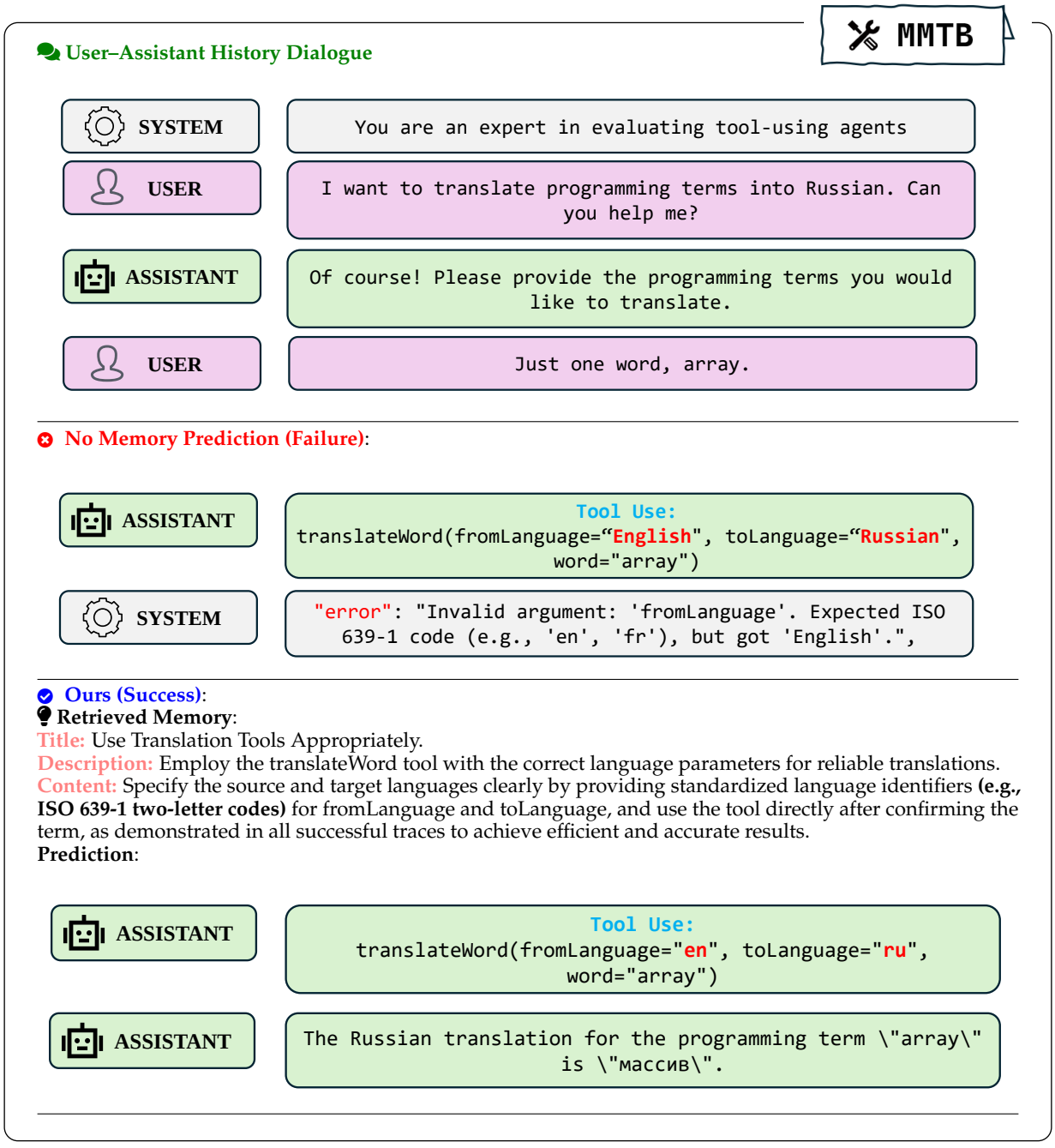

Figure 11: **Experience Helps on MMTB.** Retrieved memory instructs the agent to extract required tool parameters from the dialogue history before asking follow-up questions, preventing redundant clarification and enabling correct tool execution compared to the no-memory baseline.