AGENTSTREAM: HOW WELL DO SELF-EVOLVING LLM AGENTS PERFORM UNDER STREAMING TASKS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents can self-evolve by continually improving from their own accumulated experience. However, existing studies predominantly adopt independent evaluation. Consequently, the behavior of self-evolving agents in realistic streaming settings, where agents adapt to diverse and complex task streams, remains poorly understood. To address this gap, we introduce AgentStream, a unified framework that evaluates self-evolving agents spanning diverse evolution components by organizing agentic benchmarks into a configurable task stream. Specifically, we instantiate *Isolated*, *Sequential*, and *Interleaved* streaming scenarios at test time, which progressively vary the scope and domain composition of the stream. Over these scenarios, we combinatorially evaluate five representative self-evolving methods across three frontier foundation models, disentangling how model capability, method architecture, and streaming scenario jointly shape self-evolution. Our results show that self-evolution reliability varies across streaming scenarios, the benefit of self-evolution is gated by model capability and is non-monotonic in model strength, and no single method dominates across models and scenarios. These findings offer concrete guidance for selecting self-evolving methods across models and streaming scenarios. Overall, we advocate that self-evolving agents should be evaluated under realistic task streams rather than isolated single-task settings. Our code is available at `https://anonymous.4open.science/r/self-evolving-0C4A`.

1 INTRODUCTION

Large language model (LLM) agents are shifting from static, deploy-once systems toward adaptive architectures that continuously learn from their own accumulated experience during deployment, a paradigm broadly termed self-evolving (Gao et al., 2026; Lin et al., 2026b; Fang et al., 2025). Depending on which component of the agent is updated, recent methods evolve the prompt context, structured memory, reusable skill libraries, or an integrated harness (Zhang et al., 2026c; Xu et al., 2025; Yang et al., 2026b; Ouyang et al., 2026b; Lin et al., 2026a). Such self-evolution is expected to produce increasingly capable agents over time, making it essential to understand how agent capabilities evolve across the diverse task streams encountered in realistic deployment.

However, as illustrated in Figure 1, existing agentic benchmarks and self-evolving studies predominantly adopt independent evaluation, where each task is solved in isolation and performance is aggregated without any cross-task state (Trivedi et al., 2024; Patil et al., 2025; Jimenez et al., 2024; Barres et al., 2025; Zhang et al., 2026c; Xu et al., 2025). While a few studies move toward streaming evaluation (Wu et al., 2024; Wei et al., 2026; Ouyang et al., 2026b), they stream each benchmark independently under a single evolution component, leaving how different evolution components transfer cross-domain experience systematically unexamined. Consequently, it remains unclear whether the improvements reported for self-evolving agents persist once they are deployed in realistic streaming settings, where tasks may span diverse domains and arrive without clear domain boundaries or supervision. Answering this question requires an evaluation that considers the foundation model, the self-evolving method, and the structure of the task stream jointly rather than any one of them in isolation, which the prevailing independent evaluation is inherently unable to provide.

To this end, we propose AgentStream, a unified streaming evaluation framework that organizes tasks from multiple agentic benchmarks into a configurable stream, ranging from within-domain to cross-

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

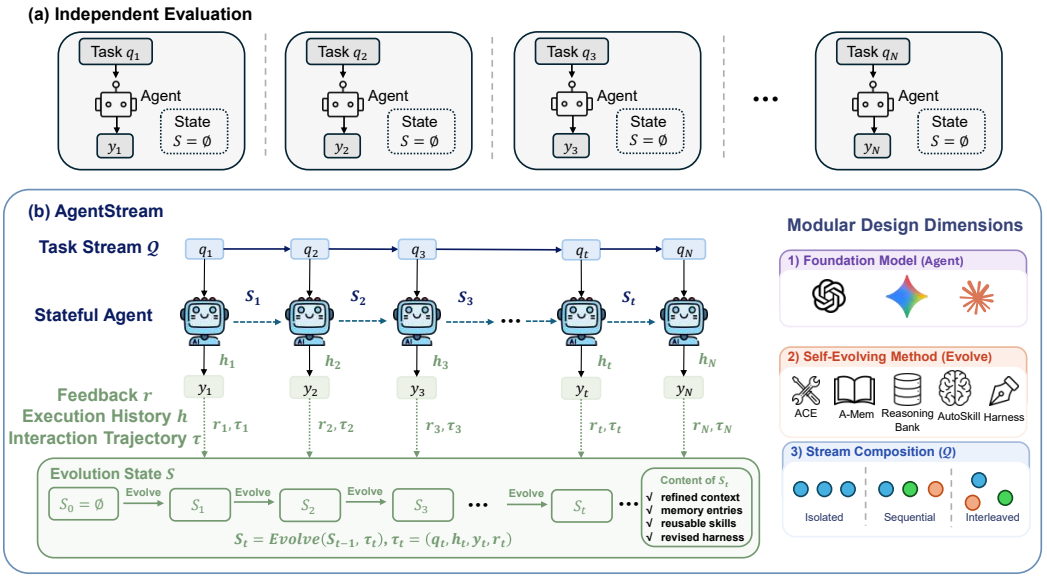

Figure 1: Independent evaluation vs. AgentStream framework. (a) The prevailing paradigm solves each task in isolation without cross-instance experience accumulation. (b) AgentStream: a stateful agent evolves through self-generated feedback over a configurable task stream, with foundation model, self-evolving method, and stream composition as modular design dimensions.

domain composition, and evaluates self-evolving methods whose evolution components span context, memory, skill, and integrated harness. AgentStream performs a combinatorial evaluation across models, self-evolving methods, and streaming scenarios, enabling us to decouple the contributions of model capability and method architecture under different stream structures. Concretely, we instantiate three test-time streaming scenarios that progressively vary in scope and domain composition: *Isolated*, where the agent evolves independently within each benchmark; *Sequential*, where the agent processes the benchmarks in a fixed order, transferring its evolution state across benchmarks; and *Interleaved*, where tasks from all benchmarks are shuffled into one unified stream. Across these scenarios, we evaluate five representative self-evolving methods, ACE (Zhang et al., 2026c), A-Mem (Xu et al., 2025), ReasoningBank (Ouyang et al., 2026b), AutoSkill (Yang et al., 2026b), and Harness (Lin et al., 2026c;a), across three frontier foundation models, GPT-5.4 (Singh et al., 2025), Gemini 3.1 Pro (Google DeepMind, 2026), and Claude Opus 4.7 (Anthropic, 2026), over six agentic benchmarks covering diverse capabilities, including AppWorld (Trivedi et al., 2024), BFCL (Patil et al., 2025), BrowseComp-Plus (Chen et al., 2026), HLE (Phan et al., 2025), SWE-bench Verified (Jimenez et al., 2024), and Tau2 (Barres et al., 2025).

Our study reveals three findings. **First**, self-evolution is not uniformly beneficial, and its reliability varies with the streaming scenario. *Isolated* is the most reliable, while *Interleaved* generally outperforms *Sequential* despite its more heavily mixed stream. **Second**, the benefit of self-evolution is gated by model capability. The weakest model exhibits negative evolution gains, and the benefit is non-monotonic in model strength, with a mid-capability model gaining more than a stronger one. **Third**, no single method dominates. Context-integrated methods favor *Isolated* while retrieval-based methods favor *Interleaved*, and the optimal method varies across models rather than transferring between them.

Overall, this work contributes AgentStream, the first framework that unifies agentic benchmarks into a configurable streaming evaluation and systematically assesses self-evolution along the streaming scenario, method, and model dimensions. Through a combinatorial analysis, we investigate how streaming scenario, model capability, and method architecture shape the performance of self-evolution. We further distill these observations into actionable guidance: applying self-evolution to sufficiently capable models, favoring context-integrated methods under within-domain streams and retrieval-based methods under cross-domain streams, and selecting the method per model rather than assuming a universal choice. We hope AgentStream encourages future research to evaluate self-evolving agents beyond isolated single-task settings and toward realistic task streams.

2 Related Work

Self-Evolving Agents. Existing self-evolving methods primarily differ in the component of the agent system that undergoes evolution. Memory-based methods focus on the accumulation, retrieval, and reorganization of experience, enabling agents to selectively retain useful knowledge while pruning irrelevant information over successive tasks (Zhang et al., 2026b; Xu et al., 2025; Chhikara et al., 2025). Skill-based methods operate at a higher level of abstraction, discovering and composing reusable capability modules into growing libraries that compound across diverse domains (Yang et al., 2026b; Ouyang et al., 2026a). Harness-level methods evolve the scaffolding infrastructure surrounding the base model, including system prompts, tool configurations, memory modules, and skill libraries (Lee et al., 2026; Lin et al., 2026a). Mechanistically, these methods are driven by reinforcement learning (Xia et al., 2026; Zhang et al., 2026a), evolutionary algorithms (Novikov et al., 2025; Xu et al., 2026), trajectory distillation (He et al., 2026; Cai et al., 2026), or gradient-analogy optimization that treats textual feedback as differentiable signals (Yuksekgonul et al., 2025; Yang et al., 2026a). However, existing work evaluates on isolated benchmarks without systematically comparing how different self-evolving methods and models behave under a controlled setting, while recent analyses further reveal that self-evolution can degrade or fail to transfer across domains (Shao et al., 2026; Fang et al., 2025). Our work provides a unified streaming evaluation framework, analyzing the roles of self-evolving methods and models under different stream structures.

Agentic Benchmarks. Agentic benchmarks have been developed to evaluate LLM agents across diverse and complex environments. Interactive web and application benchmarks require agents to complete long-horizon tasks by navigating stateful interfaces and executing actions that alter the environment state (Yao et al., 2022; He et al., 2024; Trivedi et al., 2024; Xie et al., 2024). Tool-use benchmarks assess structured function calling, measuring whether agents can select, compose, and invoke external APIs with correct arguments (Patil et al., 2025; Bandi et al., 2026; Li et al., 2026a). Software engineering benchmarks evaluate agents on repository-level tasks such as resolving real-world code issues under executable test suites (Merrill et al., 2026; Wang et al., 2026). Skill-oriented benchmarks measure how well agents acquire and reuse modular capabilities across heterogeneous tasks (Li et al., 2026b; Zhang et al., 2026d). Knowledge-intensive reasoning benchmarks further stress deep retrieval and expert-level problem solving over frontier knowledge (Phan et al., 2025; Wei et al., 2025). However, these benchmarks predominantly evaluate agents on each task in isolation, and although a few adopt streaming evaluation tailored for self-evolving agents (Wu et al., 2024; Wei et al., 2026; Wang et al., 2025), they remain limited to single-benchmark streams and evaluate only one evolving component. In contrast, our framework unifies agentic benchmarks into more complex streaming scenarios covering both within-domain and cross-domain adaptation, and evaluates diverse evolving components including prompt, memory, skill, and harness.

3 The AgentStream Framework

To study how LLM agents improve through experience accumulation over realistic task streams, we design AgentStream, an evaluation framework centered on self-evolution, in which an agent processes a stream of tasks and, after each attempt, distills the interaction trajectory into persistent experience (e.g., refined context, memory entries, reusable skills, or revised harness). Building on this framework, we instantiate the problem setup and three streaming scenarios that systematically vary the scope and composition of the task stream, as illustrated in Figure 2.

3.1 Problem Setup

We consider a test-time learning setting (Ouyang et al., 2026b; Wu et al., 2024) in which a task stream $\mathcal{Q} = \{q_1, q_2, \ldots, q_N\}$ is presented to the agent, with each task revealed only after the preceding one is completed. The agent is parameterized by a foundation model $\mathcal{M}$ and equipped with a self-evolving method that maintains an evolution state S. This state aggregates the experience distilled from the first t interactions, initialized as $S_0 = \varnothing$. For each task q_t, the agent performs a multi-step interaction conditioned on its current evolution state, generating an execution history $h_t = \{(a_{t,i}, o_{t,i})\}_{i=1}^{L_t}$ of action-observation pairs over L_t steps and a final solution $y_t = \mathcal{M}(q_t, h_t \mid S_{t-1})$. The agent subsequently updates $S_t = \textsc{Evolve}(S_{t-1}, \tau_t)$ by reflecting on the interaction trajectory $\tau_t = (q_t, h_t, y_t, r_t)$, where r_t denotes the self-generated feedback. No-

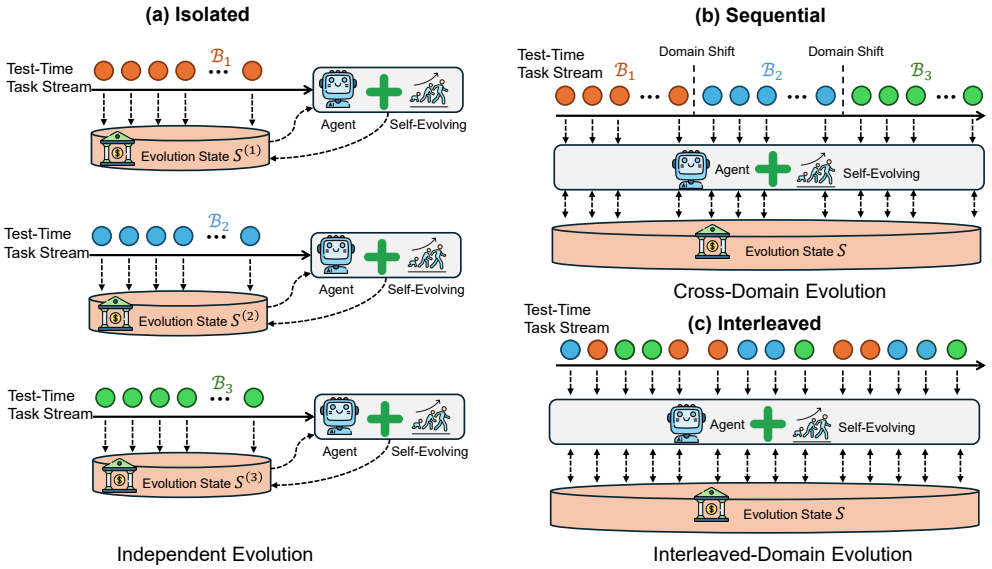

Figure 2: Three representative streaming scenarios instantiated in AgentStream. (a) *Isolated*: each benchmark maintains a separate evolution state. (b) *Sequential*: a single evolution state carries across ordered benchmark boundaries. (c) *Interleaved*: tasks from all benchmarks are shuffled into one unified stream with a shared evolution state.

tably, no ground-truth labels are accessible at test time, and the evolution relies entirely on feedback intrinsic to the interaction, such as execution outcomes and reflective self-evaluation. We quantify the benefit of self-evolution through the evolution gain:

$$\Delta = \mathrm{Perf}(\mathcal{M},\, \mathcal{Q},\, S) \;-\; \mathrm{Perf}(\mathcal{M},\, \mathcal{Q},\, \varnothing), \tag{1}$$

where $\mathrm{Perf}(\mathcal{M}, \mathcal{Q}, \varnothing)$ denotes the baseline in which the same model solves each task with $S_t = \varnothing$ for all t. A positive Δ indicates net improvement from self-evolution, while a negative Δ signals that accumulated experience introduces interference.

3.2 Streaming Scenarios

Given a set of K benchmarks $\{\mathcal{B}_1, \ldots, \mathcal{B}_K\}$, each contributing a task subset $\mathcal{Q}^{(k)}$, we instantiate three evaluation settings that systematically vary how the task stream $\mathcal{Q}$ is composed and how the evolution state S_t is scoped across benchmarks.

Isolated. Each benchmark $\mathcal{B}_k$ is assigned an independent agent instance with its own evolution state $S_t^{(k)}$. The task stream for each agent instance is simply $\mathcal{Q}^{(k)}$, and no experience transfers across benchmarks. This setting isolates intra-domain evolution and measures how effectively a self-evolving method accumulates useful experience within a single task domain.

Sequential. A single agent processes all benchmarks in a fixed order $\mathcal{B}_1 \to \mathcal{B}_2 \to \cdots \to \mathcal{B}_K$, with its evolution state S_t retained across benchmark boundaries. The resulting task stream is the concatenation $\mathcal{Q} = \mathcal{Q}^{(1)} \oplus \mathcal{Q}^{(2)} \oplus \cdots \oplus \mathcal{Q}^{(K)}$. This setting evaluates whether experience acquired in earlier domains facilitates or interferes with performance on later domains, testing forward transfer in a sequential curriculum.

Interleaved. A single agent receives a unified stream $\mathcal{Q} = \mathrm{shuffle}(\mathcal{Q}^{(1)} \cup \cdots \cup \mathcal{Q}^{(K)})$ in which tasks from all benchmarks are interleaved in randomized order. The agent maintains one shared evolution state S_t that is updated by tasks from all benchmarks indiscriminately. This setting tests the agent's ability to retrieve domain-relevant experience while suppressing cross-domain interference under maximal task diversity.

Rather than being ordered by expected difficulty, the three streaming scenarios are designed to decouple distinct aspects of self-evolution, each capturing a challenge that self-evolving agents may encounter in deployment. Crossing these streaming scenarios with the self-evolving methods and foundation models enables a controlled analysis of how each factor contributes to self-evolution.

4 EXPERIMENTAL SETTINGS

Models. We evaluate three frontier foundation models spanning different families and scales: GPT-5.4-medium (Singh et al., 2025), Gemini 3.1 Pro-medium (Google DeepMind, 2026), and Claude Opus 4.7-high (Anthropic, 2026). This diversity allows us to analyze how model capacity influences test-time evolution across self-evolving methods.

Tasks. Our evaluation suite comprises six diverse benchmarks covering a broad spectrum of agentic capabilities: (1) **AppWorld** (Trivedi et al., 2024): interactive coding tasks requiring multi-app workflow execution with API understanding and dynamic environment interaction. (2) **BFCL** (Patil et al., 2025): multi-step function calling across domains, testing context-dependent tool use with missing parameters and long contexts. (3) **BrowseComp-Plus** (Chen et al., 2026): deep-research tasks requiring iterative web retrieval and information synthesis over a controlled document corpus. (4) **HLE** (Phan et al., 2025): expert-level academic reasoning across dozens of disciplines, challenging frontier models with questions resistant to memorization. (5) **SWE-bench Verified** (Jimenez et al., 2024): real-world software engineering tasks requiring codebase understanding, fault localization, and patch generation. (6) **Tau2** (Barres et al., 2025): conversational agent tasks in dual-control environments where the agent and user act in a shared system, testing coordination and communication.

Self-Evolving Methods. We select five representative methods that span context, memory, skill, and integrated harness evolution: (1) **ACE** (Zhang et al., 2026c): evolves agent context through modular generation, reflection, and curation of prompts. (2) **A-Mem** (Xu et al., 2025): organizes agent memories using Zettelkasten-style indexing and linking, refining contextual representations as new experience is integrated. (3) **ReasoningBank** (Ouyang et al., 2026b): distills generalizable reasoning strategies from successful and failed trajectories into structured memory items. (4) **AutoSkill** (Yang et al., 2026b): extracts reusable skills from interaction experience through a lifecycle of extraction, structured representation, iterative refinement, and versioned maintenance. (5) **Harness**: inspired by Lin et al. (2026c;a), we implement a harness evolution method that jointly maintains and updates system prompts, skills, and experience memory through reflection and revision.

Implementation Details. Our evaluation infrastructure is built on Exgentic (Bandel et al., 2026), a framework that standardizes communication between heterogeneous agent interfaces and benchmarks. All self-evolving methods are adapted to operate within this framework under a test-time setting. For text embedding, we adopt the all-MiniLM-L6-v2 model across all experiments. We sample $N = 50$ tasks from each benchmark, where AppWorld uses the test-challenge split, BFCL uses the multi-turn base split, and Tau2 uses the telecom domain. Task-level performance is evaluated by the native scoring pipeline of each benchmark. All judge models and user simulator models required by the benchmarks are unified to GPT-5.4. To account for ordering effects, we run three random seeds that shuffle task order while keeping the task set fixed. In the *Sequential* setting, the benchmark order is AppWorld $\rightarrow$ BFCL $\rightarrow$ BrowseComp+ $\rightarrow$ HLE $\rightarrow$ SWE $\rightarrow$ Tau2. Across the three streaming scenarios, the within-benchmark task order is held constant to ensure comparability.

5 RESULTS

Table 1 reports the main results across models, self-evolving methods and streaming scenarios, averaged over three random seeds. Per-seed results are provided in Tables 11 to 13.

5.1 DOES SELF-EVOLUTION HELP AT ALL, AND UNDER WHICH SCENARIO?

Marginalizing over models and methods in Table 2, we find that self-evolution is not uniformly beneficial and underperforms the vanilla baseline in a substantial share of configurations. Given this variability, we read Table 1 along the streaming-scenario axis alone, asking under which scenario self-evolution most reliably attains a positive evolution gain.

***Isolated* is the most reliable streaming scenario for self-evolution.** As shown in Table 2, *Isolated* achieves the highest positive rate of 75.6% and the largest average evolution gain of +1.37%, substantially outperforming both *Sequential* at 62.2% positive rate and +0.75% gain, and *Interleaved* at 62.2% and +0.90%. This advantage is further validated by its highest Top-1 rate of 38%. When all tasks come from a single benchmark, the experience stream remains distributionally coherent. Each solved task contributes directly relevant knowledge for subsequent tasks, and no filtering or retrieval

Table 1: Main results (%) of self-evolving methods across three streaming scenarios, averaged over three random seeds. Red background indicates scores below the model's vanilla baseline, and **blue** marks the best scenario within each self-evolving method.

Methods	Mode	AppWorld	BFCL	BrowseComp+	HLE	SWE	Tau2	Avg
GPT-5.4	Vanilla	44.6	66.0	50.0	2.0	62.0	50.0	45.8
ACE	*Isolated*	39.1±5.9	68.0±12.2	46.7±2.3	5.3±1.2	60.0±3.5	63.3±6.1	**47.1**
	Sequential	38.9±5.9	64.0±5.3	48.7±6.1	7.3±1.2	62.7±5.0	45.3±22.3	44.5
	Interleaved	32.9±1.4	58.7±3.1	44.7±2.3	6.0±2.0	63.3±5.0	48.7±3.1	42.4
A-Mem	*Isolated*	41.2±9.5	68.7±2.3	52.0±7.2	8.7±3.1	65.3±1.2	50.0±36.2	47.7
	Sequential	39.9±4.5	66.7±6.4	50.7±3.1	10.0±2.0	60.0±5.3	62.7±18.9	48.3
	Interleaved	37.9±4.8	65.3±3.1	52.7±4.2	8.7±2.3	62.0±3.5	76.0±21.6	**50.4**
ReasoningBank	*Isolated*	40.9±1.8	62.7±7.0	45.3±2.3	10.0±2.0	59.3±5.0	46.7±12.2	44.2
	Sequential	42.9±0.6	62.7±2.3	47.3±3.1	14.7±4.6	62.7±4.6	36.0±14.4	44.4
	Interleaved	38.6±3.5	63.3±5.0	48.7±4.2	12.0±2.0	56.7±6.4	53.3±4.2	**45.4**
AutoSkill	*Isolated*	40.1±2.8	70.7±1.2	48.0±3.5	4.7±1.2	57.3±7.0	36.0±14.0	42.8
	Sequential	38.2±1.6	72.0±3.5	43.3±5.0	6.7±1.2	58.0±4.0	44.0±13.1	43.7
	Interleaved	40.0±1.7	72.0±5.3	46.0±2.0	4.7±1.2	60.7±1.2	40.0±8.7	**43.9**
Harness	*Isolated*	37.4±2.3	64.0±8.7	46.0±2.0	7.3±5.8	61.3±2.3	57.3±12.2	**45.6**
	Sequential	35.1±1.6	66.7±2.3	45.3±1.2	6.7±1.2	57.3±6.1	54.0±5.3	44.2
	Interleaved	37.0±4.9	65.3±5.0	50.0±5.3	6.7±1.2	57.3±2.3	46.0±5.3	43.7
Gemini 3.1 Pro	Vanilla	41.8	58.0	34.0	52.0	64.0	90.0	56.6
ACE	*Isolated*	43.2±1.9	62.7±13.3	44.0±5.3	50.0±0.0	64.0±0.0	90.7±9.5	59.1
	Sequential	41.2±3.2	72.0±9.2	43.3±5.0	52.0±0.0	68.0±0.0	95.3±3.1	**62.0**
	Interleaved	40.7±2.0	52.7±13.0	44.0±6.0	50.7±1.2	64.7±1.2	96.7±3.1	58.3
A-Mem	*Isolated*	41.2±6.7	54.0±25.0	50.0±4.0	52.0±2.0	62.0±5.3	92.7±4.6	**58.7**
	Sequential	40.4±3.5	54.7±2.3	46.0±2.0	48.7±4.2	60.7±1.2	91.3±2.3	57.0
	Interleaved	40.4±2.6	56.7±16.7	47.3±7.6	50.7±1.2	60.7±2.3	90.7±6.1	57.8
ReasoningBank	*Isolated*	42.8±3.1	54.7±4.6	50.0±5.3	48.7±4.2	62.0±4.0	95.3±2.3	58.9
	Sequential	42.0±0.8	56.0±2.0	44.0±2.0	51.3±1.2	66.0±5.3	94.7±3.1	59.0
	Interleaved	44.7±1.9	48.7±3.1	52.7±4.2	52.7±2.3	67.3±3.1	94.7±2.3	**60.1**
AutoSkill	*Isolated*	43.1±2.3	62.7±3.1	46.7±4.2	51.3±3.1	64.7±5.0	83.3±5.0	58.6
	Sequential	45.5±1.5	64.7±5.8	41.3±4.2	50.0±3.5	62.7±3.1	86.7±5.0	58.5
	Interleaved	44.7±1.2	65.3±1.2	41.3±6.1	50.0±2.0	66.0±2.0	87.3±2.3	**59.1**
Harness	*Isolated*	39.9±5.0	74.7±3.1	42.7±4.6	52.0±3.5	68.0±4.0	90.0±5.3	**61.2**
	Sequential	41.9±1.0	54.7±23.1	35.3±2.3	50.7±2.3	64.7±4.2	91.3±4.2	56.4
	Interleaved	40.9±2.0	74.7±4.2	38.0±16.4	51.3±2.3	63.3±3.1	90.7±6.1	59.8
Claude Opus 4.7	Vanilla	41.2	86.0	70.0	38.0	68.0	80.0	63.9
ACE	*Isolated*	47.5±4.2	85.3±1.2	72.0±2.0	33.3±4.2	70.7±2.3	93.3±3.1	**67.0**
	Sequential	47.5±0.9	82.7±3.1	68.7±1.2	38.7±2.3	70.7±4.2	91.3±5.0	66.6
	Interleaved	46.7±0.8	80.7±4.2	68.0±0.0	34.0±7.2	70.0±3.5	72.0±8.7	61.9
A-Mem	*Isolated*	48.7±2.5	85.3±1.2	71.3±1.2	27.3±1.2	71.3±6.4	90.7±7.6	65.8
	Sequential	48.1±0.4	82.7±1.2	68.0±4.0	32.7±3.1	69.3±4.2	96.0±3.5	**66.1**
	Interleaved	49.3±0.9	81.3±1.2	67.3±2.3	32.0±2.0	72.7±6.1	86.0±6.9	64.8
ReasoningBank	*Isolated*	44.0±1.1	84.0±0.0	70.0±2.0	36.0±2.0	69.3±4.2	90.0±0.0	65.6
	Sequential	41.8±1.4	85.3±1.2	71.3±3.1	35.3±3.1	66.7±3.1	85.3±1.2	64.3
	Interleaved	45.1±0.9	84.7±1.2	70.0±3.5	39.3±3.1	68.7±1.2	88.7±3.1	**66.1**
AutoSkill	*Isolated*	41.9±1.3	86.0±0.0	73.3±1.2	36.7±1.2	68.0±5.3	82.0±3.5	64.7
	Sequential	41.8±0.5	86.0±0.0	72.7±2.3	37.3±2.3	63.3±1.2	86.7±7.6	64.6
	Interleaved	40.6±2.4	86.7±1.2	73.3±1.2	38.0±3.5	67.3±3.1	86.7±2.3	**65.4**
Harness	*Isolated*	44.8±1.0	84.0±2.0	68.7±1.2	34.0±5.3	69.3±4.2	90.7±3.1	65.3
	Sequential	42.6±0.8	82.7±2.3	71.3±4.2	36.0±2.0	72.0±0.0	74.0±28.0	63.1
	Interleaved	46.1±2.1	85.3±1.2	69.3±1.2	37.3±1.2	69.3±2.3	87.3±9.2	**65.8**

gating is needed to avoid cross-domain interference. This makes *Isolated* the lowest-risk streaming setting regardless of method architecture.

The two cross-domain scenarios do not follow the expected difficulty ordering. A natural hypothesis is that *Interleaved*, which exposes the agent to tasks from all domains in mixed order, should pose the greatest challenge due to maximal cross-domain interference. However, the empiri-

Table 2: Positive rate and average evolution gain (%) over vanilla baseline for each streaming scenario, aggregated over all model-method configurations. Top-1 rate is the percentage of configurations for which the scenario attains the highest accuracy. **Blue** marks the best scenario per metric.

Scenario	Positive (> Vanilla)	Negative (< Vanilla)	Gain	Top-1 rate
Isolated	**34/45**	**11/45**	**+1.37**$\pm$0.80	**38%**
Sequential	28/45	16/45	+0.75$\pm$0.48	29%
Interleaved	28/45	17/45	+0.90$\pm$0.34	33%

cal results contradict this expectation. Although both scenarios achieve comparable positive rates in Table 2, *Interleaved* attains a higher average evolution gain of +0.90% against +0.75% for *Sequential*, together with a higher Top-1 rate of 33% versus 29%. The advantage of *Interleaved* becomes more apparent in pairwise comparison. Across all 15 configurations in Table 1, *Interleaved* achieves higher average accuracy than *Sequential* in 10 cases while *Sequential* leads in only 5. *Interleaved* also exhibits a smaller standard deviation than *Sequential*, indicating that its advantage is not only larger but also more stable. These results suggest that the diversity of interleaved streams compensates for cross-domain noise more effectively than the ordered domain transitions in *Sequential*.

Takeaway

Self-evolution is not uniformly beneficial, and its reliability depends on the streaming scenario. *Isolated* is the most reliable streaming scenario due to distributional coherence within a single domain. Among the two cross-domain scenarios, *Interleaved* generally outperforms *Sequential* despite introducing greater task-level interference.

5.2 When Does Self-Evolution Help? The Role of Model Capability

Different models may respond differently to self-evolution. Figure 3 provides the marginal view averaged across five methods, and Table 3 offers finer-grained breakdowns across streaming scenarios.

Table 3: Positive rate and average evolution gain (%) over vanilla baseline for each model under each streaming scenario. Positive denotes the number of methods, out of 5, exceeding vanilla baseline.

	Isolated		*Sequential*		*Interleaved*	
Model	Positive	Gain	Positive	Gain	Positive	Gain
GPT-5.4	2/5	−0.35	1/5	−0.78	1/5	−0.62
Gemini 3.1 Pro	5/5	+2.71	4/5	+1.98	5/5	+2.41
Claude Opus 4.7	5/5	+1.75	4/5	+1.05	4/5	+0.90

The benefit of self-evolution is gated by model capability. As shown in Table 3, GPT-5.4 exhibits negative average evolution gains under all streaming scenarios, with only 4 out of 15 configurations exceeding its vanilla baseline. In contrast, Gemini 3.1 Pro exceeds the vanilla baseline in 14 of 15 configurations and Claude Opus 4.7 in 13 of 15. This disparity persists across *Isolated*, *Sequential*, and *Interleaved* scenarios, as confirmed by per-seed results in Tables 11 to 13. These results suggest that self-evolution relies on a bootstrap loop in which successful task completions generate high-quality experience that benefits subsequent tasks. When the base model's solve rate is low, the experience stream is dominated by failed or partially correct trajectories, from which the agent cannot reliably extract transferable knowledge. Above a sufficient capability threshold, the model solves enough early tasks to seed the experience buffer with useful patterns, potentially initiating a virtuous cycle of accumulation and transfer that persists across all streaming scenarios.

The evolution gain is non-monotonic in base model strength across streaming scenarios. Between the two models that benefit from self-evolution, Gemini 3.1 Pro obtains a larger average evolution gain of +2.37% than Claude Opus 4.7 at +1.23%, despite Claude being the stronger model on the vanilla baseline. As shown in Table 3, this pattern holds across all streaming scenarios and is corroborated in Tables 11 to 13. Moreover, the magnitude of this difference varies with streaming complexity. The gap in evolution gain between Gemini 3.1 Pro and Claude Opus 4.7 is 0.96% under *Isolated* and 0.93% under *Sequential*, but widens to 1.51% under *Interleaved*. As the streaming scenario introduces more cross-domain mixing, the mid-capability model benefits increasingly more than the stronger model. Under *Interleaved*, the mixed task stream exposes the agent to diverse cross-domain experience, providing substantial learning signals for a model that still has broad room

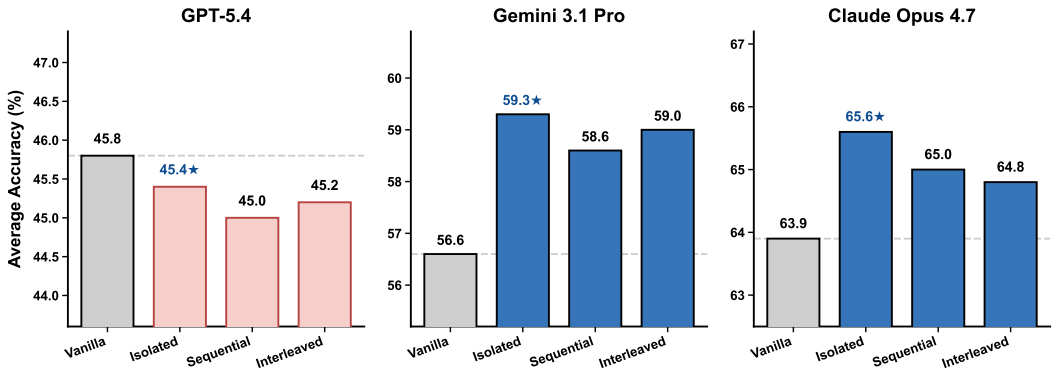

Figure 3: Average accuracy for each model under the three streaming scenarios.

to improve across multiple benchmarks. A model that already achieves high baseline accuracy on most benchmarks extracts less additional value from this diversity.

Takeaway

Self-evolution benefits are gated by model capability: a weak model fails to convert accumulated experience into gains under any streaming scenario. Moreover, the evolution gain is non-monotonic in base model strength, and this effect intensifies with streaming complexity.

5.3 How Does Model Capability Shape the Choice of Self-Evolving Method?

Having established that model capability gates when self-evolution helps, we now turn to the interaction between model and self-evolving method, asking how model capability shapes the choice of self-evolving method. Table 4 reports the accuracy of each method under each model, averaged across the three streaming scenarios.

Method sensitivity decreases with model capability. As shown in Table 4, the spread between the best and worst method contracts monotonically with model strength. On GPT-5.4, the spread is 5.3% with only 1 of 5 methods exceeding the vanilla baseline, and the worst method losing 2.3%. On Gemini 3.1 Pro, the spread narrows to 2.0% with all 5 methods above baseline. On Claude Opus 4.7, it further contracts to 0.9%, with all methods clustered in a narrow band above the vanilla baseline. The trend follows directly from the capability-gating effect identified in Section 5.2. On a weaker model, method architecture determines whether the agent can extract usable signal from a largely unsuccessful experience stream, making method choice decisive. On a stronger model, all methods reliably generate positive transfer, leaving less room for method choice to affect the outcome.

Table 4: Average accuracy (%) of each self-evolving method under each model, averaged across three streaming scenarios. Red background marks performance below the model's vanilla baseline, and **blue** marks the best method within each model.

Model	Vanilla	ACE	A-Mem	ReasoningBank	AutoSkill	Harness
GPT-5.4	45.8	44.6	**48.8**	44.7	43.5	44.5
Gemini 3.1 Pro	56.6	**59.8**	57.8	59.4	58.8	59.2
Claude Opus 4.7	63.9	65.2	**65.6**	65.3	64.9	64.7

Method choice governs whether self-evolution equalizes or amplifies inter-model gaps. Because method sensitivity concentrates on the weakest model, the choice of self-evolving method determines inter-model performance gaps. Averaging across all five methods, the gap between GPT-5.4 and Claude Opus 4.7 widens from 18.1% at vanilla to 19.9% after evolution, as evolution harms the weakest model on average while consistently benefiting the strongest. However, pairing each model with its optimal method narrows the gap to 16.8%, as A-Mem (Xu et al., 2025) lifts GPT-5.4 to 48.8% while still reaching 65.6% on Claude Opus 4.7. Critically, the optimal method is not portable across models. A-Mem is the best method on GPT-5.4 but ranks lowest on Gemini 3.1 Pro, while ACE (Zhang et al., 2026c) leads on Gemini yet ranks third on the other two models. Only ReasoningBank (Ouyang et al., 2026b) remains competitive across all three models.

Takeaway

Method sensitivity is inversely related to model capability. The optimal method varies across models rather than transferring between them, making per-model method selection critical for maximizing self-evolution gains across diverse streaming scenarios.

5.4 How Do Self-Evolving Methods Interact with Streaming Scenarios?

A single design choice splits self-evolving methods along the scenario axis: how tightly each binds experience to the execution context. Context-integrated methods such as ACE (Zhang et al., 2026c) and Harness (Lin et al., 2026c;a) fold experience directly into the agent prompt, whereas retrieval-based methods such as ReasoningBank (Ouyang et al., 2026b), AutoSkill (Yang et al., 2026b), and A-Mem (Xu et al., 2025) keep it in an external store and inject only the entries retrieved for the current task. Table 5 shows that context-integrated methods perform best under *Isolated*, whereas retrieval-based methods peak under *Interleaved*. Table 6 corroborates this at the per-model level, where ACE and Harness reach their peak accuracy under *Isolated* for most model configurations while ReasoningBank and AutoSkill consistently peak under *Interleaved*.

Table 5: Average evolution gain (%) of self-evolving methods across three streaming scenarios, aggregated over all models. The last two columns indicate the number of configurations (out of 9) in which *Isolated* or *Interleaved* attains higher performance.

Method	*Isolated*	*Sequential*	*Interleaved*	*Isolated* Higher	*Interleaved* Higher
ACE	+2.28	+2.26	−1.26	7	2
Harness	+1.91	−0.86	+1.01	5	3
A-Mem	+1.93	+1.71	+2.22	5	4
ReasoningBank	+0.78	+0.46	+1.79	2	7
AutoSkill	−0.07	+0.19	+0.72	2	7

Table 6: Average accuracy (%) of each self-evolving method across three streaming scenarios and three models. **Blue** marks the best scenario within each self-evolving method.

	GPT-5.4			Gemini 3.1 Pro			Claude Opus 4.7		
Method	*Isolated*	*Sequential*	*Interleaved*	*Isolated*	*Sequential*	*Interleaved*	*Isolated*	*Sequential*	*Interleaved*
ACE	**47.1**	44.5	42.4	59.1	**62.0**	58.3	**67.0**	66.6	61.9
Harness	**45.6**	44.2	43.7	**61.2**	56.4	59.8	65.2	63.1	**65.8**
A-Mem	47.6	48.3	**50.4**	**58.7**	57.0	57.7	65.8	**66.1**	64.8
ReasoningBank	44.2	44.4	**45.4**	58.9	59.0	**60.1**	65.5	64.3	**66.1**
AutoSkill	42.8	43.7	**43.9**	58.6	58.5	**59.1**	64.7	64.6	**65.4**

This interaction between method and streaming scenario reflects how methods couple stored knowledge with execution context. Under *Isolated*, the tight coupling of context-integrated methods converts experience into domain-specific strategies that transfer reliably across tasks within the same distribution. Under *Interleaved*, performance drops in these approaches reveal their susceptibility to cross-domain interference. Retrieval-based methods instead activate only task-relevant experience to suppress interference, while diverse tasks consolidate transferable patterns. Under *Sequential*, retrieval gating filters irrelevant experience from earlier domains, yielding moderate gains.

Takeaway

The optimal method depends on the streaming scenario. Context-integrated methods benefit most from *Isolated* where experience accumulates without interference, while retrieval-based methods favor *Interleaved* through selective retrieval and cross-domain consolidation.

6 Conclusion

We introduce AgentStream, a unified framework that organizes agentic benchmarks into a configurable task stream and evaluates self-evolving agents under three instantiated streaming scenarios across multiple models and methods. Our analysis reveals that self-evolution reliability varies across streaming scenarios, the benefit of self-evolution is gated by model capability and is non-monotonic in model strength, and no single self-evolving method consistently dominates across models and streaming scenarios. These results provide practical guidance for deploying self-evolving agents and highlight the value of evaluating them under realistic streaming settings.

AI USE STATEMENT

In this work, we used generative AI tools for writing polishing. We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, formulating mathematical claims, or interpreting results, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. The core concepts, methodology, experiments, analyses, and conclusions are the authors' original work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we provide the complete source code at `https://anonymous.4open.science/r/self-evolving-0C4A`. All experimental settings are outlined in Section 4, including the evaluated models, benchmarks, self-evolving methods, task sampling and benchmark splits, evaluation protocols, and stream construction. Appendices C and E provide the complete per-seed results and method prompts, respectively.

REFERENCES

Anthropic. Introducing Claude Opus 4.7, April 2026. URL `https://www.anthropic.com/news/claude-opus-4-7`.

Elron Bandel, Asaf Yehudai, Lilach Eden, Yehoshua Sagron, Yotam Perlitz, Elad Venezian, Natalia Razinkov, Natan Ergas, Shlomit Shachor Ifergan, Segev Shlomov, et al. General agent evaluation. In *Proc. ICML Workshops*, 2026.

Chaithanya Bandi, Razvan-Gabriel Dumitru, Ben Hertzberg, Divyansh Agarwal, Geobio Boo, Tejas Polakam, Sami Hassaan, Jeff Da, HiJae Kim, Vipul Gupta, Manasi Sharma, Andrew Park, Martin Dimakis, Ernesto Gabriel Hernandez Montoya, Dan Rambado, Ivan Salazar, Rafael Cruz, MohammadHossein Rezaei, Chetan Rane, Ben Levin, Daniel Yue Zhang, Brad Kenstler, and Bing Liu. Mcp-atlas: A large-scale benchmark for tool-use competency with real mcp servers. *arXiv preprint arXiv:2602.00933*, 2026.

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.

Yuxuan Cai, Yipeng Hao, Jie Zhou, Hang Yan, Zhikai Lei, Rui Zhen, Zhenhua Han, Yutao Yang, Junsong Li, Qianjun Pan, Tianyu Huai, Qin Chen, Xin Li, Kai Chen, Bo Zhang, Xipeng Qiu, and Liang He. Building self-evolving agents via experience-driven lifelong learning: A framework and benchmark. *arXiv preprint arXiv:2508.19005*, 2026.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. In *Proc. ACL*, 2026.

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

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng

Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Google DeepMind. Gemini 3.1 Pro model card, February 2026. URL `https://deepmind.google/models/model-cards/gemini-3-1-pro/`.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proc. ACL*, 2024.

Yufei He, Juncheng Liu, Yue Liu, Yibo Li, Tri Cao, Zhiyuan Hu, Xinxing Xu, and Bryan Hooi. Evotest: Evolutionary test-time learning for self-improving agentic systems. In *Proc. ICLR*, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *Proc. ICLR*, 2024.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. In *Proc. ICLR*, 2026a.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026b.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026a.

Minhua Lin, Hanqing Lu, Zhan Shi, Bing He, Rui Mao, Zhiwei Zhang, Zongyu Wu, Xianfeng Tang, Hui Liu, Zhenwei Dai, Xiang Zhang, Suhang Wang, Benoit Dumoulin, and Jian Pei. Position: Agentic evolution is the path to evolving llms. *arXiv preprint arXiv:2602.00359*, 2026b.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving llm agents. *arXiv preprint arXiv:2605.30621*, 2026c.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *Proc. ICLR*, 2026.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. Skillos: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026a.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *Proc. ICLR*, 2026b.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proc. ICML*, 2025.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving llm agents. In *Proc. ICLR*, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proc. ACL*, 2024.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. Cybergym: Evaluating AI agents' real-world cybersecurity capabilities at scale. In *Proc. ICLR*, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proc. ICML*, 2025.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2026.

Cheng-Kuang Wu, Zhi R Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. Streambench: Towards benchmarking continuous improvement of language agents. In *Proc. NeurIPS*, 2024.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

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 *Proc. NeurIPS*, 2024.

Guowei Xu, Zhenting Qi, Huangyuan Su, Weirui Ye, Himabindu Lakkaraju, Sham M. Kakade, and Yilun Du. Self-improving language models with bidirectional evolutionary search. *arXiv preprint arXiv:2605.28814*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Proc. NeurIPS*, 2025.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026a.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026b.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Proc. NeurIPS*, 2022.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Dengjia Zhang, Xiaoou Liu, Lu Cheng, Yaqing Wang, Kenton Murray, and Hua Wei. Selaur: Self evolving llm agent via uncertainty-aware rewards. In *Proc. PAKDD*, 2026a.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026b.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *Proc. ICLR*, 2026c.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, et al. Skillflow: Benchmarking lifelong skill discovery and evolution for autonomous agents. *arXiv preprint arXiv:2604.17308*, 2026d.

A Cost Analysis

A practical concern for deploying self-evolving agents is whether performance gains justify the additional cost. Tables 7 and 8 report the per-task cost, relative cost overhead, and average agent steps for GPT-5.4 and Gemini 3.1 Pro respectively, where all costs are reported using LiteLLM's pricing data[1]. Figure 4 summarizes these results at the method level, averaged across three streaming scenarios.

Self-evolution does not necessarily increase inference cost. On Gemini 3.1 Pro, four methods operate below the vanilla baseline cost, reducing it to 66% for ReasoningBank (Ouyang et al., 2026b), 89% for Harness (Lin et al., 2026c;a), 79% for AutoSkill (Yang et al., 2026b), and 89% for A-Mem (Xu et al., 2025). This cost reduction is largely accompanied by a decrease in average agent steps, suggesting that accumulated experience helps the agent reach solutions more efficiently. On GPT-5.4, the pattern reverses. Most methods incur higher cost, with A-Mem reaching 597% of the baseline and ACE (Zhang et al., 2026c) at 285%, while ReasoningBank remains the only cost-reducing method at 89% of the baseline.

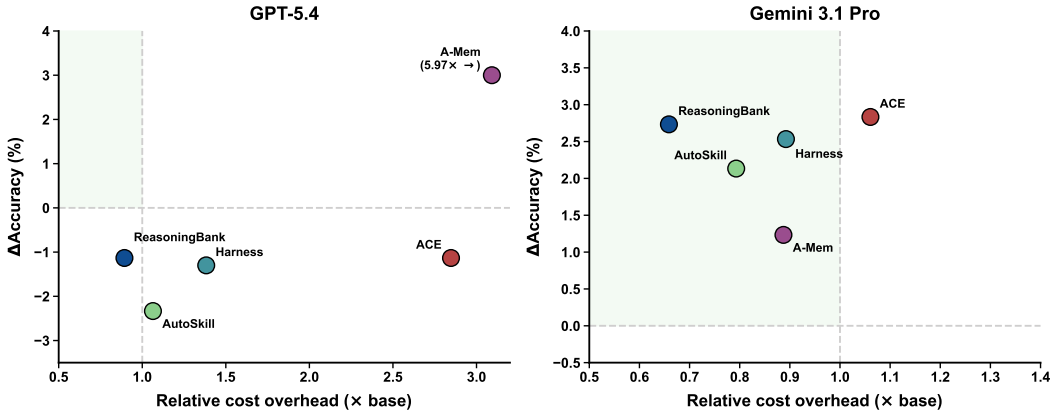

Figure 4: Cost overhead and evolution gain of each self-evolving method averaged across three streaming scenarios.

[1]Model prices.

The cost-performance tradeoff is governed by the model. On Gemini 3.1 Pro, ACE and AutoSkill both achieve evolution gains above +2% with moderate or reduced cost overhead. ReasoningBank provides +2.7% evolution gain while reducing cost to 66% of the baseline, making it the most cost-efficient method on this model. On GPT-5.4, only A-Mem achieves a positive evolution gain of +3.0%, but at 597% of the baseline cost. All other methods on GPT-5.4 incur negative evolution gains regardless of their cost overhead. This asymmetry echoes the finding in Section 5.2 that weaker models struggle to convert accumulated experience into performance improvements, and further shows that this limitation extends to cost efficiency.

Table 7: Per-task cost ($), relative cost overhead, evolution gain (%), and average agent steps for GPT-5.4. The vanilla baseline operates at $0.297 per task with 9.5 steps and 45.8% accuracy.

Method	Mode	Cost	Relative	ΔAcc.	Steps	ΔSteps (%)
ACE	*Isolated*	0.510	1.72×	+1.3	11.3	+18.9
	Sequential	1.112	3.74×	−1.3	11.6	+22.1
	Interleaved	0.915	3.08×	−3.4	10.8	+13.7
Harness	*Isolated*	0.403	1.36×	−0.2	10.3	+8.4
	Sequential	0.426	1.43×	−1.6	10.2	+7.4
	Interleaved	0.403	1.36×	−2.1	10.0	+5.3
A-Mem	*Isolated*	1.482	4.99×	+1.9	8.8	−7.4
	Sequential	1.755	5.91×	+2.5	8.7	−8.4
	Interleaved	2.084	7.02×	+4.6	8.9	−6.3
ReasoningBank	*Isolated*	0.277	0.93×	−1.6	8.4	−11.6
	Sequential	0.258	0.87×	−1.4	7.7	−18.9
	Interleaved	0.261	0.88×	−0.4	8.1	−14.7
AutoSkill	*Isolated*	0.310	1.04×	−3.0	9.4	−1.1
	Sequential	0.318	1.07×	−2.1	9.7	+2.1
	Interleaved	0.320	1.08×	−1.9	9.6	+1.1

Table 8: Per-task cost ($), relative cost overhead, evolution gain (%), and average agent steps for Gemini 3.1 Pro. The vanilla baseline operates at $4.035 per task with 21.1 steps and 56.6% accuracy.

Method	Mode	Cost	Relative	ΔAcc.	Steps	ΔSteps (%)
ACE	*Isolated*	3.788	0.94×	+2.5	21.4	+1.4
	Sequential	4.167	1.03×	+5.4	22.1	+4.7
	Interleaved	4.885	1.21×	+0.6	22.5	+6.6
Harness	*Isolated*	4.139	1.03×	+4.6	22.4	+6.2
	Sequential	3.253	0.81×	−0.2	20.7	−1.9
	Interleaved	3.408	0.84×	+3.2	22.1	+4.7
A-Mem	*Isolated*	3.656	0.91×	+2.1	17.9	−15.2
	Sequential	3.753	0.93×	+0.4	18.1	−14.2
	Interleaved	3.331	0.83×	+1.2	17.7	−16.1
ReasoningBank	*Isolated*	2.725	0.68×	+2.3	19.5	−7.6
	Sequential	2.622	0.65×	+2.4	19.6	−7.1
	Interleaved	2.628	0.65×	+3.5	19.0	−10.0
AutoSkill	*Isolated*	2.919	0.72×	+2.0	20.7	−1.9
	Sequential	3.093	0.77×	+1.9	20.8	−1.4
	Interleaved	3.587	0.89×	+2.5	21.8	+3.3

Table 9: Evolution state accumulated by each model over the *Interleaved* stream.

Method	Metric	GPT-5.4	Gemini 3.1 Pro	Claude Opus 4.7
ACE	playbook bullets	552	209	1251
ReasoningBank	memory items	894	585	898
A-Mem	evolutions	261	248	261
AutoSkill	skills	176	117	160
Harness	skills	181	137	463
Harness	prompt+memory (chars)	8,711	29,172	9,362

B Model Evolution Behavior Analysis

Beyond aggregate performance, we analyze how the three models differ in their self-evolution behavior under the *Interleaved* stream, focusing on the evolution state each model accumulates and how that state is updated as tasks are processed.

B.1 Evolution State Accumulation

As shown in Table 9, the three models accumulate evolution states of markedly different sizes and structures. Claude Opus 4.7 generates the largest states across most methods, averaging 1251 ACE (Zhang et al., 2026c) playbook bullets and 463 Harness (Lin et al., 2026c;a) skills, while Gemini 3.1 Pro consistently produces the most compact states with only 209 ACE playbook bullets and 117 AutoSkill (Yang et al., 2026b) skills. GPT-5.4 generates a moderate number of items with relatively concise per-item content, balancing breadth of coverage with manageable state complexity. The contrast extends beyond volume to architectural preference: under Harness, Claude Opus 4.7 allocates the vast majority of its evolution budget to the skill library while maintaining only 9K characters of system prompt and memory, whereas Gemini 3.1 Pro distributes 29K characters across its system prompt and memory fields, favoring centralized guidance over distributed skill entries. A similar pattern holds for memory-based methods. Under ReasoningBank (Ouyang et al., 2026b), Claude Opus 4.7 and GPT-5.4 each distill approximately 900 memory items while Gemini produces only 585, confirming that Gemini extracts fewer but more consolidated experience entries from each interaction.

B.2 Update Dynamics

Table 10 summarizes how the models update the state maintained by Harness, which provides the most informative view since it jointly maintains system prompt, memory, and skill library. The three models adopt distinct update strategies. GPT-5.4 is largely append-only, adding skills but rarely editing them, with only 5 edits on average over 300 tasks and minimal modification to the system prompt. Gemini 3.1 Pro spreads its updates more evenly, revising the memory and system prompt 267 times while also refining skills through 66 edits. Claude Opus 4.7 performs the most intensive refinement, revising on 96% of tasks while concentrating almost entirely on the skill library rather than the memory or system prompt.

Table 10: Evolution behavior on Harness state over the *Interleaved* stream.

	GPT-5.4	Gemini 3.1 Pro	Claude Opus 4.7
Revision rate (%)	65	69	96
Skills added	181	147	492
Skills edited	5	66	460
Memory edited	20	180	82
Prompt edited	0	87	21

C Per-Seed Main Results

The main results in Table 1 are averaged over three random seeds that share the same task set but differ in the global task arrival order. Tables 11 to 13 provide the complete per-seed results.

D Cumulative Accuracy Dynamics

To characterize the dynamics of self-evolution, we report the cumulative accuracy over three streaming scenarios for each self-evolving method and model in Figures 5 to 19.

E Method Prompts

For reproducibility, we provide the full set of prompts used by self-evolving methods. All prompts are adapted to our test-time setting, where no ground-truth labels are available, and to the Exgentic (Bandel et al., 2026) framework, in which the agent completes tasks through multi-step tool use rather than single-turn question answering.

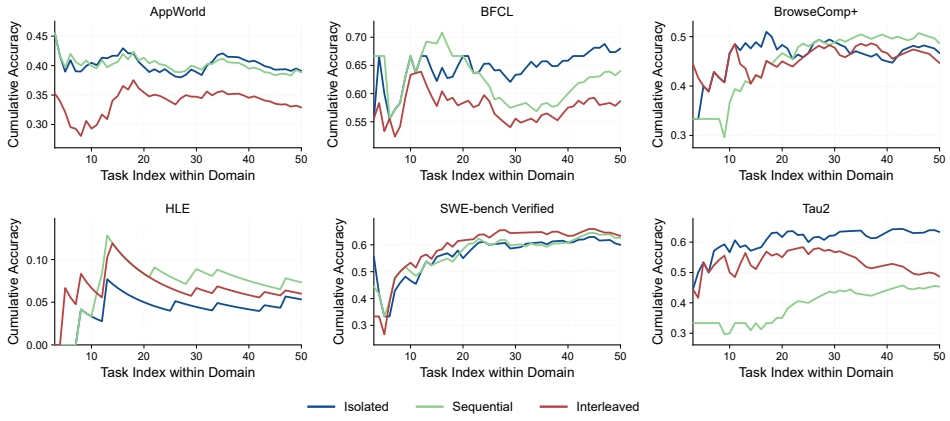

Figure 5: Cumulative accuracy dynamics of ACE on GPT-5.4 over three streaming scenarios.

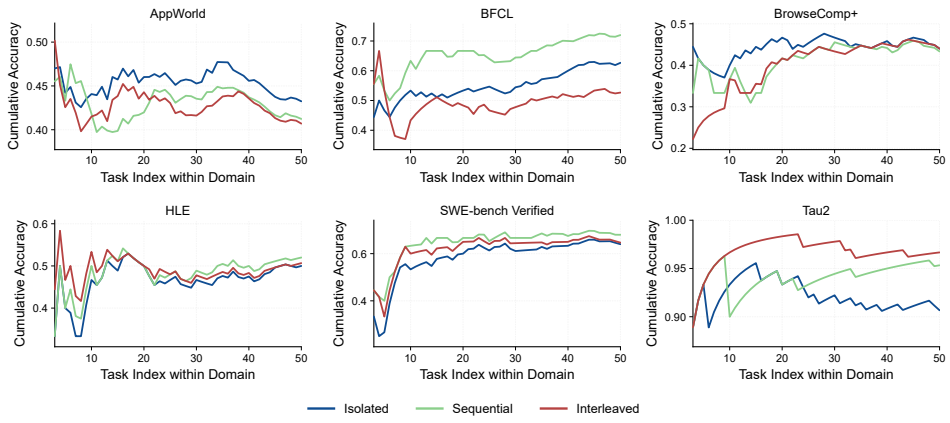

Figure 6: Cumulative accuracy dynamics of ACE on Gemini 3.1 Pro over three streaming scenarios.

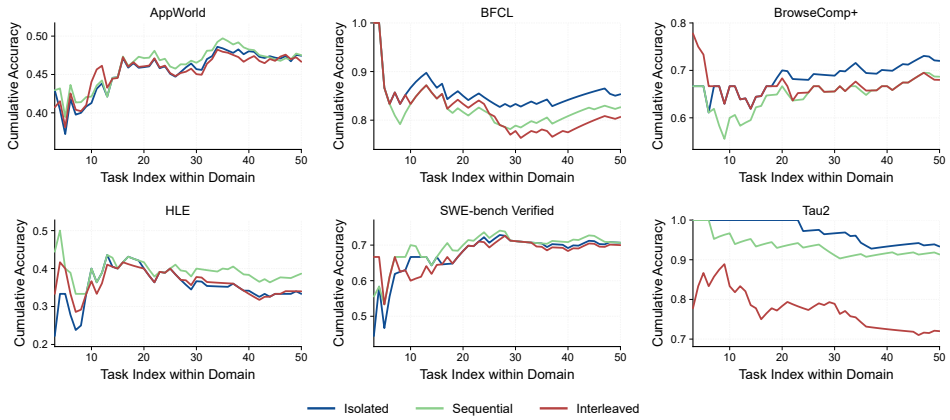

Figure 7: Cumulative accuracy dynamics of ACE on Claude Opus 4.7 over three streaming scenarios.

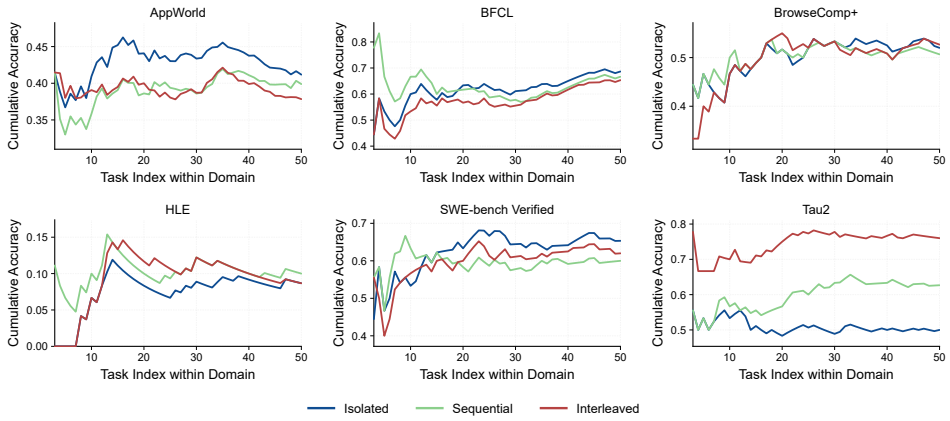

Figure 8: Cumulative accuracy dynamics of A-Mem on GPT-5.4 over three streaming scenarios.

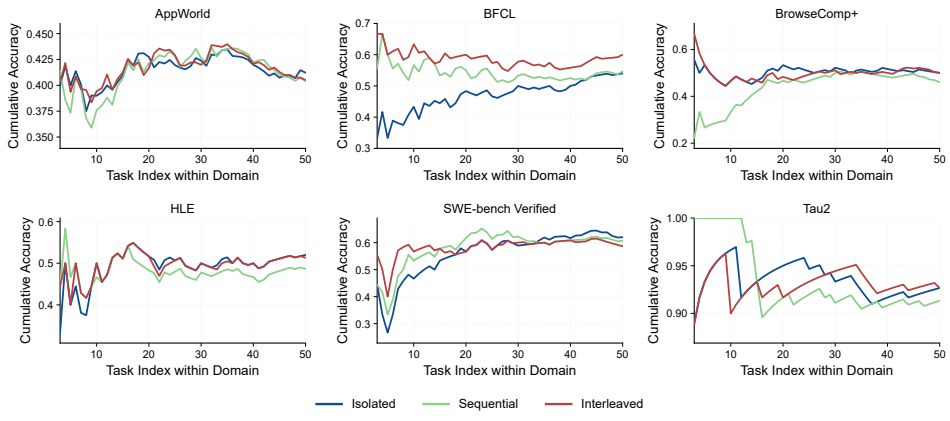

Figure 9: Cumulative accuracy dynamics of A-Mem on Gemini 3.1 Pro over three streaming scenarios.

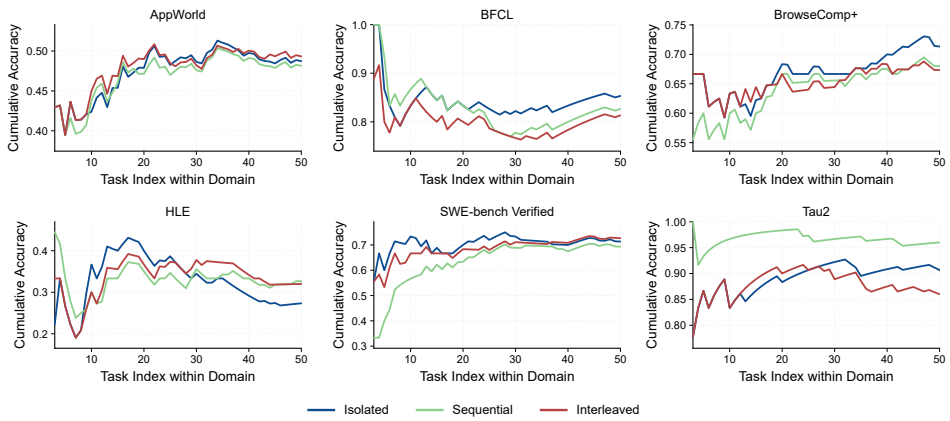

Figure 10: Cumulative accuracy dynamics of A-Mem on Claude Opus 4.7 over three streaming scenarios.

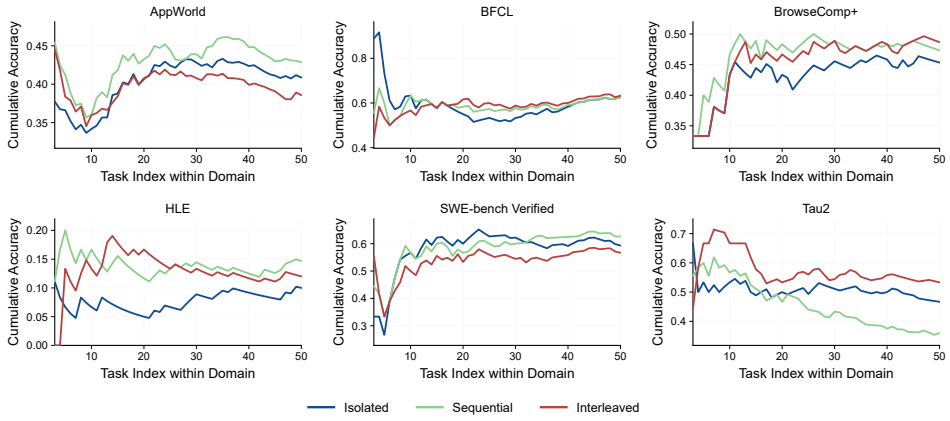

Figure 11: Cumulative accuracy dynamics of ReasoningBank on GPT-5.4 over three streaming scenarios.

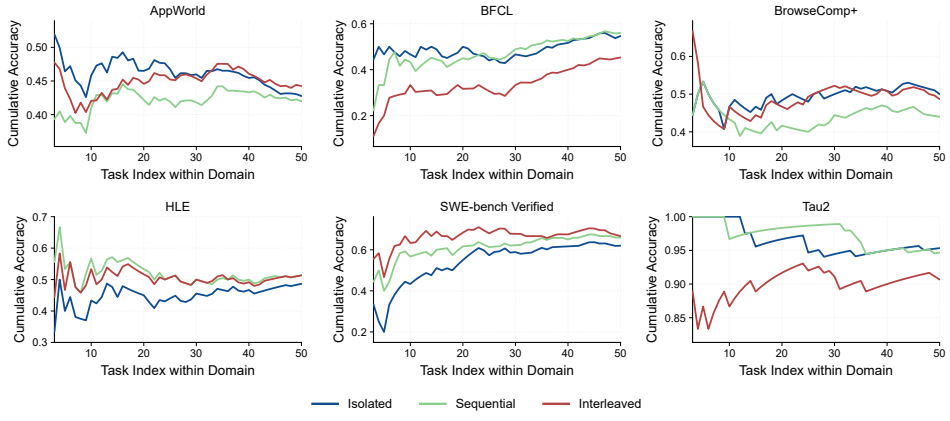

Figure 12: Cumulative accuracy dynamics of ReasoningBank on Gemini 3.1 Pro over three streaming scenarios.

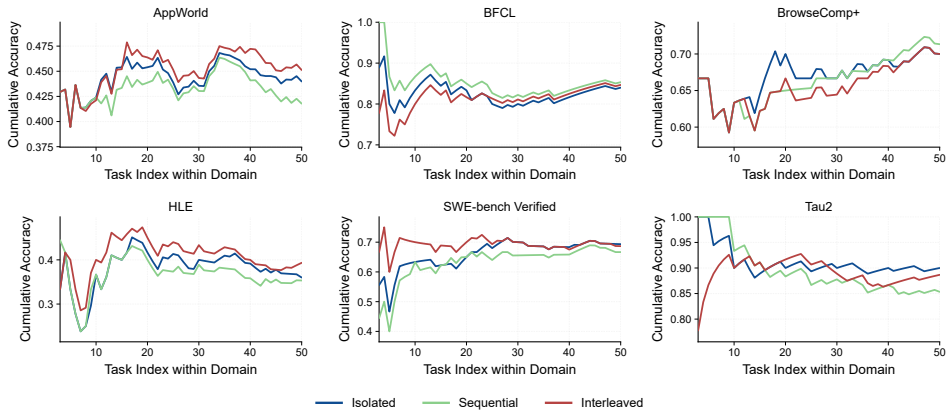

Figure 13: Cumulative accuracy dynamics of ReasoningBank on Claude Opus 4.7 over three streaming scenarios.

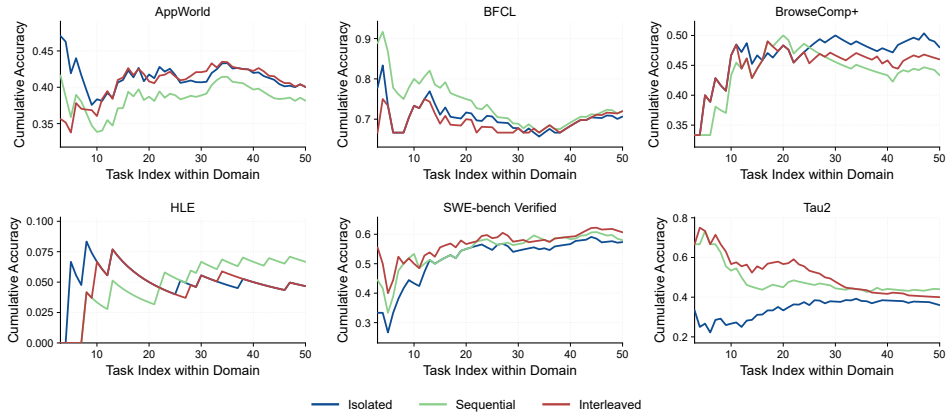

Figure 14: Cumulative accuracy dynamics of AutoSkill on GPT-5.4 over three streaming scenarios.

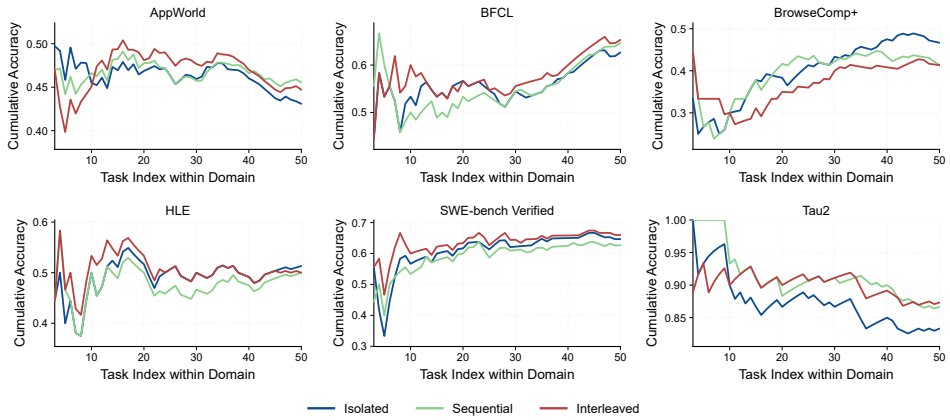

Figure 15: Cumulative accuracy dynamics of AutoSkill on Gemini 3.1 Pro over three streaming scenarios.

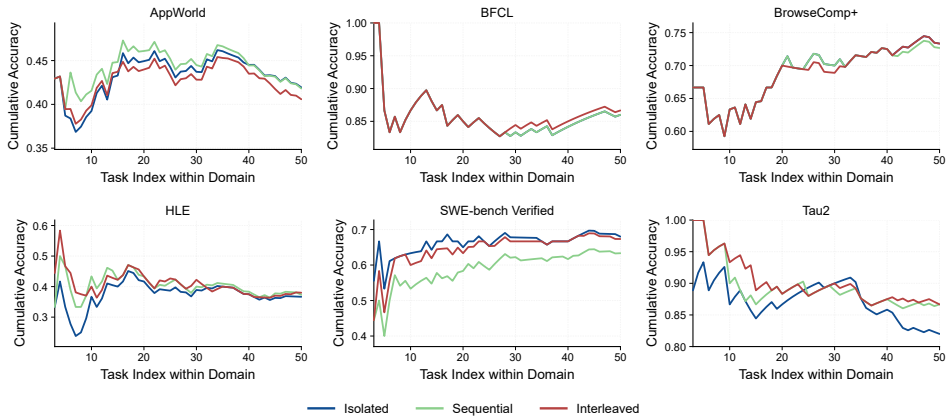

Figure 16: Cumulative accuracy dynamics of AutoSkill on Claude Opus 4.7 over three streaming scenarios.

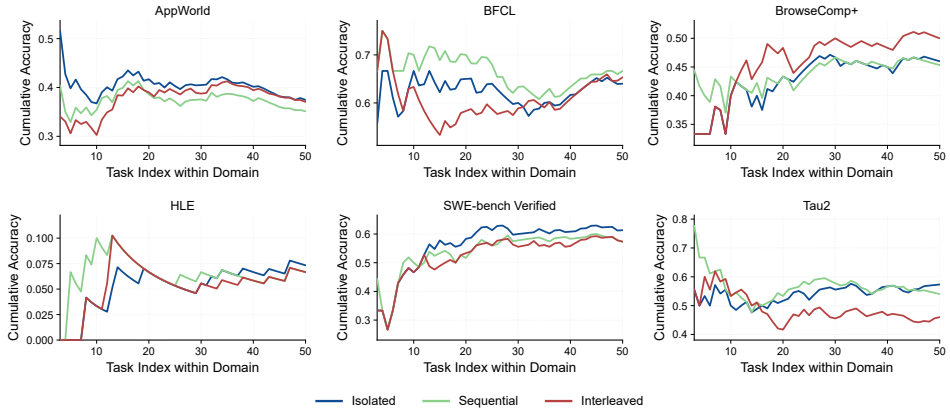

Figure 17: Cumulative accuracy dynamics of Harness on GPT-5.4 over three streaming scenarios.

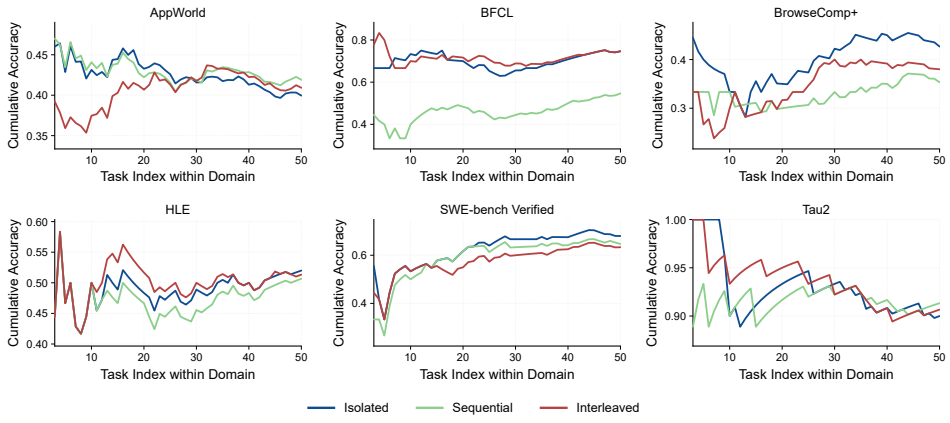

Figure 18: Cumulative accuracy dynamics of Harness on Gemini 3.1 Pro over three streaming scenarios.

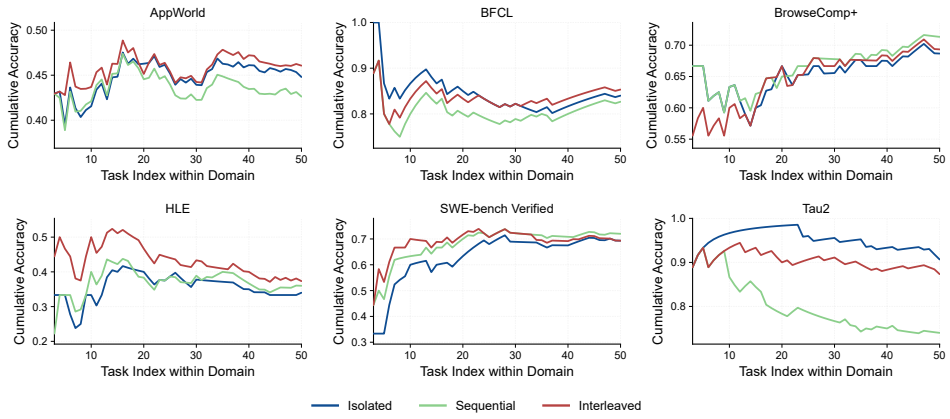

Figure 19: Cumulative accuracy dynamics of Harness on Claude Opus 4.7 over three streaming scenarios.

E.1 ACE

ACE Generator (used as the system prompt)

```
You are an expert agent that completes tasks using available
tools.
You have access to a curated playbook of strategies and
insights learned from previous tasks.  Use these to make better
decisions.

## Guidelines
- Read the playbook carefully and apply relevant strategies
- Pay attention to common mistakes listed and avoid them
- Use available tools to interact with the environment
- Think step-by-step before acting
- When you are confident in your solution, use the
finish/submit tool
- When a playbook bullet influences your decision, mention its
ID (e.g. [err-00001]) in your reasoning text

## Playbook (accumulated strategies & insights)
Each line has a bullet ID and usage stats (helpful=N means it
helped N times, harmful=N means it misled N times).
Prefer high-helpful, low-harmful bullets.

{playbook}
```

ACE Reflector

```
You are an expert analyst and educator. Your job is to analyze
a model's reasoning process and identify potential issues or
strengths based on the reasoning trace alone.

**Instructions:**
- Carefully analyze the model's reasoning trace to evaluate
its approach
- The reasoning trace includes both the model's actions and
environment observations in chronological order
- Identify potential conceptual errors, calculation mistakes,
```

```
or misapplied strategies
- Also note what the model did well
- Provide actionable insights that could help the model perform
better in future tasks
- Focus on the root cause, not just surface-level observations
- Be specific about what could be improved
- You will receive the full playbook that was available
to the agent
- Based on the reasoning trace, infer which bullets the agent
likely applied
or was influenced by, and tag each relevant bullet as
'helpful', 'harmful', or 'neutral'. Skip unrelated bullets

Your output should be a json object, which contains the
following fields
- reasoning: your chain of thought / reasoning
/ thinking process
- error_identification: what potential issues exist
in the reasoning? (or "none identified" if the approach
appears sound)
- root_cause_analysis: why might these issues occur? What
concept may have been misunderstood?
- correct_approach: what could the model do differently
or better?
- key_insight: what strategy, formula, or principle should be
remembered for future tasks?
- bullet_tags: a list of json objects with bullet id and tag
for each relevant playbook bullet

**Question:**
{question}

**Model's Reasoning Trace:**
{reasoning_trace}

**Model's Predicted Answer:**
{predicted_answer}

**Full Playbook:**
{bullets_used}

**Answer in this exact JSON format:**
{
  "reasoning": "[Your chain of thought / reasoning
  / thinking process]",
  "error_identification": "[What potential issues exist in the
  reasoning?]",
  "root_cause_analysis": "[Why might these issues occur?]",
  "correct_approach": "[What could the model do differently
  or better?]",
  "key_insight": "[What strategy or principle should
  be remembered?]",
  "bullet_tags": [
    {"id": "calc-00001", "tag": "helpful"},
    {"id": "fin-00002", "tag": "harmful"}
  ]
}
```

ACE Curator

```
You are a master curator of knowledge. Your job is to identify
what new insights should be added to an existing playbook based
on a reflection from a previous attempt.

**Context:**
- The playbook you created will be used to help answering
similar questions.
- The reflection is generated using environment feedback that
will NOT be available when the playbook is being used.

**CRITICAL: You MUST respond with valid JSON only. Do not use
markdown formatting or code blocks.**

**Instructions:**
- Review the existing playbook and the reflection from the
previous attempt
- Identify ONLY the NEW insights, strategies, or mistakes that
are MISSING from the current playbook
- Avoid redundancy - if similar advice already exists, only add
new content that is a perfect complement to the existing
playbook
- Do NOT regenerate the entire playbook - only provide the
additions needed
- Focus on quality over quantity - a focused, well-organized
playbook is better than an exhaustive one
- Format your response as a PURE JSON object with specific
sections
- For any operation if no new content to add, return an empty
list for the operations field
- Be concise and specific - each addition should be actionable

**Training Context:**
- Total token budget: {token_budget} tokens
- Training progress: Sample {current_step} out of
{total_samples}

**Current Playbook Stats:**
{playbook_stats}

**Recent Reflection:**
{recent_reflection}

**Current Playbook:**
{current_playbook}

**Question Context:**
{question_context}

**Your Task:**
Output ONLY a valid JSON object with these exact fields:
- reasoning: your chain of thought / reasoning
/ thinking process
- operations: a list of operations to be performed on the
playbook
- type: the type of operation to be performed
- section: the section to add the bullet to
```

```
- content: the new content of the bullet

**Available Operations:**
1. ADD: Create new bullet points with fresh IDs
    - section: the section to add the new bullet to
    - content: the new content of the bullet. Note: no need to
    include the bullet_id in the content like '[ctx-00263]
    helpful=1 harmful=0 ::', the bullet_id will be added by
    the system.

**RESPONSE FORMAT - Output ONLY this JSON structure
(no markdown, no code blocks):**
{
  "reasoning": "[Your reasoning here]",
  "operations": [
    {
      "type": "ADD",
      "section": "formulas_and_calculations",
      "content": "[New calculation method...]"
    }
  ]
}
```

E.2 A-MEM

A-Mem Generator (used as the system prompt)

```
You are an expert agent that completes tasks using available
tools.
Think step-by-step before acting.
Use available tools to interact with the environment.
When you are confident in your solution, use the finish/submit
tool.

Based on the context below, complete the task. Use the context
to inform your decisions.

Context:
{memory_context}
```

A-Mem Retrieval Query

```
Given the following question, generate several keywords
separated by commas.

Question: {question}

Keywords:
```

A-Mem Content Analysis

```
Analyze the following content and provide:
1. KEYWORDS: The most important keywords (nouns, verbs,
key concepts). Order from most to least important. At least
three keywords. Do not include speaker names or time references.
2. CONTEXT: One sentence summarizing the main topic, key points,
and purpose.
```

```
3. TAGS: Broad categories/themes for classification (domain,
format, type). At least three tags.

Respond using EXACTLY this format (one section per header):

KEYWORDS: keyword1, keyword2, keyword3, ...
CONTEXT: A single sentence summarizing the content.
TAGS: tag1, tag2, tag3, ...

Content for analysis:
{content}
```

A-Mem Focused Keywords

```
List exactly 5 keywords that capture the main concepts of the
following text.
Output only the keywords, comma-separated, nothing else.

Text: {content}
```

A-Mem Evolution Decision

```
You are an AI memory evolution agent. Analyze the new
memory note and its nearest neighbors to decide if evolution
is needed.

New memory:
- Context: {context}
- Content: {content}
- Keywords: {keywords}

Nearest neighbor memories:
{nearest_neighbors_memories}

Based on the relationships between the new memory and its
neighbors, decide:
- NO_EVOLUTION: The memory stands alone, no changes needed.
- STRENGTHEN: The new memory should be linked to some neighbors
and its tags updated.
- UPDATE_NEIGHBOR: The neighbors' context/tags should be
updated based on new understanding.
- STRENGTHEN_AND_UPDATE: Both strengthen and update neighbors.

Respond using EXACTLY this format:
DECISION: <one of NO_EVOLUTION, STRENGTHEN, UPDATE_NEIGHBOR,
STRENGTHEN_AND_UPDATE>
REASON: <brief explanation>
```

A-Mem Strengthen Details

```
Given the new memory and its neighbors, provide
updated connections and tags.

New memory:
- Content: {content}
- Keywords: {keywords}
```

```
Neighbor memories:
{nearest_neighbors_memories}

Which neighbor indices should the new memory connect to? What
tags best describe this memory?

Respond using EXACTLY this format:
CONNECTIONS: 0, 2, 3
TAGS: tag1, tag2, tag3, ...
```

A-Mem Update Neighbors

```
Given the new memory and its neighbor memories, update each
neighbor's context and tags based on a holistic understanding
of all these memories together.

New memory:
- Content: {content}
- Context: {context}

Neighbor memories:
{nearest_neighbors_memories}

For each neighbor (indexed 0 to {max_neighbor_idx}), provide
updated context and tags. If no change is needed, repeat the
original values.

Respond using EXACTLY this format (one block per neighbor):

NEIGHBOR 0:
CONTEXT: updated context sentence
TAGS: tag1, tag2, tag3

NEIGHBOR 1:
CONTEXT: updated context sentence
TAGS: tag1, tag2, tag3

(continue for all {neighbor_count} neighbors)
```

E.3 REASONINGBANK

ReasoningBank Generator (used as the system prompt)

```
You are an expert agent that completes tasks using available
tools.

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.

{retrieved_memory_items}
```

ReasoningBank Trajectory Evaluator (system message)

```
You are an expert in evaluating the performance of a
task-solving agent. The agent is designed to help a human user
complete a task by taking actions in an environment. Given the
user's intent, the agent's action history, the environment's
feedback, and the agent's response to the user, your goal is to
decide whether the agent's execution is successful or not.

*Strictness rules*
Before calling a task successful, verify all three:
- Completeness: every constraint in the intent is satisfied.
- Grounding: every value or result the agent reports is
traceable to a specific observation from the environment;
values that were inferred, guessed, or summarized without a
visible source count as failures.
- Right target: when the task names a specific entity, confirm
the agent acted on that exact entity and not an adjacent one.
When uncertain on any of these, mark failure. A false success
is more harmful
than a false failure, because memory induction amplifies it
into future behavior.

*IMPORTANT*
Format your response into two lines as shown below:

Thoughts: <your thoughts and reasoning process>"
Status: "success" or "failure"
```

ReasoningBank Trajectory Evaluator (user message)

```
User Intent: {intent}

Action History:
{last_actions}

Environment feedback (last observations):

'''
{cap}
'''

Agent response to the user: {response}.
```

ReasoningBank Memory Induction, success (system message)

```
You are an expert at analyzing agent task execution. 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.
  - Prefer concrete, actionable procedures over abstract
  principles. Do not embed specific product names, queries, or
  literal string contents from the task.

## 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 describing when or when
NOT to use the memory item>
## Content <1-3 sentences describing the insights learned to
successfully accomplishing similar tasks in the future>
```

ReasoningBank Memory Induction, failure (system message)

```
You are an expert at analyzing agent task execution. 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.
  - Prefer concrete, actionable recovery procedures over
  abstract principles. Do not embed specific product names,
  queries, or literal string contents from the task.

## 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 describing when or when
NOT to use the memory item>
## Content <1-3 sentences describing the insights learned to
avoid such failures and successfully accomplishing similar
tasks in the future>
```

ReasoningBank Memory Induction (user message)

```
**Query:** {query}

**Trajectory:**
{trajectory}
```

E.4 AUTOSKILL

AutoSkill Generator (used as the system prompt)

```
You are an expert agent that completes tasks using available
tools.
Think step-by-step before acting.
Use available tools to interact with the environment.
When you are confident in your solution, use the finish/submit
tool.

## Retrieved Skills (from accumulated experience)
The following skills were retrieved based on relevance to the
current task.
Use a skill ONLY when it directly matches the current intent.
Otherwise, ignore all retrieved skills and act normally.
Never explicitly mention that skills were retrieved/injected.

{skills_block}
```

AutoSkill Retrieved Skill Entry

```
### Skill: {name}
- **Description**: {description}
- **Tags**: {tags}
- **Triggers**: {triggers}

**Instructions**:
{instructions}
```

AutoSkill Query Rewrite

```
You are a retrieval query rewriter. Your job is to rewrite the
current user task into a concise, standalone search query for
skill retrieval.

Core rules:
- Produce exactly ONE line of output: the rewritten query.
- Resolve references ("it", "this", "the above") using the
provided context.
- Keep only retrieval-relevant constraints (format, audience
, quality, domain).
- Preserve the task anchor (what the task is about).
- Do NOT include generic process words without a concrete
topic anchor.

Task: {task}
Context: {context}

Rewritten query:
```

AutoSkill Skill Extraction

```
You are a skill extractor that turns agent interaction traces
into reusable skills.

## Extraction Principles
- Treat the task description and environment observations as
primary evidence.
- Extract ONLY when there are durable, reusable constraints,
policies, workflows, or strategies that would help in FUTURE
similar tasks.
- Do NOT extract one-shot task-specific facts or generic
"be helpful" patterns.
- Capture HOW TO DO similar tasks, rather than this-instance
facts.
- Remove case-specific entities (names, URLs, dates) and
preserve only portable rules.
- Do NOT invent workflow steps unless explicitly demonstrated
in the trace.
- If nothing reusable is found, return an empty skills list.

## Session Information
Task: {task}

## Session Trace (Actions & Observations)
{session_trace}

## Output Format
Return a JSON object with this schema:
{
  "skills": [
    {
      "name": "<concise, searchable name>",
      "description": "<what this skill does and when to use
      it>",
      "instructions": "<markdown body with # Goal,
      # Constraints & Style, # Workflow (optional)>",
      "triggers": ["<intent phrase 1>", "<intent phrase 2>",
       ...],
      "tags": ["<keyword1>", "<keyword2>", ...],
      "confidence": <float 0.0-1.0>
    }
  ]
}

If nothing reusable is detected, return: {"skills": []}
```

AutoSkill Skill Judge

```
You are a skill set manager. Given a newly extracted skill
candidate and the most similar existing skill from the skill
bank, decide the appropriate action.

## Decision Procedure
1. Check if the candidate represents the same capability as the
   existing skill (same job-to-be-done, same deliverable type,
   overlapping constraints).
2. Apply discard gate: reject generic, low-signal, non-portable
```

```
   candidates.
3. Compare on four axes: job-to-be-done, deliverable type, hard
   constraints/success criteria, and required tools/workflow.
4. Choose "merge" ONLY when they are the same capability after
   removing instance details.
5. Choose "add" when the candidate is a distinct durable
   capability.
6. Choose "discard" when the candidate is too generic or
   non-reusable.

## Candidate Skill
Name: {candidate_name}
Description: {candidate_description}
Instructions: {candidate_instructions}
Triggers: {candidate_triggers}
Tags: {candidate_tags}

## Most Similar Existing Skill (may be empty if no skills exist)
Name: {existing_name}
Description: {existing_description}
Instructions: {existing_instructions}
Triggers: {existing_triggers}
Tags: {existing_tags}
Similarity Score: {similarity_score}

## Output Format
Return a JSON object:
{
  "action": "add" | "merge" | "discard",
  "target_skill_id": "<id of existing skill to merge with,
   or null>",
  "reason": "<brief explanation>"
}
```

AutoSkill Skill Merge

```
You are a skill merger. Combine an existing skill with a new
candidate into one improved skill that preserves the best
of both.

## Merge Rules
- Preserve the original capability identity (name and
  core goal).
- Perform semantic union rather than raw concatenation.
- Import only reusable, non-conflicting additions from the
  candidate.
- Avoid regressions: keep important checks from the existing
  skill.
- Remove case-specific entities and one-off facts.
- Do NOT invent any new standards or details not present in
  either skill.
- Deduplicate sections, bullets, triggers, tags.
- Keep language consistent across all fields.

## Existing Skill
Name: {existing_name}
Description: {existing_description}
```

```
Instructions: {existing_instructions}
Triggers: {existing_triggers}
Tags: {existing_tags}

## Candidate Skill (new evidence)
Name: {candidate_name}
Description: {candidate_description}
Instructions: {candidate_instructions}
Triggers: {candidate_triggers}
Tags: {candidate_tags}

## Output Format
Return a JSON object with the merged skill:
{
  "name": "<merged name>",
  "description": "<merged description>",
  "instructions": "<merged instructions (markdown)>",
  "triggers": ["<trigger1>", ...],
  "tags": ["<tag1>", ...]
}
```

E.5 HARNESS

Harness Generator (used as the system prompt)

```
You are an expert agent that completes tasks using available
tools.
Think step-by-step before acting.
Use available tools to interact with the environment.
When you are confident in your solution, use the finish/submit
tool.

## Long-Term Memory
{memory}

## Retrieved Skills
### Skill: {skill_name}
*{skill_description}*

{skill_body}
```

Harness Evolver (system prompt)

```
You are an evolution engine for an agent harness. Your job is
to analyze a completed task session and improve the agent's
harness (system prompt, long-term memory, and skill library)
for future tasks.

## Available Tools

**Read tools** (use these first to inspect current state):
- read_prompt() - read the current system prompt
- read_memory() - read the current long-term memory document
- list_skills() - list all skills with names and descriptions
- read_skill(name) - read a specific skill's full body

**Write tools** (use these to make changes):
```

```
- edit_prompt(body) - replace the entire system prompt
- edit_memory(body) - replace the entire memory document
- add_skill(name, description, body) - add a new skill
- edit_skill(name, description, body) - modify an existing
  skill
- delete_skill(name) - remove a skill

## Constraints
- At most 1 edit_prompt call per session.
- At most 1 edit_memory call per session.
- No limit on skill operations.

## Guidelines
- First READ the current harness state, then decide what
  changes to make.
- Skills should be generalizable (useful across tasks), not
  task-specific.
- Memory should capture recurring patterns, proven strategies,
and environment quirks.
- System prompt changes should refine the agent's general
  approach.
- Do NOT duplicate information already present in the harness.
- If no changes are needed, simply stop without calling any
  write tools.
```

Harness Evolver (user message)

```
## This Session

### Task
{task}

### Skills Injected
{injected_skill_names}

### Session Trajectory
{trajectory}

---

Analyze the session above. Read the current harness state using
the read tools, then decide what changes (if any) would improve
the agent's future performance. Make changes using the write
tools, or stop if no changes are needed.
```

Table 11: Main results (%) of self-evolving methods across three streaming scenarios under seed 42. Red background indicates scores below the model's vanilla baseline, and **blue** marks the best scenario within each self-evolving method.

Methods	Mode	AppWorld	BFCL	BrowseComp+	HLE	SWE	Tau2	Avg
GPT-5.4	Vanilla	44.6	66.0	50.0	2.0	62.0	50.0	45.8
ACE	*Isolated*	33.1	54.0	44.0	6.0	58.0	70.0	**44.2**
	Sequential	33.1	58.0	54.0	8.0	58.0	20.0	38.5
	Interleaved	32.3	62.0	46.0	4.0	64.0	52.0	43.4
A-Mem	*Isolated*	41.8	70.0	60.0	12.0	66.0	12.0	43.6
	Sequential	41.8	64.0	48.0	10.0	64.0	84.0	**52.0**
	Interleaved	32.4	68.0	56.0	10.0	60.0	82.0	51.4
ReasoningBank	*Isolated*	42.9	70.0	48.0	8.0	54.0	60.0	**47.2**
	Sequential	42.9	60.0	48.0	12.0	60.0	52.0	45.8
	Interleaved	36.6	58.0	44.0	12.0	52.0	50.0	42.1
AutoSkill	*Isolated*	36.8	72.0	52.0	6.0	64.0	30.0	43.5
	Sequential	36.3	76.0	48.0	6.0	54.0	56.0	**46.1**
	Interleaved	42.0	66.0	46.0	4.0	60.0	46.0	44.0
Harness	*Isolated*	37.9	70.0	48.0	14.0	60.0	60.0	**48.3**
	Sequential	36.4	68.0	46.0	8.0	52.0	48.0	43.1
	Interleaved	40.6	66.0	46.0	8.0	60.0	44.0	44.1
Gemini 3.1 Pro	Vanilla	41.8	58.0	34.0	52.0	64.0	90.0	56.6
ACE	*Isolated*	41.2	74.0	46.0	50.0	64.0	94.0	61.5
	Sequential	40.9	74.0	48.0	52.0	68.0	96.0	**63.2**
	Interleaved	41.6	40.0	38.0	52.0	64.0	94.0	54.9
A-Mem	*Isolated*	33.6	62.0	54.0	50.0	66.0	90.0	59.3
	Sequential	36.9	52.0	46.0	44.0	60.0	90.0	54.8
	Interleaved	37.9	62.0	56.0	50.0	62.0	92.0	**60.0**
ReasoningBank	*Isolated*	44.4	60.0	44.0	50.0	62.0	94.0	59.1
	Sequential	41.3	58.0	42.0	52.0	64.0	92.0	58.2
	Interleaved	46.7	46.0	54.0	50.0	68.0	92.0	**59.5**
AutoSkill	*Isolated*	44.7	66.0	48.0	52.0	64.0	78.0	58.8
	Sequential	45.5	68.0	46.0	52.0	66.0	92.0	**61.6**
	Interleaved	45.8	64.0	40.0	52.0	66.0	86.0	59.0
Harness	*Isolated*	45.5	72.0	40.0	50.0	64.0	88.0	**59.9**
	Sequential	41.8	28.0	34.0	52.0	68.0	90.0	52.3
	Interleaved	38.6	70.0	20.0	54.0	64.0	92.0	56.4
Claude Opus 4.7	Vanilla	41.2	86.0	70.0	38.0	68.0	80.0	63.9
ACE	*Isolated*	48.3	86.0	70.0	32.0	68.0	90.0	**65.7**
	Sequential	48.5	82.0	68.0	36.0	66.0	86.0	64.4
	Interleaved	47.5	82.0	68.0	26.0	68.0	78.0	61.6
A-Mem	*Isolated*	46.2	86.0	70.0	28.0	64.0	94.0	64.7
	Sequential	48.0	82.0	68.0	36.0	74.0	98.0	**67.7**
	Interleaved	49.0	80.0	70.0	30.0	78.0	90.0	66.2
ReasoningBank	*Isolated*	43.9	84.0	68.0	34.0	68.0	90.0	64.6
	Sequential	40.2	86.0	68.0	32.0	66.0	84.0	62.7
	Interleaved	44.1	84.0	68.0	36.0	68.0	92.0	**65.4**
AutoSkill	*Isolated*	42.6	86.0	74.0	36.0	72.0	78.0	64.8
	Sequential	41.2	86.0	74.0	36.0	64.0	78.0	63.2
	Interleaved	42.9	86.0	74.0	40.0	70.0	88.0	**66.8**
Harness	*Isolated*	43.8	84.0	70.0	28.0	66.0	88.0	63.3
	Sequential	42.5	84.0	68.0	34.0	72.0	94.0	**65.7**
	Interleaved	45.9	84.0	68.0	38.0	68.0	82.0	64.3

Table 12: Main results (%) of self-evolving methods across three streaming scenarios under seed 44. Red background indicates scores below the model's vanilla baseline, and **blue** marks the best scenario within each self-evolving method.

Methods	Mode	AppWorld	BFCL	BrowseComp+	HLE	SWE	Tau2	Avg
GPT-5.4	Vanilla	44.6	66.0	50.0	2.0	62.0	50.0	45.8
ACE	*Isolated*	44.8	76.0	48.0	6.0	64.0	62.0	**50.1**
	Sequential	44.8	68.0	42.0	6.0	62.0	54.0	46.1
	Interleaved	34.5	58.0	46.0	8.0	58.0	48.0	42.1
A-Mem	*Isolated*	31.4	66.0	50.0	6.0	66.0	54.0	45.6
	Sequential	34.8	62.0	54.0	12.0	62.0	48.0	45.5
	Interleaved	41.2	62.0	54.0	6.0	66.0	94.0	**53.9**
ReasoningBank	*Isolated*	39.7	56.0	44.0	12.0	60.0	44.0	42.6
	Sequential	42.3	64.0	50.0	12.0	60.0	24.0	42.1
	Interleaved	36.6	68.0	50.0	14.0	54.0	58.0	**46.8**
AutoSkill	*Isolated*	41.4	70.0	46.0	4.0	58.0	52.0	**45.2**
	Sequential	39.2	70.0	44.0	6.0	58.0	46.0	43.9
	Interleaved	38.7	76.0	44.0	4.0	60.0	44.0	44.5
Harness	*Isolated*	34.9	54.0	44.0	4.0	64.0	44.0	40.8
	Sequential	33.4	68.0	46.0	6.0	56.0	56.0	**44.2**
	Interleaved	39.1	60.0	56.0	6.0	56.0	42.0	43.2
Gemini 3.1 Pro	Vanilla	41.8	58.0	34.0	52.0	64.0	90.0	56.6
ACE	*Isolated*	43.6	66.0	38.0	50.0	64.0	80.0	56.9
	Sequential	44.6	80.0	44.0	52.0	68.0	92.0	**63.4**
	Interleaved	38.4	52.0	50.0	50.0	66.0	100.0	59.4
A-Mem	*Isolated*	46.3	26.0	46.0	54.0	64.0	90.0	54.4
	Sequential	40.5	56.0	44.0	50.0	62.0	94.0	**57.8**
	Interleaved	43.0	38.0	44.0	50.0	62.0	84.0	53.5
ReasoningBank	*Isolated*	39.2	52.0	52.0	52.0	66.0	94.0	59.2
	Sequential	41.9	56.0	46.0	50.0	62.0	94.0	58.3
	Interleaved	44.5	48.0	56.0	54.0	64.0	96.0	**60.4**
AutoSkill	*Isolated*	44.1	60.0	50.0	48.0	70.0	88.0	60.0
	Sequential	44.1	68.0	38.0	46.0	62.0	86.0	57.4
	Interleaved	43.5	66.0	48.0	48.0	68.0	90.0	**60.6**
Harness	*Isolated*	36.0	74.0	40.0	56.0	72.0	86.0	**60.7**
	Sequential	43.0	68.0	38.0	52.0	66.0	96.0	60.5
	Interleaved	41.9	76.0	42.0	50.0	66.0	84.0	60.0
Claude Opus 4.7	Vanilla	41.2	86.0	70.0	38.0	68.0	80.0	63.9
ACE	*Isolated*	42.9	86.0	74.0	30.0	72.0	96.0	66.8
	Sequential	47.1	80.0	70.0	40.0	74.0	92.0	**67.2**
	Interleaved	45.9	84.0	68.0	36.0	74.0	76.0	64.0
A-Mem	*Isolated*	48.8	84.0	72.0	26.0	76.0	82.0	**64.8**
	Sequential	47.8	82.0	64.0	32.0	68.0	92.0	64.3
	Interleaved	50.3	82.0	66.0	32.0	66.0	78.0	62.4
ReasoningBank	*Isolated*	43.0	84.0	70.0	36.0	66.0	90.0	64.8
	Sequential	43.0	86.0	74.0	38.0	64.0	86.0	65.2
	Interleaved	45.6	86.0	74.0	42.0	68.0	86.0	**66.9**
AutoSkill	*Isolated*	42.8	86.0	74.0	36.0	62.0	84.0	64.1
	Sequential	42.1	86.0	74.0	36.0	62.0	90.0	65.0
	Interleaved	40.8	88.0	74.0	40.0	68.0	84.0	**65.8**
Harness	*Isolated*	44.9	82.0	68.0	38.0	74.0	90.0	**66.1**
	Sequential	41.9	80.0	70.0	36.0	72.0	42.0	57.0
	Interleaved	44.1	86.0	70.0	36.0	72.0	82.0	65.0

Table 13: Main results (%) of self-evolving methods across three streaming scenarios under seed 46. Red background indicates scores below the model's vanilla baseline, and **blue** marks the best scenario within each self-evolving method.

Methods	Mode	AppWorld	BFCL	BrowseComp+	HLE	SWE	Tau2	Avg
GPT-5.4	Vanilla	44.6	66.0	50.0	2.0	62.0	50.0	45.8
ACE	*Isolated*	39.3	74.0	48.0	4.0	58.0	58.0	46.9
	Sequential	38.7	66.0	50.0	8.0	68.0	62.0	**48.8**
	Interleaved	31.9	56.0	42.0	6.0	68.0	46.0	41.7
A-Mem	*Isolated*	50.3	70.0	46.0	8.0	64.0	84.0	**53.7**
	Sequential	43.1	74.0	50.0	8.0	54.0	56.0	47.5
	Interleaved	40.0	66.0	48.0	10.0	60.0	52.0	46.0
ReasoningBank	*Isolated*	40.0	62.0	44.0	10.0	64.0	36.0	42.7
	Sequential	43.4	64.0	44.0	20.0	68.0	32.0	45.2
	Interleaved	42.6	64.0	52.0	10.0	64.0	52.0	**47.4**
AutoSkill	*Isolated*	42.0	70.0	46.0	4.0	50.0	26.0	39.7
	Sequential	39.0	70.0	38.0	8.0	62.0	30.0	41.2
	Interleaved	39.4	74.0	48.0	6.0	62.0	30.0	**43.2**
Harness	*Isolated*	39.5	68.0	46.0	4.0	60.0	68.0	**47.6**
	Sequential	35.6	64.0	44.0	6.0	64.0	58.0	45.3
	Interleaved	31.4	70.0	48.0	6.0	56.0	52.0	43.9
Gemini 3.1 Pro	Vanilla	41.8	58.0	34.0	52.0	64.0	90.0	56.6
ACE	*Isolated*	44.9	48.0	48.0	50.0	64.0	98.0	58.8
	Sequential	38.2	62.0	38.0	52.0	68.0	98.0	59.4
	Interleaved	42.1	66.0	44.0	50.0	64.0	96.0	**60.4**
A-Mem	*Isolated*	43.7	74.0	50.0	52.0	56.0	98.0	**62.3**
	Sequential	43.8	56.0	48.0	52.0	60.0	90.0	58.3
	Interleaved	40.2	70.0	42.0	52.0	58.0	96.0	59.7
ReasoningBank	*Isolated*	44.7	52.0	54.0	44.0	58.0	98.0	58.5
	Sequential	42.8	54.0	44.0	52.0	72.0	98.0	**60.5**
	Interleaved	43.0	52.0	48.0	54.0	70.0	96.0	**60.5**
AutoSkill	*Isolated*	40.4	62.0	42.0	54.0	60.0	84.0	57.1
	Sequential	47.0	58.0	40.0	52.0	60.0	82.0	56.5
	Interleaved	44.9	66.0	36.0	50.0	64.0	86.0	**57.8**
Harness	*Isolated*	38.3	78.0	48.0	50.0	68.0	96.0	**63.1**
	Sequential	41.0	68.0	34.0	48.0	60.0	88.0	56.5
	Interleaved	42.3	78.0	52.0	50.0	60.0	96.0	**63.1**
Claude Opus 4.7	Vanilla	41.2	86.0	70.0	38.0	68.0	80.0	63.9
ACE	*Isolated*	51.2	84.0	72.0	38.0	72.0	94.0	**68.5**
	Sequential	46.9	86.0	68.0	40.0	72.0	96.0	68.2
	Interleaved	46.6	76.0	68.0	40.0	68.0	62.0	60.1
A-Mem	*Isolated*	51.2	86.0	72.0	28.0	74.0	96.0	**67.9**
	Sequential	48.6	84.0	72.0	30.0	66.0	98.0	66.4
	Interleaved	48.7	82.0	66.0	34.0	74.0	90.0	65.8
ReasoningBank	*Isolated*	45.1	84.0	72.0	38.0	74.0	90.0	**67.2**
	Sequential	42.1	84.0	72.0	36.0	70.0	86.0	65.0
	Interleaved	45.7	84.0	68.0	40.0	70.0	88.0	66.0
AutoSkill	*Isolated*	40.4	86.0	72.0	38.0	70.0	84.0	65.1
	Sequential	42.1	86.0	70.0	40.0	64.0	92.0	**65.7**
	Interleaved	38.1	86.0	72.0	34.0	64.0	88.0	63.7
Harness	*Isolated*	45.8	86.0	68.0	36.0	68.0	94.0	66.3
	Sequential	43.5	84.0	76.0	38.0	72.0	86.0	66.6
	Interleaved	48.2	86.0	70.0	38.0	68.0	98.0	**68.0**