SERVELEARNBENCH: HOW WELL CAN AGENTS SELF-IMPROVE FROM SERVING EXPERIENCE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents are increasingly deployed to perform complex tasks in real-world environments. However, the knowledge required for correct behavior in these environments is often implicit, undisclosed, and subject to change over time. Recent continual-learning harnesses seek to address this challenge by enabling agents to improve from serving experience. Yet the effectiveness and limitations of these methods are not yet well characterized. Existing benchmarks provide only partial coverage: some explicitly provide the target knowledge, others assume a static environment, and those that support continual adaptation remain limited in scale and knowledge diversity. To enable systematic evaluation, we formalize an *evolving-environment streaming dataset* (EESD), in which agents must infer, apply, and revise latent environment knowledge from interaction and outcome feedback as hidden policies evolve, and introduce ServeLearnBench, spanning retail support, banking, and sales-pitch generation with 53 environment windows and 7,718 tasks. We evaluate five learning harnesses (RAG, Mem0, SkillOpt, Continual Harness, and Prime) across six models (GPT-5.6 Terra, Opus 5, Kimi K3, GLM-5.3, DeepSeek V4.1 Flash, and GLM-5.3 Flash), covering 28 model–harness pairs and 252 learning runs. Our evaluation reveals three main findings: a substantial gap remains between task capability and learning from experience; continual adaptation is costly and can degrade already-correct behavior; and insufficient exploration emerges as a key bottleneck to effective adaptation. Overall, ServeLearnBench provides a controlled testbed for diagnosing these limitations and tracking progress toward agents that continually and reliably improve through serving experience.

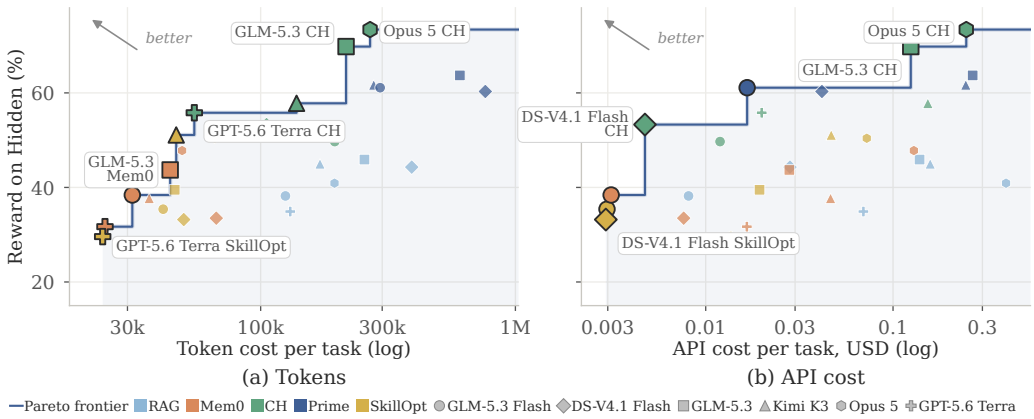


Figure 1: **Cost vs. Performance.** **(a)** Hidden-case reward versus token cost per test task (prefill, KV-cache hits and output tokens included). **(b)** Hidden-case reward versus API cost per test task. Reward is the Avg. H of Table 2 (mean over the nine settings). Lines connect Pareto-efficient pairs, for which no other plotted pair achieves both lower cost and higher reward.

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.

1 INTRODUCTION

Language agents increasingly handle complex real-world workflows (Yao et al., 2025; Zhou et al., 2024; Wei et al., 2025; Jimenez et al., 2024; Zheng et al., 2025b; Gao et al., 2025), but deployment is not a one-time test of capability. Operational policies, system constraints, and user preferences may change over time, often without corresponding updates to the information available to the agent. Reliable service therefore requires agents to learn from interaction and failure rather than continue acting on an increasingly outdated model of their environment. Recent continual-learning harnesses, including SkillOpt (Yang et al., 2026), Continual Harness (Karten et al., 2026b), and Prime (Karten et al., 2026a), aim to support this process by updating persistent memories, skills, or strategies from experience. Yet the ability to update an agent does not necessarily imply that it has acquired useful, reusable knowledge. This raises a central question: *how effectively can these systems infer previously unknown environment knowledge from serving experience, revise it as conditions change, and maintain reliable behavior over time?*

Understanding this capability requires answering three questions: **1)** How well do current agents learn continually from serving experience? **2)** What does this learning cost, in both computation and existing capabilities? **3)** What enables effective continual learning? Existing benchmarks have begun to study parts of this problem (Cheng et al., 2023; Wu et al., 2024; Zheng et al., 2025a; Shu et al., 2026; Guan et al., 2026; Asawa et al., 2026), but leave important gaps. Some *explicitly provide the target knowledge*, testing whether agents can retain and apply it rather than discover it through interaction and failure (Maharana et al., 2024; Wu et al., 2025; Hsieh et al., 2024). Others require agents to discover hidden knowledge but assume *static environments* (He et al., 2025; Jansen et al., 2024; Zheng et al., 2026), while benchmarks that support changing environments remain limited in *scale and knowledge diversity* (Asawa et al., 2026). These limitations make it difficult to systematically measure how agents acquire and revise hidden knowledge over time, quantify the costs of adaptation, and identify the mechanisms that make continual learning effective.

An ideal benchmark for continual learning from serving experience should satisfy three criteria. First, it should require *learning from interaction and failure*: agents must infer the knowledge needed for future tasks from their previous interactions with the environment, including failed attempts and the resulting feedback, without being explicitly told that knowledge. Second, it should feature *evolving environments*: policies, constraints, or preferences should change during the task stream, requiring agents to recognize when learned knowledge becomes outdated and adapt their behavior using new experience. Third, it should offer *sufficient scale*: many tasks involving different kinds of environment knowledge should support systematic analysis of how agents acquire, retain, apply, and revise that knowledge over extended serving histories.

In this paper, we formalize this setting as an *evolving-environment streaming dataset* (EESD): a chronological stream of tasks governed by latent environment knowledge that evolves over time. We introduce ServeLearnBench as a controlled suite of EESDs (Figure 2). To enforce *learning from interaction and failure*, the target environment knowledge is never explicitly revealed in task inputs or feedback, agents must instead infer it from their accumulated experience interacting with the environment. To instantiate *evolving environments*, the hidden policy changes over time rather than remaining fixed, requiring agents to continually adapt their knowledge and behavior. Finally, to provide *sufficient scale*, ServeLearnBench comprises 53 environment windows, 4,508 serving tasks, and 3,210 held-out test tasks across retail support, banking case review, and sales-pitch generation, covering categorical rules, numeric and compositional thresholds, and latent preferences.

To evaluate how effectively current continual learning methods learn from serving experience and improve future behavior, we compare five learning harnesses across six models. We conduct the evaluation on five harnesses (RAG (Lewis et al., 2020), Mem0 (Chhikara et al., 2025), SkillOpt (Yang et al., 2026), Continual Harness (CH) (Karten et al., 2026b), Prime (Karten et al., 2026a)) and six models (GPT-5.6 Terra (OpenAI, 2026), Opus 5 (Anthropic, 2026), Kimi K3 (Kimi Team, 2026), GLM-5.3 (Z.ai, 2026a), DeepSeek V4.1 Flash (DeepSeek-AI, 2026), and GLM-5.3 Flash (Z.ai, 2026b)). The evaluation yields three key findings:

1. **Performance gaps remain in continual learning.** (Section 4.2) All models can perform the tasks once the policy is disclosed: Oracle achieves 95.4% accuracy on Hidden cases on average. Yet no agent learns the policy reliably. On Retail L3, the best of the 28 model–harness pairs reaches only 59.6%, while the median pair achieves 22.0%.

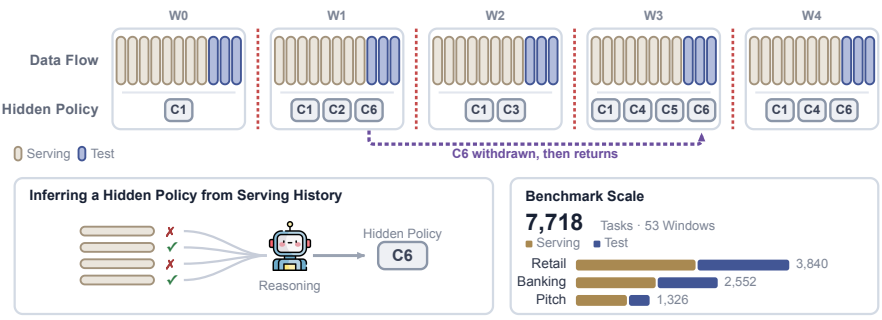


Figure 2: **ServeLearnBench protocol.** *Top:* the stream is divided into environment windows, each containing feedback-provided serving tasks followed by feedback-hidden test tasks. The active policy in window s consists of behavioral components $\mathcal{C}(\theta_s)$, which may be introduced, withdrawn, or reactivated (dashed lines). *Bottom left:* the learner receives neither policy-component descriptions nor corrections, and must infer the active policy from serving interactions and outcome feedback. *Bottom right:* benchmark scale across domains, tiers, windows, and tasks.

2. **No free lunch for continual learning.** (Sections 4.3 and 5.2) Harness with continual learning support is more expensive and can degrade already-correct behavior. Inferring a hidden policy costs 4–101$\times$ as much as following a disclosed one: on Kimi K3, Prime spends \$2,479 compared with \$102 for Oracle. Adaptation also hurts performance on tasks that require no learning.
3. **Effective exploration is critical for continual learning.** (Section 5.3) A harness's ability to explore during unsuccessful adaptation is strongly associated with its final performance. Harnesses that vary their candidate behaviors more achieve higher overall reward. This relationship highlights effective exploration as an important design target for future continual-learning harnesses.

2 Related Work

Memory and continual-learning benchmarks. Interactive-agent benchmarks first established outcome-based evaluation of policy following and tool use; τ-bench is especially relevant because it verifies user interaction, tool execution, and final database state under a given policy (Yao et al., 2025). Feedback-stream benchmarks then made improvement across tasks measurable: LLF-Bench provides rich language feedback, whereas StreamBench includes continuous input–feedback streams and binary outcomes (Cheng et al., 2023; Wu et al., 2024). LifelongAgentBench, AgentCL, ContinualSkillBench, SkillLearnBench, and CL-Bench extend this view to acquisition, reuse, composition, and retention across sequential tasks and persistent agent states (Zheng et al., 2025a; Shu et al., 2026; Guan et al., 2026; Zhong et al., 2026; Asawa et al., 2026). StateMemBench more specifically tests whether memory systems track revised facts and distinguish the current state from superseded ones (Fan et al., 2026b). Dynamic-environment suites add complementary forms of change: EvoArena (Xu et al., 2026) updates interactive environments, while Harrington et al. (2026) separate performance on drifting and stable knowledge. EnvHarness instead reshapes static environments into adaptive practice environments while retaining their verifiers (Huang et al., 2026). These benchmark families emphasize different capabilities rather than sharing one limitation. What remains only partially combined is an evaluator-controlled evolving environment in which the operative state is hidden, feedback reveals outcomes rather than corrections, prior knowledge can become stale or valid again, and learning gains can be separated from collateral effects through matched controls. ServeLearnBench targets this intersection with evolving-environment streaming datasets, Blind/Oracle references, and Hidden versus Fully Specified evaluation slices.

Continual learning for self-evolving agents. Self-evolving agents differ in the state they update and reuse. RAG and Mem0 retain prior evidence; ReasoningBank distills strategies from successes and failures, WorldEvolver revises predictive rules from observed transitions, and Reflexion and ExpeL preserve verbal lessons (Lewis et al., 2020; Chhikara et al., 2025; Ouyang et al., 2026; Zhang et al., 2026c; Shinn et al., 2023; Zhao et al., 2024). GEPA, RHO, RHI, and SkillOpt revise prompts, harness specifications, or skills using trajectory feedback, while ACE curates context and PolicyBank refines structured policy knowledge (Agrawal et al., 2026; Pan et al., 2026; Lee et al., 2026; Yang et al., 2026; Zhang et al., 2026b; Choi et al., 2026). Continual Harness and Prime update broader

harness state during interaction; HCL examines retention during such updates, Adaptive Auto-Harness routes changing task streams, JIT-Agent generates task-specific harnesses, and RSIAgent explores new environments to build reusable knowledge (Karten et al., 2026b;a; Kang et al., 2026; Liu et al., 2026b; Zhang et al., 2026a; Zhu et al., 2026). AlphaEvolve and EvoX instead evolve code and search strategies, while controlled harness-design experiments isolate the effects of planning, actions, and context management across models (Novikov et al., 2025; Liu et al., 2026a; Fan et al., 2026a). ServeLearnBench evaluates how well agents infer hidden knowledge from serving outcomes, revise it after changes, and preserve behavior on Fully Specified cases under a common task stream.

3 SERVELEARNBENCH

In this section, we present the design of ServeLearnBench. We first formalize the evolving-environment streaming dataset abstraction (Section 3.1), then describe task creation, reward metrics, and reference baselines across domains and difficulty tiers (Section 3.2).

3.1 EVOLVING-ENVIRONMENT STREAMING DATASET (EESD)

To evaluate continual learning from experience, a benchmark should require agents to infer hidden environment knowledge and adapt when that knowledge changes. We capture this with an *evolving-environment streaming dataset* (EESD), a chronological task stream in which the learner-visible interface remains fixed while the evaluator-private environment evolves over time. This formulation tests if agents can acquire, apply, and revise environment knowledge from prior interactions.

Stream and environment windows. For each domain–tier instance, ServeLearnBench defines an EESD as a chronological sequence of environment windows, $\mathcal{D} = W_0 \,\|\, W_1 \,\|\, \cdots \,\|\, W_{S-1}$, where $\|$ denotes temporal concatenation. Each window W_s is governed by a latent environment state $\theta_s \in \Theta_g$ that remains fixed within the window and may change at window boundaries. Across windows, the environment may introduce new policies, revise existing ones, or return to previously active policies. Neither the active state nor its changes are disclosed to the learning harness.

Serving and test splits. Each environment window contains serving tasks followed by test tasks, $W_s = \mathcal{D}_s^{\text{serve}} \,\|\, \mathcal{D}_s^{\text{test}}$. Serving and test tasks cover the same behavioral components active under θ_s, while using disjoint task instances. Let $\mathcal{C}(\theta_s)$ denote the set of active components. We require

$$\text{Cov}(\mathcal{D}_s^{\text{serve}}) = \text{Cov}(\mathcal{D}_s^{\text{test}}) = \mathcal{C}(\theta_s), \qquad \mathcal{D}_s^{\text{serve}} \cap \mathcal{D}_s^{\text{test}} = \emptyset. \tag{1}$$

This ensures that test tasks evaluate transfer to new instances under the same active policy rather than memorization of serving examples.

Hidden-Dependent and Fully Specified data. Orthogonal to the serving/test split, each task is categorized by whether correct behavior depends on the latent environment state. A task is *Hidden-Dependent* if learner-visible information alone is insufficient to determine the correct behavior, and *Fully Specified* otherwise. Hidden-Dependent tasks measure whether agents acquire and apply hidden environment knowledge, whereas Fully Specified tasks measure whether adaptation preserves behavior that is already supported by visible information.

3.2 SERVELEARNBENCH: TASK CREATION

ServeLearnBench defines nine scenarios by crossing three domains (Retail, Banking, and Pitch) with three difficulty tiers. Each scenario follows the same EESD protocol while fixing its task interface, verifier, hidden policies, and window schedule:

Retail. Retail is a multi-turn support environment covering four core request types: canceling or modifying pending orders, and returning or exchanging delivered items. Tasks vary users, orders, products, payment methods, destinations, and single- versus multi-item requests, with harder variants requiring multi-item resolution or fallback selection. Each task operates on an isolated database and is considered correct only if it reaches the intended final state or produces the correct refusal.

Banking. Banking is a case-review environment covering three core decision types: authorizing pending card transactions, reviewing credit-limit increases, and screening outbound transfers. Harder variants compose these decisions into ranked alternatives or multi-line payment batches, requiring selection of the first admissible option or a verdict for each line. Each task operates on an account and case in a database-backed environment, with hidden policies governing verification requirements, spending and travel limits, account standing, requested-limit caps, merchant restrictions, and payee-age thresholds.

Table 1: Detailed benchmark statistics (W: number of windows). Appendix C provides more details.

Scenario	W	Serving	Test
Retail L1	6	720	384
Retail L2	7	810	765
Retail L3	8	624	537
Banking L1	4	336	178
Banking L2	5	500	387
Banking L3	6	600	551
Pitch L1	4	216	96
Pitch L2	6	324	144
Pitch L3	7	378	168
Total	**53**	**4,508**	**3,210**

Pitch. Pitch is a single-turn generation environment in which each task provides a factual product sheet and asks the agent to produce a 40–80 word sales pitch. The hidden state is the active customer preference profile, and a graded verifier rewards coverage of positively valued attributes while penalizing undesirable attributes and unsupported claims. Because the appropriate emphasis always depends on this latent profile, all Pitch tasks are Hidden-Dependent.

Difficulty design. The three difficulty tiers vary how environment knowledge changes over time and how task complexity is expressed across domains. *L1 (acquisition)* introduces previously unseen states, using mostly single-policy Retail and Banking tasks and non-repeating Pitch profiles. *L2 (revision and composition)* revises earlier states, adds multi-policy Retail requests and ranked or batched Banking decisions, and introduces recurring Pitch profiles over a broader set of attributes. *L3 (non-monotonic evolution)* further introduces reversals, removals, and re-entry while retaining L2-style Retail and Banking composition; Pitch likewise cycles back to previously seen profiles.

Reward metric. ServeLearnBench uses domain-specific reward functions. Retail rewards correct database transitions or refusal reasons; Banking rewards correct verdicts and resulting states; and Pitch uses a structured rubric that rewards preference coverage while penalizing undesirable attributes and unsupported claims. Appendix C.6 provides the full evaluation details.

Reference baselines. We include two non-learning baselines to contextualize performance. *Blind* uses only learner-visible task information and retains no serving experience, measuring performance without adaptation. *Oracle* additionally receives the active hidden policy for each window, measuring performance when hidden-state inference is removed while all other task demands remain unchanged.

4 Evaluation

We evaluate ServeLearnBench by crossing models with learning harnesses across all scenario–difficulty settings. We first describe the experimental setting (Section 4.1), then present the main evaluation matrix (Section 4.2), and finally organize the analysis around three questions: how much learnable headroom the benchmark creates, how much of that headroom a streaming learner captures, and whether adaptation harms fully specified cases.

4.1 Experimental Setting

Model. The main evaluation compares six models: GPT-5.6 Terra (OpenAI, 2026), Opus 5 (Anthropic, 2026), Kimi K3 (Kimi Team, 2026), GLM-5.3 (Z.ai, 2026a), DeepSeek V4.1 Flash (DeepSeek-AI, 2026), and GLM-5.3 Flash (Z.ai, 2026b). GPT-5.6 Terra and Opus 5 are accessed through the OpenAI and Anthropic APIs, respectively; the other four models are accessed through Fireworks.

Harnesses. The main comparison includes Blind and Oracle references alongside five learning harnesses: RAG (Lewis et al., 2020), Mem0 (Chhikara et al., 2025), SkillOpt (Yang et al., 2026), Continual Harness (CH) (Karten et al., 2026b), and Prime (Karten et al., 2026a). Crossing six models with the five harnesses yields 30 possible model–harness pairs across nine scenario–difficulty

Table 2: Performance comparison of five harnesses across six models on ServeLearnBench.

Model	Harness	Retail						Banking						Pitch			Avg.		Cost
		Hidden			Fully Specified			Hidden			Fully Specified			Hidden			H	F	
		L1	L2	L3	L1	L2	L3	L1	L2	L3	L1	L2	L3	L1	L2	L3			
Kimi K3 (2.8T-A104B)	Oracle	100.0	96.0	95.5	100.0	100.0	99.7	100.0	98.9	98.7	100.0	100.0	99.2	89.2	92.1	89.5	95.5	99.8	102
	Blind	1.7	0.0	2.2	100.0	100.0	99.0	1.2	14.0	16.0	95.7	96.5	96.6	31.7	25.9	18.3	12.3	98.0	120
	RAG	65.4	18.8	15.7	86.8	90.8	95.9	69.0	60.8	35.8	91.5	85.1	86.1	47.9	51.0	40.8	45.0	89.4	1,073
	Mem0	20.4	14.4	23.8	94.4	91.6	94.6	69.0	55.4	35.5	97.9	93.5	87.8	34.4	48.1	38.0	37.7	93.3	829
	CH	100.0	54.2	33.6	99.3	82.0	83.1	51.2	54.3	51.1	69.1	87.1	84.9	68.4	56.0	51.4	57.8	84.2	1,335
	Prime	87.1	61.6	52.0	95.8	89.1	93.6	97.6	79.0	67.4	98.9	91.5	94.1	64.0	16.9	29.9	61.7	93.8	2,479
	SkillOpt	38.8	4.8	37.7	81.9	87.9	86.6	46.4	68.8	59.7	79.8	88.6	81.1	65.3	74.2	64.6	51.1	84.3	605
GLM-5.3 (743B-A40B)	Oracle	100.0	95.2	92.4	99.3	97.9	97.8	100.0	100.0	99.7	100.0	100.0	99.2	79.4	90.1	85.7	93.6	99.0	29.1
	Blind	13.3	0.0	1.8	100.0	99.6	95.9	0.0	16.7	15.7	91.5	92.5	92.4	32.6	25.8	19.5	13.9	95.3	40.2
	RAG	62.9	30.0	15.2	91.7	87.9	84.4	58.3	64.0	55.6	88.3	89.6	84.0	45.0	44.5	37.4	45.9	87.6	932
	Mem0	20.8	12.4	17.5	98.6	96.2	95.5	66.7	69.4	58.1	98.9	94.0	92.4	51.3	52.0	45.0	43.7	95.9	565
	CH	100.0	72.8	59.6	99.3	93.7	98.4	88.1	89.2	59.4	93.6	98.0	97.1	76.3	41.9	40.9	69.8	96.7	1,077
	Prime	90.8	61.4	39.5	97.2	87.9	98.1	94.0	69.4	78.9	97.9	96.5	96.2	73.8	41.4	24.0	63.7	95.6	2,930
	SkillOpt	69.2	0.0	4.9	91.7	92.9	96.5	81.0	29.0	31.3	90.4	90.5	86.1	49.7	64.7	25.6	39.5	91.4	251
DeepSeek V4.1 Flash (552B-A16B)	Oracle	100.0	99.6	93.7	100.0	99.6	100.0	100.0	100.0	100.0	100.0	100.0	99.6	91.4	90.6	85.1	95.6	99.9	7.4
	Blind	1.2	0.2	0.9	100.0	99.2	99.0	0.0	19.4	15.7	88.3	82.1	78.2	37.8	32.4	21.9	14.4	91.1	11.0
	RAG	51.2	23.6	13.9	91.7	80.8	80.3	52.4	48.4	32.6	85.1	77.1	84.9	68.7	61.3	46.7	44.3	83.3	192
	Mem0	18.8	4.2	13.9	95.8	90.4	98.7	44.0	36.0	28.1	95.7	96.5	88.7	59.8	49.0	47.8	33.5	94.3	75.9
	CH	50.0	24.1	22.4	95.8	77.8	99.0	83.3	61.8	73.8	89.4	91.0	95.0	54.4	52.0	58.1	53.3	91.3	52.0
	Prime	80.0	64.1	25.1	95.1	86.2	92.0	71.4	70.4	64.9	97.9	89.1	94.1	76.1	50.9	40.2	60.3	92.4	429
	SkillOpt	23.8	12.0	22.9	93.8	85.8	86.6	19.0	37.6	31.9	85.1	95.5	75.6	49.1	53.6	48.6	33.2	87.1	39.8
GLM-5.3 Flash (320B-A18B)	Oracle	98.8	93.5	89.2	100.0	95.4	95.5	100.0	98.9	99.4	100.0	100.0	100.0	83.9	89.9	83.9	93.1	98.5	5.0
	Blind	14.6	0.4	4.5	100.0	99.6	94.3	0.0	15.6	17.3	95.7	95.0	95.4	33.8	25.4	20.2	14.6	96.7	6.8
	RAG	45.4	14.6	14.3	100.0	88.7	95.2	69.0	54.3	29.7	95.7	85.1	81.9	50.1	44.5	22.3	38.2	91.1	56.3
	Mem0	18.3	4.6	13.5	99.3	97.9	92.7	67.9	57.0	49.2	87.2	90.0	89.1	51.9	43.2	39.7	38.4	92.7	48.8
	CH	75.8	29.7	28.7	97.9	85.4	92.0	53.6	47.3	69.3	63.8	76.6	81.1	48.2	57.6	36.7	49.7	82.8	88.5
	Prime	76.7	32.1	26.9	99.3	88.3	97.5	94.0	84.4	78.6	98.9	93.0	94.5	70.0	48.8	38.6	61.1	95.2	172
	SkillOpt	20.8	31.7	22.0	100.0	87.4	93.3	32.1	34.9	38.0	80.9	70.1	79.0	40.4	52.2	46.4	35.4	85.1	35.1
Opus 5	Oracle	100.0	99.6	99.6	100.0	100.0	99.7	100.0	100.0	100.0	100.0	100.0	100.0	90.8	91.0	93.3	97.1	100.0	131
	Blind	0.0	1.9	1.8	100.0	97.9	99.7	0.0	23.7	18.8	69.1	52.7	58.8	36.3	30.9	17.3	14.5	79.7	207
	RAG	64.2	23.8	17.5	88.9	84.5	87.6	66.7	42.5	25.9	94.7	86.6	86.6	48.6	43.0	35.7	40.9	88.1	2,602
	Mem0	27.9	14.4	15.7	90.3	80.3	85.4	58.3	67.2	55.0	97.9	94.0	93.7	68.9	52.1	70.9	47.8	90.3	2,003
	CH	97.9	51.5	30.5	99.3	93.7	97.8	82.1	80.1	76.0	96.8	95.0	94.5	99.7	83.6	58.9	73.4	96.2	2,449
	SkillOpt	69.2	1.1	17.0	97.9	93.3	89.8	57.1	67.7	49.5	94.7	87.6	92.9	57.6	72.6	61.9	50.4	92.7	629
GPT-5.6 Terra	Oracle	99.6	95.1	91.9	99.3	98.7	98.1	100.0	98.9	98.1	96.8	99.0	100.0	98.5	98.1	98.6	97.6	98.6	35.6
	Blind	0.0	0.0	0.0	100.0	97.9	99.4	0.0	14.5	16.6	97.9	95.0	96.6	43.9	35.0	25.5	15.1	97.8	39.6
	RAG	27.9	11.4	13.9	97.9	92.9	94.3	52.4	30.1	23.6	91.5	87.1	84.9	59.7	55.9	39.6	34.9	91.4	496
	Mem0	0.8	0.2	6.3	100.0	97.9	98.4	56.0	15.1	30.4	100.0	88.6	91.6	72.2	65.5	38.9	31.7	96.1	206
	CH	53.3	23.4	30.5	96.5	82.4	98.7	86.9	57.0	69.6	95.7	96.0	95.8	63.7	63.6	54.6	55.8	94.2	239
	SkillOpt	34.2	0.2	26.9	100.0	87.0	94.3	19.0	14.0	42.8	100.0	99.5	86.6	55.0	42.7	31.2	29.6	94.6	146

settings; 28 were evaluated, with two omitted due to cost (Table 2). We use identical task streams and scoring across methods while preserving each harness's native runtime, update schedule, replay access, learning computation, and state capacity. Implementation details and operating conditions are provided in Appendix B.2, with model and evaluation settings in Appendix B.

4.2 Main Evaluation Results

Table 2 compares the performance of different model–harness combinations on ServeLearnBench. Across six models, Oracle achieves an average Hidden score of 95.4, compared with 14.1 for Blind, showing that access to environment knowledge substantially improves performance. However, knowing that knowledge does not always lead to better behavior: on Pitch L1, Opus 5 scores 90.8 with Oracle, below the 99.7 achieved with CH. When informed of the customer's dislikes, Oracle sometimes produces misleading claims to appeal to those preferences. For example, it describes an already-trending product as one to buy "before word spreads," contradicting the product sheet and incurring penalties. CH instead learns to mention the preferred attributes and omit the others; both achieve full preference coverage, but CH avoids nearly all such penalties.

Among the learning harnesses, Prime and CH generally achieve stronger Hidden performance than RAG, Mem0, and SkillOpt, although substantial gaps to Oracle remain. Figure 3 summarizes this pattern across the four open-weight models: Prime and CH average 61.7 and 57.6, respectively, compared with 38.3–43.4 for the other three harnesses. On Kimi K3, for example, Prime reaches 61.7, versus 45.0 for RAG and 37.7 for Mem0. The ranking is not uniform across models: CH performs best on GLM-5.3, scoring 69.8 against Prime's 63.7, while Prime leads on the other three

open-weight models. Higher Hidden performance can also come with losses on Fully Specified tasks, so improvements in learning must be considered alongside preservation of existing capabilities (Section 5.2).

4.3 Cost Comparison

Performance–resource frontiers. Accuracy alone does not determine whether a deployed learner is useful: adaptation changes both the reasoning emitted by the model and the auxiliary calls made by its harness. Figure 1 compares reward on Hidden cases with the tokens consumed and the API cost per held-out test task; tokens count every input token of every model call, cached or not, plus the output tokens. Figure 1 covers the 28 model–harness pairs of Table 2 (252 learning lanes) and uses the table's held-out test Avg. H as its reward axis. The grouped cost comparisons use the four open-weight models (180 lanes), with costs computed from archived runs and rewards averaged equally across the nine settings. Figure 4 combines serving and test tasks within each setting, whereas Figure 12 reports the two phases separately. The dashboard labels *adapt* and *general* correspond to Hidden and Fully Specified cases, respectively. API cost accounts for uncached input, cache reads, cache writes, and output at each provider's official prices, so token efficiency and monetary efficiency can induce different rankings.

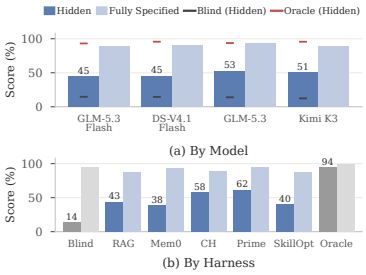


Figure 3: **Average performance** of the four open-weight models, **(a)** by model and **(b)** by harness.

Cost per task against reward. Figure 4 compares average per-task API cost with reward, grouped by acting model and learning harness. Cost includes both serving and held-out test execution, including any serving-side learning computation, and reward is averaged across the same tasks and then equally across the nine settings. Across harnesses, higher cost generally accompanies higher Hidden reward: Prime achieves the highest reward (59.0) at \$0.195 per task, while CH reaches 55.1 at less than half the cost (\$0.083). RAG, Mem0, and SkillOpt are cheaper but trail CH by 13–20 points. Across models, GLM-5.3 and Kimi K3 achieve the highest Hidden reward (49.8 and 49.2), but at 14–16× the per-task cost of GLM-5.3 Flash (\$0.010), which reaches 42.0. Appendix D.4 provides phase-level and learning-cost breakdowns.

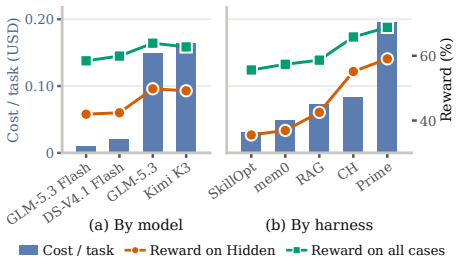


Figure 4: **Cost vs. performance.** Average API cost per task and reward, aggregated **(a)** by model and **(b)** by learning harness. Bars show cost, including learning; lines show Hidden and overall reward, averaged equally across the nine settings. Blind and Oracle are excluded.

5 Analysis

The comparison above ranks harnesses and models by their final scores; this section asks how those scores come about: how quickly a harness picks up a changed policy, what adaptation does to the cases that needed no learning, and whether the way a harness explores and retains predicts its reward.

5.1 How Fast Do Harnesses Learn a New Policy?

Final performance does not reveal how quickly a harness adapts to a new policy. We therefore track each policy from the window in which it becomes active until it changes again, and measure correctness on the k-th Hidden serving example governed by that policy. We perform this analysis on Banking and Retail, where Hidden tickets can be attributed to explicit rules and states; Pitch is excluded because its latent preference profiles do not decompose into such rules. The analysis uses the recorded serving streams of four open-weight models across all five learning harnesses, covering 31 Banking and 37 Retail policies (Appendix D.1).

Let a_k denote mean correctness after exactly k previous examples of the same policy. We summarize learning speed using $\mathrm{AULC}_K = \frac{1}{K}\sum_{k=0}^{K-1} a_k$, with $K = 10$. AULC_{10} measures average correctness

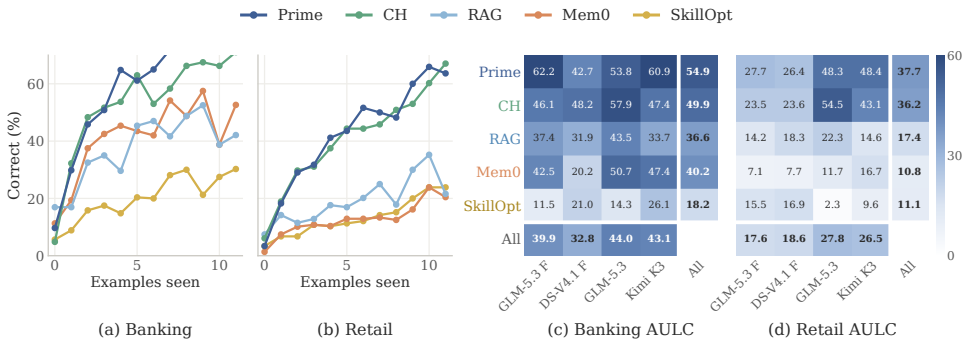


Figure 5: **Learning speed for new policies. (a, b)** Correctness on Hidden serving tickets as a function of the number of prior tickets from the same policy, pooled across four open-weight models for Banking and Retail, respectively. **(c, d)** AULC_{10} (%) by model and harness.

over the first ten encounters with a new policy, rewarding both rapid and high-quality learning without requiring a threshold for when a policy is considered learned.

Figure 5 reveals substantial differences in learning speed across harnesses. Prime and CH adapt fastest in both domains, reaching AULC_{10} scores of 54.9 and 49.9 in Banking and 37.7 and 36.2 in Retail; after five examples of a new Banking policy, both answer roughly 60% of subsequent tickets correctly. Mem0 and RAG learn more slowly, particularly in Retail, while SkillOpt is slowest in both domains, with AULC_{10} scores of 18.2 in Banking and 11.1 in Retail. Learning speed also varies substantially across acting models: GLM-5.3 and Kimi K3 achieve the highest pooled AULC_{10} in Banking, and Retail is consistently harder to learn across models. Appendix B.2 discusses harness-specific learning mechanisms, and Appendix D.2 compares acquiring a policy with dropping it once the corresponding rule is withdrawn.

5.2 Does Adaptation Harm Fully Specified Cases?

Fully Specified scores reveal effects that Hidden improvements alone cannot capture. Figure 6 averages them over the four open-weight models: every learning harness loses Fully Specified accuracy against Blind in Retail (98.9 for Blind against 89.5–95.5), and all but Prime and Mem0 lose in Banking (91.7 against 83.6–86.2, Prime 95.2, Mem0 92.7). On GLM-5.3 Flash, CH raises average Hidden performance from Blind's 14.6 to 49.7, while Fully Specified performance falls from 96.7 to 82.8. Prime attains higher Hidden performance, 61.1, while retaining 95.2 on Fully Specified cases. Pitch contributes only to Hidden averages because the operative customer preferences remain latent.

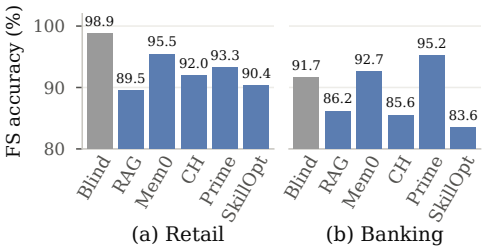
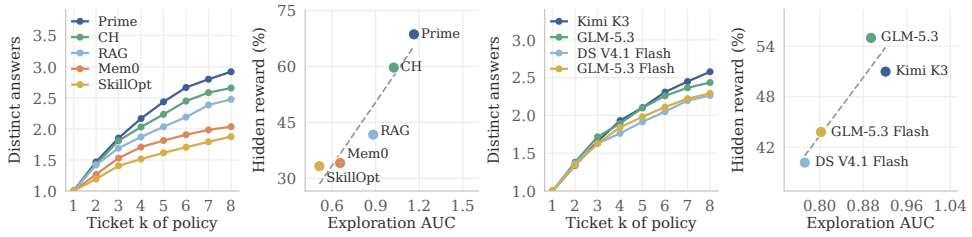


Figure 6: **Fully Specified accuracy with and without adaptation.** Mean over four open-weight models and three difficulty levels.

5.3 Does Exploration Predict Adaptation?

Exploration. A harness receives only outcome feedback after each serving ticket, so discovering a changed policy often requires trying behavior beyond what the visible specification suggests. We therefore measure *exploration* by how many distinct answers a harness produces while encountering the same hidden policy, regardless of correctness. We analyze Retail policies whose correct behavior is a refusal and Banking tickets governed by a single rule; Appendix D.3 provides more details.

For a policy, let c_k denote the number of distinct answers produced over its first k Hidden serving tickets. We summarize exploration over the first K tickets as $\text{AUC}_{\text{E}} = \frac{1}{K}\sum_{k=1}^{K}(c_k - 1)$, with $K = 8$. A harness that always repeats the same answer has $\text{AUC}_{\text{E}} = 0$, while larger values indicate greater behavioral exploration. Results are robust to $K \in [4, 12]$ (Appendix D.3).



(a) Diversity by harness (b) Reward vs. AUC, ρ=1.00 (c) Diversity by model (d) Reward vs. AUC, ρ=0.80

Figure 7: **Exploration across models and harnesses. (a, c)** Mean number of distinct answers after k Hidden serving examples of the same policy, aggregated by harness and model. **(b, d)** Exploration AUC versus mean Hidden test reward across the six Retail and Banking settings; ρ denotes Spearman rank correlation. Per-domain results are provided in Appendix D.3.

Figure 7 shows a strong association between exploration and adaptation performance. Prime, CH, RAG, Mem0, and SkillOpt have exploration AUCs of 1.16, 1.02, 0.88, 0.65, and 0.51, respectively, exactly matching their ordering in mean Hidden reward and yielding $\rho = 1.00$. The relationship persists at the finer harness–domain–tier level ($\rho = 0.75$; Appendix D.3). The gap is especially pronounced in Retail, where Prime tries substantially more distinct answers while Mem0 and SkillOpt often continue repeating the visible-specification behavior. Across models, the same relationship holds but variation is smaller: exploration spans 0.77–0.92 across models, compared with 0.51–1.16 across harnesses. These results suggest that effective exploration is primarily a property of the learning harness and is strongly associated with successful continual adaptation.

Knowledge Acquisition. A first success does not necessarily mean that an agent has learned the underlying policy. We therefore measure whether that success becomes reusable knowledge by aligning each policy at its first correct ticket and tracking correctness over the next K occurrences. Letting r_j denote correctness on the j-th subsequent ticket, we define the *knowledge-acquisition AUC* as $\mathrm{AUC}_{\mathrm{KA}} = \frac{1}{K}\sum_{j=1}^{K} r_j$, and use $K = 5$ throughout our analysis (Appendix D.3).

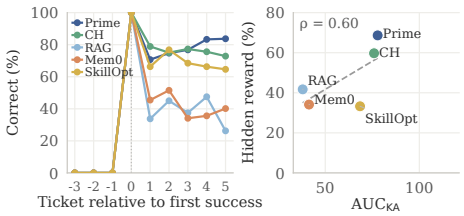


Figure 8: **Knowledge acquisition after first success.** Left: correctness on Hidden serving tickets aligned at the first correct response (ticket 0), averaged across Retail and Banking. Right: mean correctness over the next five tickets ($\mathrm{AUC}_{\mathrm{KA}}$) versus mean Hidden test reward; ρ denotes Spearman rank correlation.

Figure 8 shows large differences across harnesses. Prime and CH achieve $\mathrm{AUC}_{\mathrm{KA}}$ values of 78% and 76%, followed by SkillOpt at 68%, while Mem0 and RAG reach only 41% and 38%. The post-success curves are largely flat, suggesting that successful behavior either becomes reusable almost immediately or remains unreliable over subsequent encounters. Knowledge acquisition also differs from exploration: its rank correlation with Hidden reward is only 0.60. SkillOpt performs well once it discovers the correct behavior but struggles to find it in the first place, whereas RAG often discovers the correct behavior but fails to reuse it reliably.

6 Conclusion

We introduce ServeLearnBench, a benchmark for continual learning from serving experience in hidden, evolving environments. Across three domains and nine scenarios, our evaluation shows that substantial gaps remain in acquiring and revising hidden environment knowledge, even when the underlying tasks are well within model capabilities once the relevant policy is disclosed. Continual adaptation also comes with significant computational cost and can degrade behavior that was already correct. Our analyses further identify exploration and knowledge acquisition as distinct bottlenecks: agents must not only discover useful behavior, but also convert successful experience into reusable knowledge. We hope ServeLearnBench provides a systematic testbed for diagnosing these challenges and tracking progress toward agents that learn continually and reliably from deployment experience.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting and revision, code development, figure preparation, and interpretation of experimental results. Separately, language models serve as the evaluated agents and as the Pitch judge; their configurations and scoring procedures are documented in Appendices B and C.6. We reviewed all AI-assisted work. The authors take responsibility for the final manuscript, implementation, and reported findings, including all AI-assisted content.

ETHICS STATEMENT

ServeLearnBench evaluates agents in controlled retail, banking, and sales-pitch environments. Benchmark performance does not establish suitability for real-world deployment, particularly for financial decisions or personalized persuasion. Although the Pitch rubric penalizes unsupported claims, it does not comprehensively assess misleading or manipulative language. Matching a customer's preferences should not justify inventing product attributes or concealing material risks. Real-world applications require additional safeguards for privacy, fairness, and truthful communication beyond the behaviors measured here.

REPRODUCIBILITY STATEMENT

Appendix B documents model configurations, harness implementations, evaluation protocols, and aggregation procedures. Appendix C specifies task interfaces, environment changes, and scoring rules, while Appendix D details the analyses. Reported results are tied to a fixed experimental snapshot, with per-setting scores provided for independent checks. Exact replication may be affected by stochastic model outputs and changes to hosted model services. Analysis scripts and per-task data will be released with the benchmark.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.19457.

Anthropic. Introducing claude opus 5, 2026. URL https://www.anthropic.com/news/claude-opus-5.

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

Ching-An Cheng, Andrey Kolobov, Dipendra Misra, Allen Nie, and Adith Swaminathan. Llf-bench: Benchmark for interactive learning from language feedback. *arXiv preprint arXiv:2312.06853*, 2023. URL https://arxiv.org/abs/2312.06853.

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. URL https://arxiv.org/abs/2504.19413.

Jihye Choi, Jinsung Yoon, Long T. Le, Somesh Jha, and Tomas Pfister. Policybank: Evolving policy understanding for llm agents. *arXiv preprint arXiv:2604.15505*, 2026. URL https://arxiv.org/abs/2604.15505.

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026. URL https://arxiv.org/abs/2609.19969.

Run-Ze Fan, Zihao Zhang, Simin Ma, Yebowen Hu, Shouju Wang, Kaiqiang Song, Fei Liu, Hamed Zamani, and Xiaoyang Wang. An empirical study of harness design for coding agents. *arXiv preprint arXiv:2609.20804*, 2026a. URL https://arxiv.org/abs/2609.20804.

Xinyi Fan, Miri Liu, Ruozhen Yang, Siru Ouyang, and Jiawei Han. Can agent memory systems track evolving state? *arXiv preprint arXiv:2608.19652*, 2026b. URL https://arxiv.org/abs/2608.19652.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025.

Tianyi Guan, Yiding Wang, Haotong Yang, Siyuan Cao, Shirui Liu, Yi Hu, Jiaqi Li, and Muhan Zhang. Continualskillbench: Can llm agents truly evolve their capabilities? *arXiv preprint arXiv:2608.03874*, 2026. URL https://arxiv.org/abs/2608.03874.

Anne Harrington, Nayan Saxena, Michael Murphy, Anastasia Borovykh, Zeyu Yun, Sridhar Kamath, Ara Eindra Kyi, Trevor Darrell, Jitendra Malik, and Yutong Bai. When does continual learning require learning. *arXiv preprint arXiv:2607.07847*, 2026. URL https://arxiv.org/abs/2607.07847.

Kaiyu He, Mian Zhang, Shuo Yan, Peilin Wu, and Zhiyu Zoey Chen. IDEA: Enhancing the rule learning ability of large language model agent through induction, deduction, and abduction. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL https://arxiv.org/abs/2408.10455.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06654v3.

Chengsong Huang, Zifeng Wang, Rujun Han, et al. Envharness: Awakening static worlds for agent learning. *arXiv preprint arXiv:2608.19880*, 2026. URL https://arxiv.org/abs/2608.19880.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. DiscoveryWorld: A virtual environment for developing and evaluating automated scientific discovery agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/13836f251823945316ae067350a5c366-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

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 *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06770.

Borui Kang, Jinrui Gu, Junhan Lv, Wenbin Li, Lei Wang, and Yang Gao. Harness continual learning: Continual adaptation beyond model parameters. *arXiv preprint arXiv:2608.19013*, 2026. URL https://arxiv.org/abs/2608.19013.

Seth Karten, Alex L. Zhang, Kevin Thomas, Sebastian Müller, Elie Bakouch, Daniel Auras, Mika Senghaas, Fares Obeid, Konstantin Dunas, Johannes Hagemann, and Sami Jaghouar. Prime agent: A self-improving rlm harness. *arXiv preprint arXiv:2608.23552*, 2026a. URL https://arxiv.org/abs/2608.23552.

Seth Karten, Joel Zhang, Tersoo Upaa, Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents. *arXiv preprint arXiv:2605.09998*, 2026b. URL https://arxiv.org/abs/2605.09998.

Kimi Team. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653v2*, 2026. URL https://arxiv.org/abs/2607.24653v2.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement. *arXiv preprint arXiv:2607.15524*, 2026. URL https://arxiv.org/abs/2607.15524.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2005.11401.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, et al. Evox: Meta-evolution for automated discovery. *arXiv preprint arXiv:2602.23413*, 2026a. URL https://arxiv.org/abs/2602.23413.

Zewen Liu, Zhan Shi, Yisi Sang, et al. Adaptive auto-harness: Sustained self-improvement for agentic system deployment on open-ended task streams. *arXiv preprint arXiv:2606.01770*, 2026b. URL https://arxiv.org/abs/2606.01770.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2402.17753.

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. URL https://arxiv.org/abs/2506.13131.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition, 2026. URL https://openai.com/index/gpt-5-6/.

Siru Ouyang, Jun Yan, I-Hung Hsu, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25140.

Wenbo Pan, Shujie Liu, Chin-Yew Lin, Jingying Zeng, Xianfeng Tang, Xiangyang Zhou, Yan Lu, and Xiaohua Jia. Evolving agents in the dark: Retrospective harness optimization via self-preference. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026. URL https://arxiv.org/abs/2606.05922. Accepted.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Yiheng Shu, Bernal Jiménez Gutiérrez, Saisri Padmaja Jonnalagedda, Yuguang Yao, Huan Sun, and Yu Su. Agentcl: Toward rigorous evaluation of continual learning in language agents. *arXiv preprint arXiv:2606.02461*, 2026. URL https://arxiv.org/abs/2606.02461.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/7107d4d2e837bde2171c6b71b5bde954-Abstract-Conference.html.

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. Streambench: Towards benchmarking continuous improvement of language agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c189915371c4474fe9789be3728113fc-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.10813.

Jundong Xu, Qingchuan Li, Jiaying Wu, Yihuai Lan, Shuyue Stella Li, Huichi Zhou, Bowen Jiang, Lei Wang, Jun Wang, Anh Tuan Luu, Caiming Xiong, Hae Won Park, Bryan Hooi, and Zhiyuan Hu. Evoarena: Tracking memory evolution for robust llm agents in dynamic environments. *arXiv preprint arXiv:2606.13681*, 2026. URL https://arxiv.org/abs/2606.13681.

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*, 2026. URL https://arxiv.org/abs/2605.23904.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.12045.

Z.ai. GLM-5.3 model card, 2026a. URL https://huggingface.co/zai-org/GLM-5.3.

Z.ai. GLM-5.3-Flash model card, 2026b. URL https://huggingface.co/zai-org/GLM-5.3-Flash.

Guibin Zhang, Leo Lu, Fangzhou Xie, et al. Jit-agent: Scaling harness intelligence via just-in-time harness evolution. *arXiv preprint arXiv:2608.25593*, 2026a. URL https://arxiv.org/abs/2608.25593.

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 *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2510.04618.

Xuan Zhang, Wenxuan Zhang, See-Kiong Ng, and Yang Deng. Self-evolving world models for llm agent planning. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026c. URL https://arxiv.org/abs/2606.30639. Accepted.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2308.10144.

Anshun Asher Zheng, Kanishka Misra, David I. Beaver, and Junyi Jessy Li. Hero's Journey: Testing complex rule induction with text games. *arXiv preprint arXiv:2606.02556*, 2026. URL https://arxiv.org/abs/2606.02556.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025a. URL https://arxiv.org/abs/2505.11942.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, 2025b.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo F. R. Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. Skilllearnbench: Benchmarking continual learning methods for agent skill generation on real-world tasks. *arXiv preprint arXiv:2604.20087*, 2026. URL https://arxiv.org/abs/2604.20087.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.13854.

Sibo Zhu, Shicheng Fan, Xinyue Wang, Wenyi Wu, Kun Zhou, and Biwei Huang. Rsiagent: Autonomous exploration for recursive self-improvement in new environments. *arXiv preprint arXiv:2609.15364*, 2026. URL https://arxiv.org/abs/2609.15364.

A Appendix Overview

- **Appendix B: Additional Experimental Setting.** The setting behind Section 4.
  - B.1 Model and Budgets
  - B.2 Learning Harnesses
  - B.3 Evaluation Protocol
  - B.4 Data Processing and Reporting
- **Appendix C: Detailed Benchmark Specification.** Interfaces, tiers, hidden rules, and verification for the nine scenarios of Section 3.
  - C.1 Task and Scenario Interfaces
  - C.2 Tier Semantics and Window Sizes
  - C.3 Retail Hidden Rules and Activation Schedule
  - C.4 Banking Hidden Rules and Activation Schedule
  - C.5 Pitch Latent Preferences and Activation Schedule
  - C.6 Task Success Verification
- **Appendix D: Additional Evaluation.** Methods and extra results behind the analyses of Section 5.
  - D.1 Learning-Speed Analysis
  - D.2 Learning a Policy versus Dropping It
  - D.3 Exploration by Domain
  - D.4 Cost per Phase and Learning Cost Breakdown
  - D.5 Training-Based Methods
  - D.6 Interpreting Fully Specified Reference Performance

B Additional Experimental Setting

B.1 Model and Budgets

All reported cells use the unified high-effort regime. Within each model block, Blind, Oracle, and the learning harnesses use the same model so that differences among harnesses reflect access to environment knowledge and serving history rather than base-model capability. The provider endpoints expose different reasoning histories: Terra returns no chain of thought and Opus returns a summary, while the Fireworks trajectories retain the full returned reasoning. Harnesses that learn from trajectories therefore receive different amounts of reasoning information across these providers. Pitch uses a fixed GLM-5.3 Flash judge with high reasoning effort to produce structured rubric labels, which deterministic code maps to the continuous rating specified in Appendix C.6. The same judge scores all models, including GLM-5.3 Flash itself. The shared loop supplies tools through the provider's native tool-calling interface, returns observations as tool-result messages, and exposes `finish` as a tool. It permits one tool call per turn; a response containing multiple calls is rejected without executing any of them. Each scored item has one attempt. The shared loop imposes a budget of 50 environment tool calls with terminal actions exempt, a 100-step runner guard, and a 256k output-token cap. Assistant messages, including reasoning, are replayed as returned by the provider. Prime retains its own action interface, with persistent harness state and a fresh session for each ticket. Test-side tasks remain excluded from all learner updates, validation, and candidate selection. Infrastructure failures trigger task reruns rather than scored failures; three consecutive infrastructure failures stop the lane. Model context overflow is scored on the task outcome as it stands.

B.2 Learning Harnesses

The five learning harnesses are complete systems of very different size and design: Prime is an agent runtime with its own sandbox and code interpreter, SkillOpt is a prompt optimizer built for batched offline training, Mem0 and CH are memory layers with their own extraction or refinement models, and RAG is a retrieval index with no learning-side model calls. Aligning them to one update cadence, one state budget or one learning-side compute budget would change the methods themselves, so we did not attempt it. The comparison uses the same task streams and held-out test sets, the same model and reasoning effort within each model block, the same per-task budget (50 tool calls, 100 turns), and score-only outcome feedback. Hidden-policy descriptions and corrected actions are withheld from learning harnesses, and held-out test tasks and feedback are excluded from learning (Appendix B.3). Update schedules, replay access, state capacity, and learning-side computation follow each method's design and the adaptations described below. Comparisons therefore characterize complete systems under their disclosed operating conditions; they do not isolate harness effects under identical information and resource budgets.

Acting agent. RAG, Mem0, SkillOpt and CH share one ReAct acting loop with one tool call per turn: RAG injects the retrieved cases and SkillOpt its skill document into the system prompt, Mem0 adds two memory tools, and CH adds the Continual Harness tools for memories, skills, subagents and code, all of which count against the task's call budget. Prime keeps its own runtime: a code agent that acts through a persistent interpreter and a shell inside a sandbox, may issue several calls per turn, and reaches the case system through a scripted client. Its base agent is therefore stronger than the ReAct loop, and any CH-versus-Prime difference includes the runtime as well as the state layer.

When and how each harness learns. Prime opens a fresh session per ticket; after grading, the score line enters the same session and the agent takes a post-score turn in which it may read but not act, and writes its memory, skill and prompt files itself. Mem0 can write memories during the task and takes a post-outcome memory turn after every serving ticket, with the score line in the same conversation; the stock Mem0 has no such turn, and its extraction model runs with Mem0's own defaults. CH's acting agent never sees its score; the score reaches only the Refiner, which runs on the original's cadence (every 25 model steps for the first 200 steps, then every 100), evaluated between tickets, and revises the prompt, memories, skills and subagents in four model calls per pass. RAG adds a block of twelve served cases, each with its score line, to a BM25 index once the block ends and retrieves the eight most similar cases for each new task. SkillOpt treats every block of twelve serving tickets as one optimization step: the first six are the training batch and the last six the selection set on which a candidate skill is replayed for selection before it can replace the current skill. The adapter maps each environment window to one optimization epoch: it runs a slow update after that window's serving tasks, samples replay tasks from the same window, and schedules meta-skill updates on the same epoch cycle (skipping the first epoch). This gives the optimizer access to window segmentation through scheduling and replay selection, even though model prompts contain no explicit window labels or hidden-policy descriptions. To adapt the original fixed-validation setup to a stream, the selection set slides with the served tasks and the skill starts empty; each incoming task receives one scored rollout, and the strict acceptance gate accepts only about one step in nine. Its validation replays re-run tasks that have already been served, about 0.85 additional task executions per served ticket, which is possible in a simulator but not in a deployment where each ticket is a live interaction, as in Pitch.

State and capacity. Prime injects a bounded summary of its harness files into the prompt and can search all of its past transcripts; CH shows overviews of its stores and bounds the Refiner's window; SkillOpt keeps one skill document with a per-step edit budget. These capacities are not equal and are part of each method.

Accounting. Learning-side calls (Mem0's memory turn and extraction model, CH's Refiner passes, SkillOpt's optimizer and replays) are booked as maintenance. Because Prime's relay ledger books its post-score turn together with the ticket turns, the learning share reported for Prime in Figure 13 is recovered from its per-ticket session records: every assistant message carries its own usage, and the runner's outcome message (`Ticket outcome. Score: n/100`) marks where the ticket work ends and the learning turn begins; the ticket-side totals so obtained equal the per-task usage in the result files exactly, and the ledger's remainder is counted as serving. SkillOpt's validation replays use serving-side tasks only and never contribute to test scores.

B.3 Evaluation Protocol

Evaluator and learner protocol. The evaluator has access to the full benchmark state, including the latent environment policy, task type, protocol role, and window boundaries. Learning harnesses receive task-visible information and accumulate serving interactions and outcome feedback; hidden-policy descriptions and corrections are withheld. Harness-specific use of window structure for update scheduling and replay is described in Appendix B.2. Learners may update their persistent state from serving experience, whereas test tasks reveal no feedback and do not affect subsequent adaptation. The objective is to improve performance on Hidden-Dependent tasks while preserving performance on Fully Specified tasks, thereby testing whether agents can learn hidden environment knowledge without disrupting behavior that is already correct.

Model-visible inputs and optimizer access. The acting and reflection components may receive: the visible task input, the agent's own tool trajectory or generated response, scalar outcome feedback, monotonic stream timestamp, and the learner's own persistent state. They may not receive: slice labels, latent-state values, rule-family labels, oracle bulletins, corrected actions, evaluator explanations, or raw world-event time. Window segmentation may be used by a harness's update scheduler or replay sampler, as specified in Appendix B.2; the active policy values and the content of policy changes remain hidden. The benchmark's release gate audits model-visible prompts and metadata against these content restrictions.

Serving-side and test-side isolation. All tasks in a domain–tier instance belong to one evaluator-ordered EESD partitioned into environment windows. Each window contains a serving side followed by a test side under the same latent environment state, and both sides cover every behavioral component exercised by the tier with disjoint concrete instances. Each serving-side task is scored once with the candidate active when it

arrives; its observed outcome may be used by the update rule. Subsequent replay scores, where supported, are used for harness optimization and are not included in reported serving or test scores. After the serving tasks and any scheduled harness updates, the evaluator freezes the candidate state and runs every test-side probe independently from that state before the next environment state takes effect. Probe-local state is discarded, and test-side trajectories and scores are excluded from every learner update, reflection minibatch, validation reservoir, and candidate-selection decision. Only serving-side tasks receive learner-visible monotonic timestamps; the evaluator's test block does not create a visible timestamp gap.

B.4 Data Processing and Reporting

Table 2 is generated from the benchmark's completed-cell results, snapshot dated September 25, 2026; the values are pinned and the table is rendered and checked by a script rather than typed. Only current completed cells enter the table; archived superseded runs are excluded. Every performance value in Table 2 is a held-out test aggregate.

Aggregation convention. Cross-setting reward summaries first compute a mean task reward within each model–harness–setting cell and then average cells with equal weight. The tasks included depend on the reported phase and slice: Retail and Banking rewards are binary and scaled to 0–100, whereas Pitch uses the 0–100 rubric score. Table 2 uses held-out test tasks: Avg. H averages the nine Hidden setting scores and Avg. F the six Retail/Banking Fully Specified scores. Figure 3 averages these test scores over the relevant models or harnesses, and Figure 1 directly uses the table's Avg. H. Figure 4 pools serving and test tasks within each setting before averaging across settings; Figure 12 and Table 3 instead compute serving and test rewards separately. These displays share the table's cross-setting weighting, but only their Hidden test summaries use the same task subset as Avg. H. The reward axes in the exploration and knowledge-acquisition analyses (Section 5.3) use only the six Retail/Banking Hidden test settings, excluding Pitch; per-domain comparisons use the three settings of that domain. Their learning-dynamics metrics are computed from serving trajectories using the policy- and ticket-level definitions in Section 5, not by the cross-setting reward formula. Item-weighted pooling would instead let the settings with the most test items dominate: Retail holds 1,686 of the 3,210 test items (53%) and has the lowest Hidden scores, so item-weighted Hidden averages are several points lower for every harness (for example 30.7 rather than 38.3 for Mem0 and 60.2 rather than 61.7 for Prime over the four open-weight models). We therefore use equal setting weights for cross-setting reward summaries. Averaging Retail and Banking accuracies with the Pitch rubric score treats each setting's score as a 0–100 reward and is a choice of the benchmark; the per-setting values remain available in Table 2. Costs and token counts are not averaged this way: they are totals over all tasks of the lanes concerned, divided by the number of tasks, so a per-task cost is the mean over tasks. Table 2 computes its averages from the published one-decimal setting scores and rounds to one decimal; an incomplete set of constituent settings yields a dash. The resource figures and Table 3 are computed from the archived result files of the completed lanes (bundle of September 25, 2026) with the same convention applied to the raw per-item rewards, which can move a value by 0.1 relative to the table. Table 3 lists the per-task costs and reward values behind Figure 12, aggregated by model and by learning harness over the same lanes per column (45 per model column, 36 per harness column).

Each cell summarizes one run; no standard error or confidence interval is implied. Pitch L1 has only 6–8 effective attribute combinations per 24-item environment window, so the item count should not be interpreted as independent statistical replication.

Table 3: **Average cost per task and reward, by model and by learning harness.** Serving: API cost of one serving task including the harness's learning work (Refiner passes, memory model, optimizer, Prime's post-score turn), and reward on the serving stream. Test: API cost of one held-out test task under the frozen learned state, and test reward. Cost is total dollars over total tasks; reward is the mean over the column's model–harness–setting cells of the cell's Hidden (Adapt) or all-case (Total) reward, so every setting has equal weight as in Table 2 (binary in Retail and Banking, the rubric rating in Pitch). Every model column aggregates 45 complete lanes (five harnesses $\times$ nine suites) and every harness column 36 (four models $\times$ nine suites); Blind and Oracle are excluded. Fireworks serverless prices of 2026-09-06.

	Serving (incl. learning)			Test (frozen state)		
	$/task	Adapt	Total	$/task	Adapt	Total
By model						
GLM-5.3 Flash	0.0117	40.4	59.5	0.0086	44.6	58.2
DeepSeek V4.1 Flash	0.0228	41.8	61.0	0.0170	44.9	60.1
GLM-5.3	0.1732	48.5	64.5	0.1153	52.5	64.3
Kimi K3	0.1880	48.9	63.9	0.1298	50.7	62.1
By learning harness						
SkillOpt	0.0388	33.2	56.7	0.0180	39.8	55.9
RAG	0.0658	43.2	60.5	0.0832	43.4	57.4
Mem0	0.0691	36.3	59.1	0.0213	38.3	56.0
CH	0.0891	54.6	66.2	0.0736	57.7	66.7
Prime	0.2320	57.2	68.6	0.1423	61.7	69.9

C Detailed Benchmark Specification

This appendix specifies the learner-facing task interfaces, tier and window design, evaluator-side hidden-state schedules, and task-success verification. The latent-state component descriptions below are evaluator metadata and are not provided to learning harnesses. Harness-specific use of window structure is described in Appendix B.2.

C.1 Task and Scenario Interfaces

Table 4 separates the surface task from the latent object that changes. Retail and banking require tool-mediated decisions whose correctness is verified from the final environment state; pitch requires a constrained natural-language response scored by a fixed rubric.

Table 4: Task interfaces and latent environment components.

Domain	Task families	Required output	Latent state components
Retail	Cancel a pending order; exchange or return delivered items; modify a pending order; resolve multi-item and ranked-fallback requests.	API actions that produce the correct database state, or a refusal with the correct reason code.	Categorical eligibility scopes over payment form, product category, destination, refund method, order composition, and exception status.
Banking	Authorize a pending card transaction; review a credit-limit increase; screen an outbound transfer; resolve ranked alternatives and payment batches in L2/L3.	Approve, deny, or escalate with the correct reason; execute the first admissible alternative or screen each batch line when applicable.	Numeric thresholds and scoped restrictions over merchant corridor, category limits, daily totals, account standing, requested limits, and payee age.
Pitch	Write a 40–80 word sales pitch from a factual product sheet.	A pitch that foregrounds positively valued attributes, suppresses negatively valued attributes, and introduces no unsupported claims.	Valence over product attribute–value pairs; the active preference table is hidden and only the rubric score is returned.

C.2 Tier Semantics and Window Sizes

The tier labels describe temporal and compositional demands rather than a monotonic increase in surface-form complexity (Table 5). Within a tier, serving and test cover the same active behavioral components using disjoint instances.

Table 5: Cross-domain interpretation of the three difficulty tiers.

Tier	Temporal demand	Within-task demand	Primary diagnostic
L1	Discover a latent state or newly introduced rule from outcome feedback; Pitch presents four profiles without re-entry.	Mostly single-rule decisions; Pitch uses five visible attribute dimensions.	Initial acquisition relative to the Blind–Oracle gap.
L2	Revise beliefs after tightening, relaxation, scope migration, or limited profile re-entry.	Scoped rules, correlated changes, ranked fallbacks, multi-line cases, or ten-dimensional preference selection.	Hidden-state adaptation and impact on fully specified cases.
L3	Track oscillation, rule removal, and re-entry of previously observed states.	Multiple simultaneously relevant components and staggered reversals; retail also includes permissive departures from the document.	Returned fully specified behavior, stale beliefs, and interference.

Table 6 gives the exact number of serving and test tasks in every released window. Each row is one domain–tier scenario, #W is its number of released windows, and an entry S/T denotes S serving tasks followed by T test probes under the same environment state.

Table 6: Sample counts by scenario, difficulty, and environment window. Each entry is serving/test; the final row reports 53 windows and 4,508/3,210 tasks.

Domain	Tier	#W	W0	W1	W2	W3	W4	W5	W6	W7	Total S/T
Retail	L1	6	79/44	140/64	140/64	120/64	120/64	121/84	n/a	n/a	720/384
	L2	7	99/81	128/104	135/115	131/128	134/130	90/111	93/96	n/a	810/765
	L3	8	78/60	75/65	84/68	78/67	77/76	74/72	80/64	78/65	624/537
Banking	L1	4	84/41	84/47	84/42	84/48	n/a	n/a	n/a	n/a	336/178
	L2	5	100/74	100/77	100/76	100/80	100/80	n/a	n/a	n/a	500/387
	L3	6	100/72	100/90	100/89	100/97	100/105	100/98	n/a	n/a	600/551
Pitch	L1	4	54/24	54/24	54/24	54/24	n/a	n/a	n/a	n/a	216/96
	L2	6	54/24	54/24	54/24	54/24	54/24	54/24	n/a	n/a	324/144
	L3	7	54/24	54/24	54/24	54/24	54/24	54/24	54/24	n/a	378/168
All	9 tiers	53	53 environment windows across all scenarios								4508/3210

C.3 Retail Hidden Rules and Activation Schedule

The L1/L2/L3 scenarios use 7/14/12 categorical hidden rules over payment form, product category, destination, refund method, order composition, and exceptions. We first define each atomic retail rule in Table 7; Table 8 then reports only whether that rule is active in each window. A block rule overrides a documented permission, whereas a permit rule introduces an exception to a documented restriction.

Table 7: Dictionary of atomic retail hidden rules.

Tier	ID	Meaning when active
L1	H1	Block modification of a pending order paid with multiple payment methods.
L1	D1	Block cancellation of a pending order paid with a gift card.
L1	D2	Block exchange of an electronics item.
L1	D3	Block exchange when the replacement destination is Florida.
L1	D4	Block refund to the original payment method.
L1	D5	Permit return of a clearance item.
L1	D6	Permit a return outside the documented return window.
L2	R1-E	Block exchange of an electronics item.
L2	R1-A	Block exchange of an audio item.
L2	R2-G	Block cancellation of a gift-card-paid order.
L2	R2-S	Block cancellation of a split-paid order.
L2	R3-All	Block every refund to the original card.
L2	R3-Amex	Block refund to the original card only when that card is American Express.
L2	R4-FL	Block exchange when the replacement destination is Florida.

Tier	ID	Meaning when active
L2	R4-TX	Block exchange when the replacement destination is Texas.
L2	R5	Block modification of a split-paid order.
L2	R6-M	Block a partial return from a multi-item order.
L2	R6-ME	Block a partial return only when the multi-item order contains electronics.
L2	R7	Block a partial exchange.
L2	R8	Block exchange to a PO Box.
L2	R9	Permit a gift-card-funded exchange upcharge.
L3	Z1-A	Block exchange of an audio item.
L3	Z1-E	Block exchange of an electronics item.
L3	Z2-G	Block cancellation of a gift-card-paid order.
L3	Z2-S	Block cancellation of a split-paid order.
L3	Z3-O	Permit refund to the original payment method.
L3	Z3-V	Permit refund to the original card only when that card is Visa.
L3	Z4	Permit return of a clearance item.
L3	Z5	Block modification of a split-paid order.
L3	Z6	Block exchange to a PO Box.
L3	Z7-FL	Block exchange when the replacement destination is Florida.
L3	Z7-TX	Block exchange when the replacement destination is Texas.
L3	Z8	Permit a gift-card-funded exchange upcharge.

In the activation matrices, ✓ means active, ✗ means inactive, and n/a means that the tier has no such window.

Table 8: Retail hidden-rule activation by window. A check mark denotes active, a cross denotes inactive, and n/a denotes an unavailable window. Rule meanings are defined in Table 7.

Tier	ID	W0	W1	W2	W3	W4	W5	W6	W7
L1	H1	✓	✓	✓	✓	✓	✓	n/a	n/a
L1	D1	✗	✓	✓	✗	✗	✗	n/a	n/a
L1	D2	✗	✗	✓	✓	✓	✓	n/a	n/a
L1	D3	✗	✗	✗	✗	✓	✓	n/a	n/a
L1	D4	✗	✗	✗	✗	✗	✓	n/a	n/a
L1	D5	✓	✓	✓	✓	✓	✓	n/a	n/a
L1	D6	✗	✗	✗	✓	✓	✓	n/a	n/a
L2	R1-E	✗	✓	✓	✗	✗	✗	✗	n/a
L2	R1-A	✗	✗	✗	✓	✓	✗	✗	n/a
L2	R2-G	✗	✗	✓	✓	✗	✗	✗	n/a
L2	R2-S	✗	✗	✗	✗	✓	✓	✗	n/a
L2	R3-All	✗	✗	✓	✓	✗	✗	✗	n/a
L2	R3-Amex	✗	✗	✗	✗	✓	✓	✓	n/a
L2	R4-FL	✗	✗	✗	✓	✓	✗	✗	n/a
L2	R4-TX	✗	✗	✗	✗	✗	✓	✓	n/a
L2	R5	✓	✓	✓	✓	✓	✓	✓	n/a
L2	R6-M	✗	✓	✓	✓	✗	✗	✗	n/a
L2	R6-ME	✗	✗	✗	✗	✓	✓	✓	n/a
L2	R7	✗	✗	✓	✓	✓	✓	✗	n/a
L2	R8	✓	✓	✓	✓	✓	✓	✓	n/a
L2	R9	✓	✓	✓	✓	✓	✓	✓	n/a
L3	Z1-A	✗	✓	✗	✗	✗	✗	✗	✗
L3	Z1-E	✗	✗	✗	✓	✓	✗	✗	✗
L3	Z2-G	✗	✓	✗	✓	✓	✗	✗	✗
L3	Z2-S	✗	✗	✓	✗	✗	✓	✓	✗
L3	Z3-O	✗	✗	✓	✗	✗	✗	✗	✗
L3	Z3-V	✗	✗	✗	✗	✓	✓	✗	✗
L3	Z4	✗	✗	✓	✗	✗	✗	✓	✓
L3	Z5	✓	✓	✓	✓	✓	✓	✓	✓
L3	Z6	✓	✓	✓	✓	✓	✓	✓	✓
L3	Z7-FL	✗	✓	✗	✗	✗	✓	✓	✗
L3	Z7-TX	✗	✗	✗	✓	✗	✗	✗	✗
L3	Z8	✓	✓	✓	✓	✓	✓	✓	✓

C.4 BANKING HIDDEN RULES AND ACTIVATION SCHEDULE

The L1/L2/L3 scenarios use 5/9/11 rule-value states across 4/6/6 policy dimensions; verification checks the submitted verdict and terminal environment state. Table 9 turns each threshold or categorical scope into an atomic rule. Table 10 then marks the windows in which that exact value is used.

Table 9: Dictionary of atomic banking hidden rules.

Tier	ID	Meaning when active
L1	S1	Basic verification is insufficient for a gift-card purchase.
L1	S2-650	Set the electronics instant-approval ceiling to $650.
L1	D1-J	Apply the overseas-merchant restriction to jewelry.
L1	D2-3	Use the three-payee restricted-payee roster.
L1	D2-5	Use the five-payee restricted-payee roster.
L2	S1-2000	Set the same-day transaction-total cap to $2,000.
L2	S2-6500	Set the review threshold for recently added payees to $6,500.
L2	D1-650	Set the electronics instant-approval ceiling to $650.
L2	D1-460	Set the electronics instant-approval ceiling to $460.
L2	D2-J	Apply the overseas-merchant restriction to jewelry.
L2	D2-E	Apply the overseas-merchant restriction to electronics.
L2	D3-900	Set the travel instant-approval ceiling to $900.
L2	D3-1150	Set the travel instant-approval ceiling to $1,150.
L2	D4-3800	Permit credit-limit increases up to $3,800.
L3	P1-650	Set the electronics instant-approval ceiling to $650.
L3	P1-460	Set the electronics instant-approval ceiling to $460.
L3	P2-900	Set the travel instant-approval ceiling to $900.
L3	P2-1150	Set the travel instant-approval ceiling to $1,150.
L3	P3-J	Apply the overseas-merchant restriction to jewelry.
L3	P3-E	Apply the overseas-merchant restriction to electronics.
L3	P4-2000	Set the same-day transaction-total cap to $2,000.
L3	P4-1700	Set the same-day transaction-total cap to $1,700.
L3	P5-3800	Permit credit-limit increases up to $3,800.
L3	P6-6500	Set the review threshold for recently added payees to $6,500.
L3	P6-7200	Set the review threshold for recently added payees to $7,200.

Table 10: Banking hidden-rule activation by window. A check mark denotes active, a cross denotes inactive, and n/a denotes an unavailable window. Rule meanings are defined in Table 9.

Tier	ID	W0	W1	W2	W3	W4	W5
L1	S1	✓	✓	✓	✓	n/a	n/a
L1	S2-650	✓	✓	✓	✓	n/a	n/a
L1	D1-J	✗	✓	✓	✓	n/a	n/a
L1	D2-3	✓	✓	✓	✗	n/a	n/a
L1	D2-5	✗	✗	✗	✓	n/a	n/a
L2	S1-2000	✓	✓	✓	✓	✓	n/a
L2	S2-6500	✓	✓	✓	✓	✓	n/a
L2	D1-650	✓	✗	✗	✗	✗	n/a
L2	D1-460	✗	✓	✓	✓	✓	n/a
L2	D2-J	✓	✓	✗	✗	✗	n/a
L2	D2-E	✗	✗	✓	✓	✓	n/a
L2	D3-900	✓	✓	✓	✗	✗	n/a
L2	D3-1150	✗	✗	✗	✓	✓	n/a
L2	D4-3800	✗	✗	✗	✗	✓	n/a
L3	P1-650	✓	✗	✗	✓	✓	✓
L3	P1-460	✗	✓	✓	✗	✗	✗
L3	P2-900	✓	✓	✓	✗	✗	✓
L3	P2-1150	✗	✗	✗	✓	✓	✗
L3	P3-J	✓	✓	✗	✗	✓	✗
L3	P3-E	✗	✗	✓	✓	✗	✗
L3	P4-2000	✓	✓	✓	✓	✗	✗
L3	P4-1700	✗	✗	✗	✗	✓	✓
L3	P5-3800	✗	✓	✓	✓	✗	✗
L3	P6-6500	✓	✓	✗	✗	✗	✗
L3	P6-7200	✗	✗	✓	✓	✓	✓

C.5 Pitch Latent Preferences and Activation Schedule

Each Pitch window activates exactly one complete customer preference profile. Table 11 defines the six profiles used across the benchmark, and Table 12 reports their use by tier and window. Every window contains 54 serving and 24 test questions (Table 6). L1 uses four non-repeating profiles over four windows and five visible attributes; L2 uses four profiles over six windows, revisits two profiles, and expands to ten attributes; L3 uses four profiles over seven windows with repeated returns and ten attributes.

Table 11: Dictionary of latent pitch preferences.

ID	Meaning when active
P-Val	Use the value-for-money buyer profile.
P-Id	Use the identity-expressive buyer profile.
P-AM	Use the anti-marketing rationalist profile.
P-RA	Use the risk-averse buyer profile.
P-Eco	Use the eco-conscious buyer profile.
P-Conf	Use the conformist buyer profile.

Table 12: Pitch preference activation by window. A check mark denotes active, a cross denotes inactive, and n/a denotes an unavailable window. Preference meanings are defined in Table 11.

Tier	ID	W0	W1	W2	W3	W4	W5	W6
L1	P-Val	✓	✗	✗	✗	n/a	n/a	n/a
L1	P-Id	✗	✓	✗	✗	n/a	n/a	n/a
L1	P-AM	✗	✗	✓	✗	n/a	n/a	n/a
L1	P-RA	✗	✗	✗	✓	n/a	n/a	n/a
L2	P-RA	✓	✗	✗	✗	✗	✓	n/a
L2	P-Val	✗	✓	✗	✗	✓	✗	n/a
L2	P-Eco	✗	✗	✓	✗	✗	✗	n/a
L2	P-Id	✗	✗	✗	✓	✗	✗	n/a
L3	P-RA	✓	✗	✗	✗	✓	✗	✗
L3	P-Id	✗	✓	✗	✓	✗	✗	✓
L3	P-AM	✗	✗	✓	✗	✗	✗	✗
L3	P-Conf	✗	✗	✗	✗	✗	✓	✗

Together, the dictionaries and binary matrices separate profile meaning from temporal activation, making discovery, revision, and re-entry directly visible.

C.6 Task Success Verification

Retail and Banking: deterministic verification. Retail and Banking share a rule-based, outcome-only verifier with no LLM judgment. For task n, let D_n^{agent} be the terminal database produced by the agent. The evaluator constructs the reference terminal database $D_n^{\star}$ by reloading a clean database, applying the same task-specific view, and replaying the ground-truth tool actions. After canonicalizing dictionaries and sets as order-independent and lists as order-sensitive, it computes

$$r_n^{\text{state}} = \mathbb{I}\big[H(D_n^{\text{agent}}) = H(D_n^{\star})\big], \tag{2}$$

where $H(\cdot)$ is a SHA-256 hash of the canonicalized database and $\mathbb{I}[\cdot]$ is the indicator function. The verifier therefore scores the terminal outcome rather than requiring the agent to reproduce a particular sequence of intermediate actions.

The answer check r_n^{answer} requires exactly one structured `finish` action. For state-changing tasks, the finish declaration must be consistent with carrying out the request; for refusal tasks, it must declare a refusal and provide one of the accepted reason codes; and for value tasks, it must match the expected typed value. Integer answers must contain exactly one integral value, floating-point answers must contain exactly one value within 0.01 of the reference, normalized strings must match exactly after lowercasing and removing non-alphanumeric characters, and sets are compared after element-wise normalization. The binary task reward is

$$R_n^{\text{tool}} = r_n^{\text{state}} r_n^{\text{answer}} \in \{0, 1\}. \tag{3}$$

Thus a correct final answer cannot compensate for an incorrect database transition, and a correct database state cannot compensate for a missing or inconsistent finish declaration.

Pitch: structured LLM judgment. Pitch has no mutable database, so each submitted pitch is evaluated by a fixed GLM-5.3 Flash judge with high reasoning effort and provider-default sampling. For every attribute on the product sheet, the judge returns one structured verdict: M if the pitch mentions or paraphrases the true attribute consistently, C if it directly contradicts or reverses the attribute, and N if it does not address the attribute. The judge separately lists concrete factual claims unsupported by any product-sheet attribute. Its output must be a complete JSON object covering every attribute; malformed or incomplete outputs fail closed and are retried rather than being interpreted as favorable judgments. The LLM produces only this checklist, while deterministic code maps the active customer profile and checklist to the final numerical rating.

Let $\mathcal{A}_n$ be the product attributes for task n, $c_i \in \{\text{strong}, \text{pos}, \text{neu}, \text{neg}, \text{red}\}$ the preference class of attribute i under the active profile, and $v_i \in \{\text{M}, \text{C}, \text{N}\}$ the judge verdict. Strongly positive and positive attributes receive weights $w(\text{strong}) = 2$ and $w(\text{pos}) = 1$, with all other classes receiving zero positive-coverage weight. The weighted positive coverage is

$$C_n = \frac{\sum_{i \in \mathcal{A}_n} w(c_i)\, \mathbb{I}[v_i = \text{M}]}{\sum_{i \in \mathcal{A}_n} w(c_i)}, \tag{4}$$

with $C_n = 0$ if the denominator is zero. Let N_{neg} and N_{red} count negative and red-line attributes, respectively, whose verdict is either M or C; let N_{con} count all C verdicts; and let N_{fab} be the number of distinct unsupported claims, capped at five. The pre-length score is

$$S_n^{\text{raw}} = 100 C_n\, (0.5)^{N_{\text{neg}}} (0.3)^{N_{\text{red}}} (0.5)^{N_{\text{con}}} (0.5)^{N_{\text{fab}}}. \tag{5}$$

Although the requested pitch length is 40–80 words, the verifier allows a grace margin through 90 words. For word count L_n, the final rating and normalized reward are

$$S_n = \begin{cases} S_n^{\text{raw}}, & L_n \le 90, \\ \min(S_n^{\text{raw}}, 25), & L_n > 90, \end{cases} \qquad R_n^{\text{pitch}} = \frac{S_n}{100}. \tag{6}$$

The experiments report the continuous rating S_n; the implementation additionally records $S_n \ge 75$ as the binary task-success flag.

D Additional Evaluation

This appendix details policy-level learning speed, the behavior of agents when a previously learned policy is withdrawn, answer diversity per domain, the breakdown of serving-stream costs, preliminary attempts at training-based learning, and interpretation of Fully Specified reference performance.

D.1 Learning-Speed Analysis

Policies. A policy is one rule of a Banking or Retail tier in one state, from the window in which that state takes effect (or window W0 for a rule present from the start) to the window before the rule next changes. Intervals whose state agrees with the manual are skipped for Retail, since no Hidden ticket can arise from them. Across the three tiers this yields 31 Banking and 37 Retail policies: 23 present from the start, 23 tightened, 9 relaxed, 11 re-scoped, one newly introduced, and one removed.

Attributing tickets to rules. A serving ticket belongs to a rule when that rule decides its correct outcome. For Banking we use the evaluator-side, state-parameterised decision engine of the benchmark's learning-dynamics tools, which is checked to reproduce every locked verdict before use: a rule decides a ticket when setting that rule alone to another value changes the correct decision. For Retail L1 each rule has exactly one Hidden task family, so the family identifies the rule. For Retail L2 and L3 we apply the same counterfactual with each tier's own ground-truth engine: rule r decides a ticket when replacing r's true state by the state the manual describes, holding the other rules fixed, changes the correct actions or answer. The attribution reproduces the recorded Hidden/Fully Specified label of every one of the 810 Retail L2 and 624 Retail L3 serving tickets. Three Retail L2 Hidden tickets change outcome only when several rules change together and are left out. The 27 Banking L3 serving tickets that the bundle files mark *reopen* are classified by the same engine: 8 are Hidden and 19 Fully Specified. Retail windows are taken from the rule timeline at the ticket's time, because the Retail L3 result rows number serving windows differently.

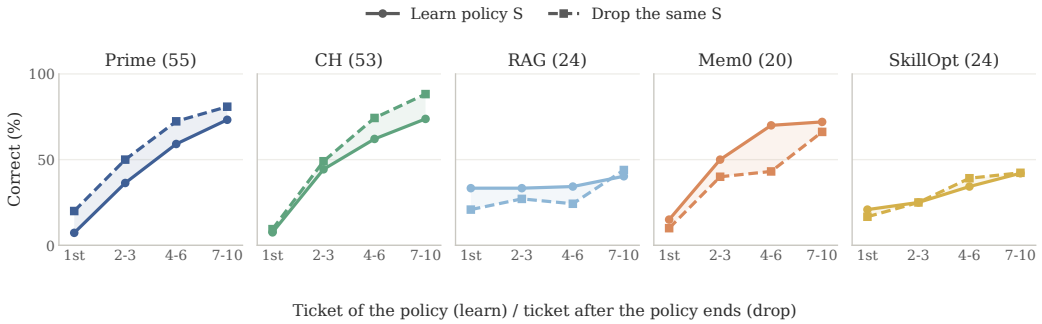


Figure 9: **Learning a policy versus dropping the same policy.** Solid: correctness on the k-th Hidden ticket of policy S. Dashed: correctness on the k-th ticket after S ends that S would answer wrongly. Same model, harness and policy on both sides; pairs in which S was learned; number of pairs in parentheses; four models, Banking and Retail pooled.

Tickets decided by several rules. Almost half of the Hidden serving tickets in Banking L2 and L3 are decided by two rules at once (82 of 172 and 101 of 198), mostly a daily spending cap together with a category limit, or the new-payee and reporting thresholds together; Retail has 9 such tickets. Every attributed ticket counts for each rule that decides it, both as an example the harness has seen (the k axis) and as a measurement (a_k). Counting them as seen but measuring only single-rule tickets gives the same ranking of harnesses, but in Banking the set of policies behind each a_k then changes with k, because some policies contribute single-rule tickets only late in their interval; this produces spurious dips in the curve.

Curves and AULC. a_k pools all attributed tickets of a harness with exactly k earlier tickets of the same policy, over policies, models, and tiers of a domain; panels (a, b) of Figure 5 show positions with at least 20 tickets. AULC_{10} (Section 5.1) is computed per heat-map cell from the same tickets; a position with fewer than three tickets is dropped, and the cell is left empty if fewer than eight of the ten positions remain. The scripts and the per-ticket data will be released with the benchmark.

D.2 Learning a Policy versus Dropping It

A policy that has been learned becomes a liability once the rule returns to what the manual says: the harness must stop applying it. We compare this with learning the same policy, in the same lane. For every rule state S that is followed by a change of the rule (31 states: 10 in Banking L2–L3, 21 in Retail L1–L3; 15 of them return the rule to the manual, the others re-scope it), the *learning* side is the sequence of attributed Hidden serving tickets while S is in force (Appendix D.1), and the *dropping* side is the sequence of serving tickets after S ends whose correct answer is the manual's and would be wrong under S. Dropping tickets are therefore Fully Specified: nothing new has to be learned, only S has to be given up. They are found with the same counterfactual engines: Banking via the discriminating tickets of each rule change, Retail L2–L3 by re-deciding each general ticket with the rule set back to its previous state, and Retail L1 by the one lifted ban (gift-card cancellation, lifted at t=360). Both sides come from one (model, harness, suite, rule, S) pair, so the policy, the model and the harness are identical. We keep the pairs in which the harness had learned S (at least 60% correct on its last five tickets of S); without this condition, a harness that never learned S has nothing to drop and scores highly on the dropping side for that reason alone.

Figure 9 shows that, for the same policy, dropping it is about as fast as learning it. Prime starts higher when dropping (20% on the first ticket, against 7% when learning), CH barely (9% against 8%), and both end 8–14 points higher after seven to ten tickets (81% and 88%, against 73% and 74%); the two curves have the same shape. On the first ticket after the change most harnesses still act on the superseded policy, although a harness that had learned nothing would follow the manual and answer these tickets correctly. Mem0 is the exception: it learns S faster than it drops it (70% against 43% on the fourth to sixth ticket). RAG, whose exemplar store is also append-only, drops policies slowly as well. This explanation is consistent with the curves but not verified at the level of retrieved memories, and the Mem0, RAG and SkillOpt panels rest on 20–24 pairs.

D.3 Exploration by Domain

Figure 10 shows the answer-diversity curves and the reward relation of Figure 7 for Retail and Banking separately. The sequences are the attributed Hidden serving tickets of each policy (Appendix D.1); for Retail we use the restrictive policies, whose correct answer is a refusal, and for Banking the tickets decided by a single rule. A

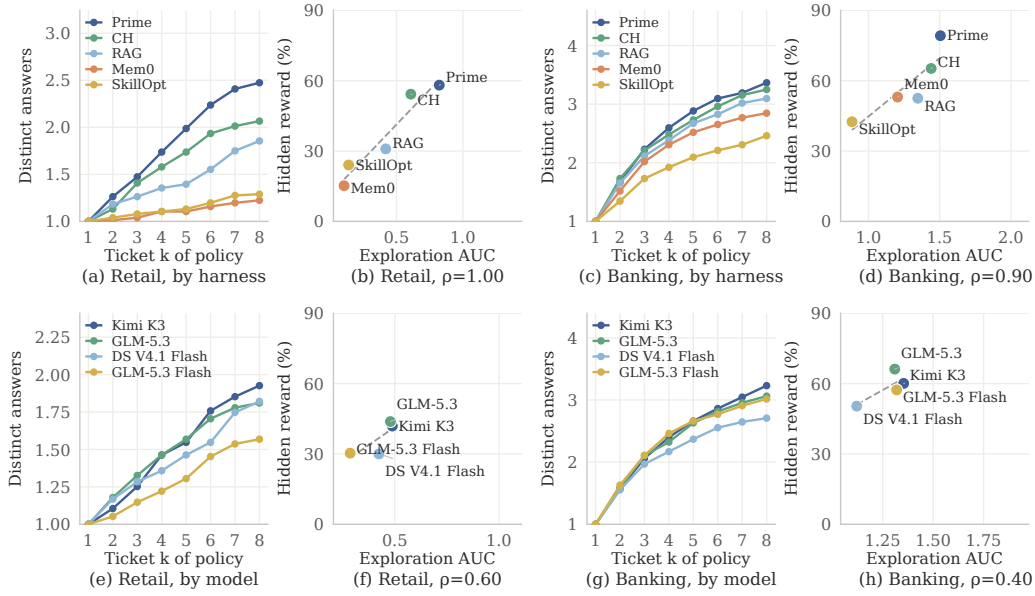


Figure 10: **Answer diversity and reward per domain. (a, c, e, g)** Mean number of distinct answers given on a policy after k of its Hidden serving tickets, by harness (top) and by model (bottom); correctness is ignored. **(b, d, f, h)** Exploration AUC of those curves against the domain's average reward on Hidden cases, with the Spearman ρ and a least-squares fit.

Retail answer is *execute* or *refuse* with one of the accepted refusal codes; the few Retail tickets whose correct answer is a value rather than an action are excluded; a Banking answer is one of approve, deny or escalate together with its reason code, so the answer space is larger in Banking and its curves are higher. Only policies with at least $K = 8$ tickets enter a curve (Section 5.3), so every curve averages the same policies at every k: 20 of 20, 36 of 52 and 20 of 44 policies per harness in Retail L1–L3 and 20 of 20, 12 of 20 and 20 of 24 in Banking L1–L3; the excluded policies are the short ones of Retail L2–L3 and Banking L2–L3, and policies longer than K (median 42 tickets in Retail L1, up to 109) contribute only their first K. The horizon is a trade-off between reading more of the curve and keeping more policies: for every K from 4 to 12 (112 to 44 Retail and 64 to 40 Banking policies per harness) the harness order of the exploration AUC is Prime, CH, RAG, Mem0, SkillOpt, its rank correlation with average Hidden reward is 1.00, and the model-level correlation is 0.80 (1.00 at $K = 6$–7). SkillOpt revises its skill once per block of twelve serving tickets, so within a block its answers can change only through the model's own variation; its curves are therefore a lower bound on the diversity a per-ticket update would show.

The harness ranking of Figure 7(a) holds in both domains except that Mem0 and SkillOpt swap in Retail. In Retail the exploration AUCs are 0.82 (Prime), 0.61 (CH), 0.42 (RAG), 0.14 (SkillOpt) and 0.11 (Mem0), against average Hidden rewards of 58.1, 54.2, 30.9, 24.0 and 15.2 ($\rho = 1.00$); in Banking they are 1.50, 1.44, 1.35, 1.20 (Mem0) and 0.88 (SkillOpt), against 79.2, 65.2, 52.5, 53.0 and 42.5 ($\rho = 0.90$). By model the correlations are 0.60 in Retail and 0.40 in Banking, over four points. Figure 11 repeats the comparison one level finer, with one point per harness, domain and tier (30 cells): the rank correlation is 0.75 over all cells, 0.54 within Retail and 0.69 within Banking, and 0.72, 0.90 and 0.87 within L1, L2 and L3. The finer split adds variance that has nothing to do with the harness, since tiers differ in reward by up to 60 points and the two domains have answer spaces of different size, so the correlation is lower than at the harness level while the direction is the same in every subset; the Retail outliers are RAG, which explores little on L1 yet scores 56% and explores more on L3 yet scores 15%, consistent with its retention problem in Figure 8.

At the finer level of the 20 model–harness cells of a domain, the AUC correlates with the cell's Hidden reward at $\rho = 0.92$ in Retail and 0.57 in Banking; the Banking value is lower because fast-learning cells stop varying their answers early. A variant that counts, over the tickets strictly before a policy's first correct one (policies with at least two such tickets), the number of distinct answers divided by the number of tickets, pooled over a cell's policies, reaches $\rho = 0.94$ and 0.73, and keeps a median within-model correlation of 0.90 and 0.75 across harnesses, so the relation is not explained by stronger models both exploring more and scoring higher. The knowledge-acquisition curves of Figure 8 average Retail and Banking with equal weight; per harness they rest

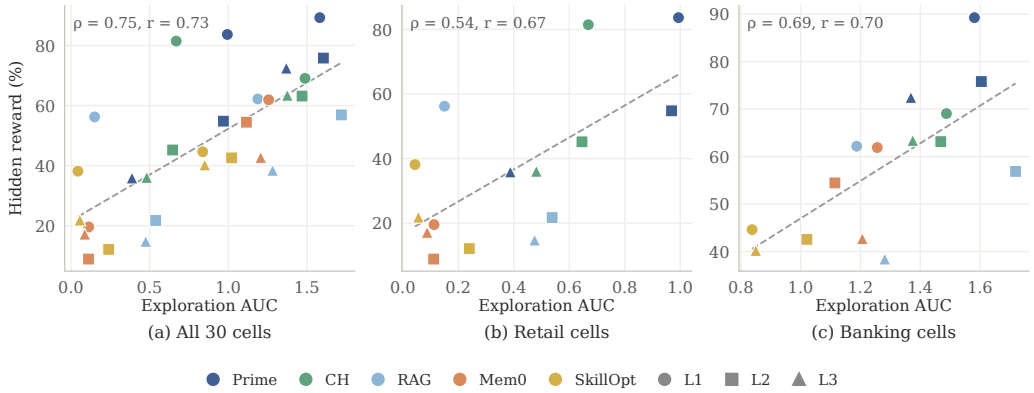


Figure 11: **Exploration and reward per harness, domain and tier.** One point per (harness, domain, tier) cell: exploration AUC (Section 5.3, $K = 8$) over the cell's policies with the four open-weight models pooled, against the cell's reward on Hidden cases from Table 2 averaged over the same models. **(a)** all 30 cells, **(b)** Retail, **(c)** Banking; colour is the harness, shape the tier; ρ and r are the Spearman and Pearson correlations, the dashed line a least-squares fit.

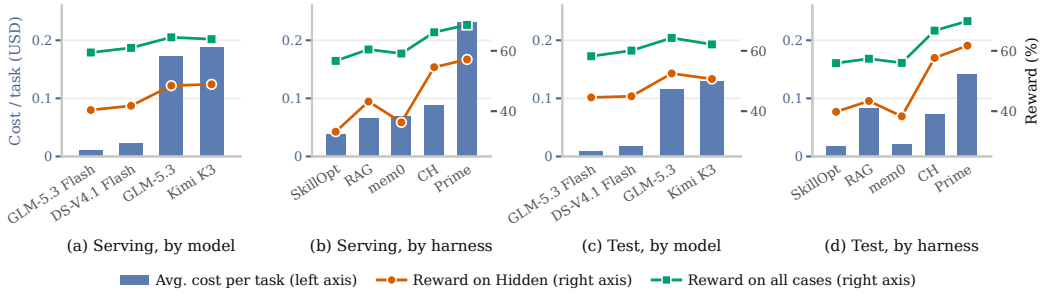


Figure 12: **Average cost per task against reward, by phase.** Bars (left axis) give the API cost of one task; lines (right axis) give reward on Hidden (adapt) cases and on all cases, averaged over the nine suites with equal weight. **(a, b)** Serving stream: cost includes the harness's learning work, reward is measured on the serving cases. **(c, d)** Held-out test under the frozen learned state. Columns are grouped by model (a, c) and by learning harness (b, d), ordered by serving cost; every column aggregates the same lanes (45 per model column, 36 per harness column). Blind and Oracle are excluded.

on 49/41 (Prime), 49/40 (CH), 40/40 (RAG), 6/33 (Mem0) and 13/30 (SkillOpt) Retail/Banking policies, so the Mem0 value is the least certain. The scripts and per-policy sequences will be released with the benchmark.

D.4 Cost per Phase and Learning Cost Breakdown

Figure 12 shows the comparison of Figure 4 with the serving stream and the held-out test kept apart. Serving cost charges everything the harness spends while the stream runs, the model's own turns and the learning work, divided by the number of serving tasks, and is paired with reward on the serving cases; test cost is the inference cost of one held-out task under the frozen learned state, paired with test reward. Reward is averaged equally over the column's model–harness–setting cells, separately for each phase. Hidden test values match the corresponding aggregates of Table 2 up to rounding; serving and all-case rewards use different task subsets. Prime is the most expensive in both phases ($0.232 and $0.142 per task) and reaches the highest test reward on Hidden cases (61.7), while CH reaches 57.7 at 52% of Prime's test cost ($0.074). The two phases can rank a harness differently: SkillOpt and Mem0 spend most of their serving budget on learning, so their test tasks cost less than half of their serving tasks, whereas RAG is the only harness whose test task ($0.083) costs more than its serving task ($0.066), because the retrieved exemplars keep lengthening its prompt. Across models, GLM-5.3 and Kimi K3 buy the highest reward at roughly thirteen to sixteen times the per-task cost of GLM-5.3 Flash, whose test reward on Hidden cases (44.6) trails theirs by 6–8 points. Table 3 lists every value.

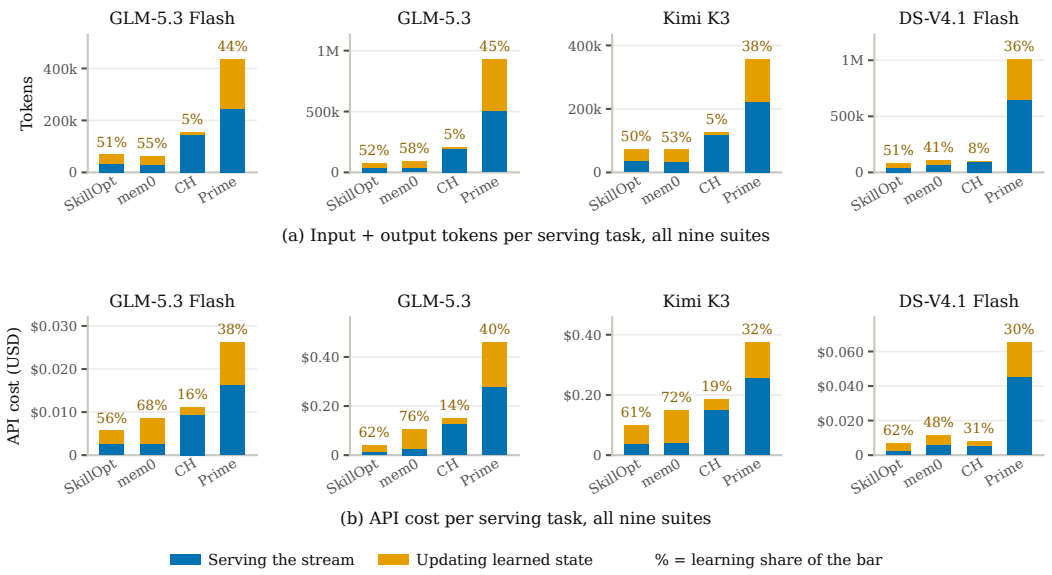


Figure 13: **Adaptation Cost Analysis. (a)** input plus output tokens and **(b)** API cost per serving task, pooled over all nine suites, split between serving the stream and updating the learned state. Percentages give the learning share of each bar. Prime's learning part is its post-score turn, recovered from the session records; its serving part is the rest of the wire ledger's serving bucket. Scored test tasks are excluded. Each panel has its own scale; RAG (no learning-side compute under BM25) is omitted.

The cost of adaptation depends on how a harness divides computation between serving tasks and updating its learned state. Figure 13 splits every learning lane's serving-stream spend into the two parts, per serving task, for all four open-weight models and all nine suites. SkillOpt's learning work is skill optimization and validation, Mem0's is its post-outcome learning turn and memory-extraction model, CH's is its automatic Refiner passes, and Prime's is the post-score turn in which the agent reads the outcome, writes memory and calls `refine`, measured from its per-ticket session records (Appendix B.2). The split is a property of the harness more than of the model: learning takes 56–62% of SkillOpt's API cost and 48–76% of Mem0's on every model, against 14–31% for CH and 30–40% for Prime. Counting tokens, input (cached or not) plus output, the learning share is 42–58% for Mem0, 50–52% for SkillOpt and 36–45% for Prime, but only 5–8% for CH, whose serving turns re-send its prompt, memory and skill overviews on every call and are therefore token-heavy but mostly served from cache. Prime consumes the most tokens per task on every model, because each ticket opens a fresh session that re-reads the harness state over 20–29 model calls (24 on average). Prime is the most expensive harness per task on every model, from $0.026 on GLM-5.3 Flash to $0.46 on GLM-5.3; of the four harnesses in Figure 13, CH is second on three models and Mem0 on DeepSeek V4.1 Flash.

D.5 Training-Based Methods

We explored supervised fine-tuning (SFT) and GRPO-style reinforcement learning (Shao et al., 2024). In preliminary trials, 14B and 32B models struggled to infer hidden environment knowledge and complete tasks successfully, even with strong harnesses, making successful self-generated trajectories difficult to obtain.

This limits the correct demonstrations available for success-filtered SFT and the reward contrast available for GRPO under sparse outcome rewards. When all trajectories in a group receive the same failure reward, their group-relative reward advantages provide no preference among them. Obtaining informative trajectories is therefore a bottleneck for training from serving experience, before the choice of parameter-update algorithm can become effective.

D.6 Interpreting Fully Specified Reference Performance

Fully Specified means that the visible specification is sufficient for the task, not that a model will necessarily trust or follow it. For example, Opus 5 achieves Fully Specified accuracy of 69.1%, 52.7%, and 58.8% under Blind on Banking L1, L2, and L3, respectively, compared with 100.0% under Oracle in all three settings (Table 2). The Banking prompt warns that actual practice may differ from the documents, and we observe that

Opus 5 sometimes speculates about undocumented restrictions without supporting task-specific evidence. Such speculation can lead it to depart from the visible rules and reduce Blind performance even on Fully Specified tasks. Oracle's bulletin may provide additional confirmation and discourage these unsupported assumptions. We therefore interpret changes relative to Blind as differences from the no-learning harness, rather than a pure measure of learning-induced damage.