AGENT-REFLEX: Teaching Language Agents to Act Reflectively

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models have evolved from single-turn chatbots into interactive agents capable of long-horizon reasoning. In real deployments, MCP tool services can return noisy feedback and users may intervene to correct the agent, but existing methods provide limited supervision for reflection learning and failure recovery. To bridge this gap, we propose AGENT-REFLEX, a general training framework for incentivizing proactive reflection in agents. Specifically, AGENT-REFLEX comprises three stages. Firstly, Environment Reflection Skill Discovery (RSD) injects grounded tool-feedback perturbations into environment scaling, samples a capability-diverse LLM pool, mines matched success–failure pairs, and summarizes reusable reflection meta-skills. Building on this skill library, we then apply Reflective Supervised Fine-Tuning (RESFT) that converts each skill into supervision for self-reflection, skill recovery, and user-centric correction, strengthening complementary reflective behaviors. Lastly, we propose On-Policy Reflective Policy Optimization (ORPO) to combine agentic reinforcement learning under noisy interactions with reflection-guided on-policy distillation to further elicit proactive reflection. Experiments on six challenging agent benchmark suites demonstrate the effectiveness of AGENT-REFLEX. Further analyses show that AGENT-REFLEX improves agents' robustness to noise while maintaining stable training dynamics, supporting scalable agent training.

1 INTRODUCTION

Recent advances in large language models (LLMs) have extended their applications from natural language tasks to practical workflows that require long-horizon reasoning (Yao et al., 2023; Patil et al., 2025; Trivedi et al., 2024; Barres et al., 2025). Meeting this demand requires agents not only to master tool interfaces but also to accumulate transferable interaction experience from dynamic environments. Building diverse, stateful environments that faithfully capture real-world interaction logic has therefore become the foundation for scaling agent capability.

Recent efforts to scale agent training environments broadly follow two approaches. One family of approaches, exemplified by Simia and GenEnv, treats LLMs as implicit textual world models that generate feedback from tool specifications and interaction histories (Li et al., 2025; Guo et al., 2025). These approaches substantially reduce environmental engineering costs, but the feedback generated can be hallucinated. Instead, a complementary family explicitly reproduces real interaction logic, synthesizing executable tools and environment databases: it extracts databases from real-world service themes, generates executable tools, and synthesizes tasks with verifiable results (Song et al., 2026; Wang et al., 2026b; Tu et al., 2026; Dong et al., 2026). Thus, these executable environments give agents a basis for learning interaction patterns that transfer across environments.

Although environment scaling improves interaction diversity and out-of-distribution generalization, existing agent-training paradigms still rely on an idealized assumption: agents interact with stable, well-controlled environments under minimal user intervention. This mismatch with real-world deployment reveals two inherent challenges (Figure 1(a,b)):

(1) Lack of noisy feedback. Real-world tool services are unreliable, frequently returning timeouts and service errors. Existing training environments provide mainly stable and noise-free feedback, leaving agents underprepared for failure detection and recovery (Wang et al., 2026a; Gupta, 2026).

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

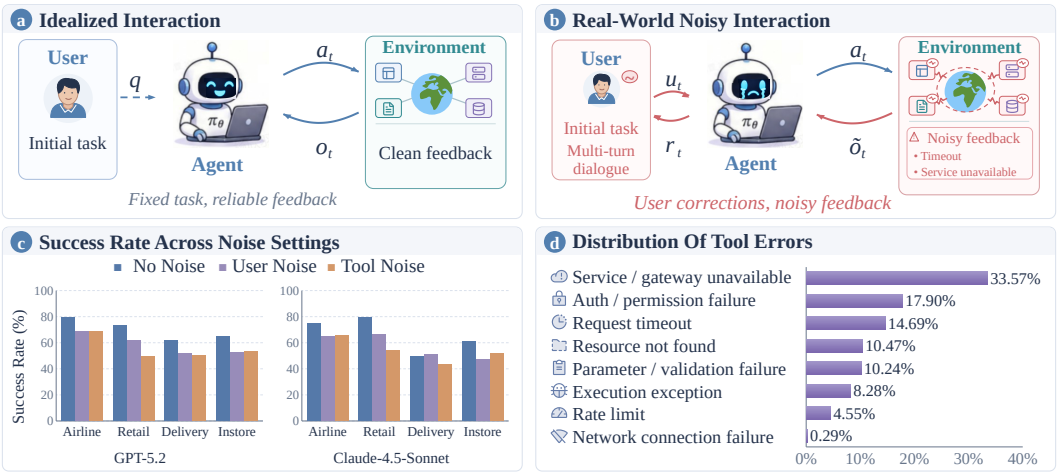


Figure 1: **From idealized interaction to reflective recovery. (a)** Prior environments assume clean tool feedback, whereas **(b)** real ones return noisy observations and user corrections at arbitrary turns. **(c)** Success rate under no, user, and tool noise. **(d)** Share of the eight tool-error families.

(2) Lack of user-guided correction. Real users actively intervene when an agent deviates from their intent, correcting its reasoning and actions. Existing user simulators provide information primarily through standardized templates, with limited coverage of corrective instructions and diverse interaction styles during execution (Barres et al., 2025; Qian et al., 2025; Suhr & Artzi, 2023).

Together, these challenges reveal that agents still lack proactive reflection under noisy external feedback. Ideally, a robust agent should continuously perceive environmental noise, task failures, and user correction intent, then promptly revise its actions. Existing reflection methods explore inference-time reflection, correction-trajectory learning, and noise-augmented training (Shinn et al., 2023; Madaan et al., 2023; Yuan et al., 2025; Bi et al., 2026; Zhang et al., 2026; Chen et al., 2026), but most are trained in stateless or simplified environments, leaving one question open: ***How can we elicit proactive reflective capability in agents from execution failures in real-world environments?***

To address this gap, we propose AGENT-REFLEX, a general training framework for incentivizing proactive reflective thinking in agents. It models reflection as a learnable meta-skill: transferable meta-skills are explicitly mined from execution failures and progressively internalized into the policy through verifiable reflection learning. In detail, AGENT-REFLEX comprises three components:

(1) Environment Reflection Skill Discovery (RSD). We anchor 500 high-usage real-world environment themes and automatically construct database-backed executable environments and tasks, injecting diverse tool-return failures to emulate noisy conditions. From a diverse LLM agent pool, we sample trajectories covering varied task types and response styles, and matched success/failure pairs are then mined into 1,170 reflection meta-skills across 500 environments.

(2) Reflective Supervised Fine-Tuning (ReSFT). Given the skill library, we design three auxiliary and verifiable reflection sampling strategies: *Self-Reflection* (SR), *Skill Recovery* (KR), and *User-Centric Reflection* (UR), which strengthen reflection at different levels. Supervised fine-tuning then internalizes these explicit skills into the agent policy.

(3) On-Policy Reflective Policy Optimization (ORPO). To further elicit proactive reflection, ORPO combines Noisy Agent RL with reflection-guided on-policy distillation. For each task we match its environment's reflection meta-skill and rollout in the noisy environment, optimizing the policy by reinforcement learning from verifiable rewards. The same action tokens are then re-scored under a privileged skill context, and a confidence-weighted OPD objective (Wu et al., 2026) turns this contrast into dense supervision for reflection decisions.

Experiments on six long-horizon tool-use and noisy-interaction benchmarks demonstrate the effectiveness of AGENT-REFLEX. Further analysis shows that it improves noise robustness and keeps training dynamics stable, supporting scalable general-agent training.

Therefore, our contributions are threefold:

- We identify two gaps between idealized synthetic environments and real-world deployment in agent interaction, namely *lack of noisy environment feedback* and *lack of user-guided correction during execution*, highlighting the importance of reflection learning for noisy interactions.
- We introduce RSD, which systematically builds a scalable noisy-environment ecosystem by incorporating real-tool perturbations and mining reflection meta-skills from large-scale success/failure trajectory pairs sampled by an LLM Agent Pool.
- We further propose RESFT and ORPO. RESFT builds multi-level reflection data through three sampling strategies and internalizes it by supervised fine-tuning; ORPO adds reflection-guided on-policy distillation to Noisy Agent RL for token-level supervision of failed executions.

2 RELATED WORK

Agent Reflection Learning. Agent reflection differs in where the corrective signal lives. Inference-time methods leave parameters untouched: Reflexion and Self-Refine append a verbal self-critique, and ExpeL retrieves cross-task insights at test time (Shinn et al., 2023; Madaan et al., 2023; Zhao et al., 2024). Because the critique comes from the policy that failed, a well-formed but wrong tool return is never contradicted. Training-time methods move the signal into a per-trajectory repair: Agent-R, ReflectRL, and LoongReflect learn from that repair (Yuan et al., 2025; Bi et al., 2026; Zhang et al., 2026), and NoisyAgent exposes the policy to perturbed interaction under only a scalar return (Chen et al., 2026). Either way, nothing accumulates into an environment-level statement a later episode can reuse. AGENT-REFLEX mines cross-model success/failure contrasts under the same injected noise into environment-indexed meta-skills and internalizes them through two-stage training.

Scalable Environment Synthesis for Agent Training. Diverse, stateful environments underpin tool use and long-horizon decision making. One line of work simulates interaction feedback with language models, treating tool specifications and dialogue history as an implicit world model (Li et al., 2025; Guo et al., 2025). A second makes the environment itself executable: EnvScaler and ScaleEnv synthesize tools with verifiable tasks (Song et al., 2026; Tu et al., 2026), Agent-World scales database-backed services over hundreds of real-world themes and verifies outcomes through state transitions (Dong et al., 2026; Wang et al., 2026b), and EnvACE internalizes environment dynamics through world rehearsal (Xu et al., 2026). All of them scale the *supply* of solvable environments, whereas AGENT-REFLEX keeps the database-backed paradigm but turns the failures inside those environments into supervision, mining diagnosis and recovery rules from cross-model success/failure contrasts under perturbations derived from real tool errors. Appendix A gives an extended discussion of reflection learning, environment synthesis, and agentic RL.

3 METHODOLOGY: AGENT-REFLEX

Overview. Figure 2 summarizes the pipeline. To convert environment failures into a reflection capability, AGENT-REFLEX comprises three stages:

(1) Environment Reflection Skill Discovery (RSD) (§ 3.1). Build failure-aware executable environments and mine verifier-matched pairs into an offline skill library.

(2) Reflective Supervised Fine-Tuning (ReSFT) (§ 3.2). Convert each skill into verifier-validated self-reflection, recovery, and user-correction data.

(3) On-Policy Reflective Policy Optimization (ORPO) (§ 3.3). Combine Noisy Agent RL with same-token skill-guided OPD to provide dense reflection supervision.

3.1 STAGE 1: ENVIRONMENT REFLECTION SKILL DISCOVERY (RSD)

3.1.1 NOISE-GROUNDED ENVIRONMENT SCALING

To construct realistic environments with tool failures, we select the 500 most-used MCP server specifications from Smithery[1], each defining an environment topic $m \in \mathcal{M}$. Following Agent-World (Dong et al., 2026), a research agent equipped with search, browser, coding, and operating-

[1] https://smithery.ai/

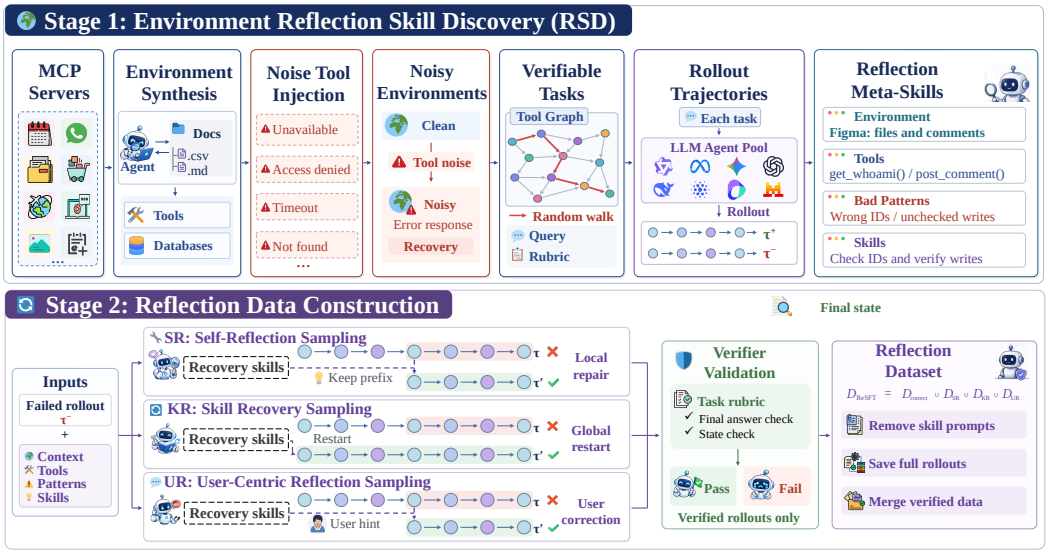


Figure 2: **RSD and ReSFT. Stage 1** perturbs tool returns of 500 MCP-anchored executable environments with mined failure families, contrasting success/failure rollouts into an offline skill library. **Stage 2** samples local-repair (SR), global-recovery (KR), and user-correction (UR) continuations from that library, keeping only verifier-validated ones with skill prompts stripped.

system tools builds and iteratively enriches a topic-grounded database. A coding agent then generates candidate tools and unit tests from each topic and its database. We retain only tools that compile, pass all tests, satisfy state-transition constraints, and support executable task verification. Thus, each environment $\mathcal{E}_m$ consists of a database, a validated tool set, and corresponding verifiers. The resulting ecosystem is $\mathcal{E} = \{\mathcal{E}_m : m \in \mathcal{M}\}$, with 60 verifier-backed tasks per topic; the resulting task set $\mathcal{Q}$ supplies the ReSFT data.

Tool-Failure Mining and Injection. Because real deployment is unstable, we ground perturbations in observed failures rather than invent them. We replay agents across nine public agentic benchmarks whose environments expose production-grade tool ecosystems and log every tool return (Appendix D.2). In total we collect 177,700 tool returns, of which 24,510 are error returns, and group them by surface form into eight actionable failure families (Table 4). We then inject these failures into tool returns probabilistically to emulate real deployment:

$$o_{t+1}^{\text{train}} = \begin{cases} J(o_{t+1}), & b_{t+1}^{\text{inj}} = 1, \\ o_{t+1}, & b_{t+1}^{\text{inj}} = 0, \end{cases} \qquad b_{t+1}^{\text{inj}} \sim \text{Bernoulli}(p_{\text{inj}}), \tag{1}$$

where the renderer J replaces the true return with a failure drawn from those compatible with the invoked tool f_t, balanced across the eight families so that rare ones still receive enough instances for recovery to be learned. We set p_{inj} and keep clean and perturbed tasks only if they admit a verifiable solution and recovery, respectively.

3.1.2 Reflection Meta-Skill Extraction

Capability-Diverse Trajectory Sampling and Pairing. Repeated sampling from one model can concentrate trajectories around its model-specific reasoning tendencies (Jiang et al., 2023; Du et al., 2024). We therefore sample trajectories from a capability-diverse LLM pool Π_{pool} of nine models spanning five families, ranging from 4B to 284B parameters (Table 5). Each model interacts with the noisy ecosystem $\mathcal{E}$ on every task $q \in \mathcal{Q}$ under the same sampling protocol.

Because rule-based metrics cannot reliably assess long-horizon trajectories, an analyzer model and the environment verifier jointly label each rollout, yielding $\mathcal{D}_{\text{correct}} = \{(q_i, \tau_i^+)\}_{i=1}^{N_+}$ and $\mathcal{D}_{\text{error}} = \{(q_i, \tau_i^-, e_i, t_i^\star, s_{t_i^\star})\}_{i=1}^{N_-}$, where τ_i^+ and τ_i^- denote correct and erroneous trajectories, e_i is the diagnosed error rationale, $t_i^\star$ is the first erroneous turn, and $s_{t_i^\star}$ is its environment-state snapshot. We pair correct and erroneous trajectories from the same task, limit the number of pairs contributed

by each task, and require each environment to draw pairs from at least five distinct tasks, forming: $\mathcal{D}_{\text{pair}} = \{(q, \tau^+, \tau^-, e, t^\star, s_{t^\star})\}$.

Meta-Skill Extraction. An ideal reflective meta-skill is an environment recovery rule that maps an agent's bad pattern to a corrective action, enabling reuse across tasks within the same environment. To mine such skills, an analyzer examines paired successful and failed trajectories, identifies recurring failure patterns, and converts them into executable recovery rules. For each environment m, we store $\zeta_m = (d_m, \mathcal{T}_m, \mathcal{B}_m, \mathcal{K}_m)$, where d_m describes the environment and its business objects, $\mathcal{T}_m$ its tools, $\mathcal{B}_m$ its observed bad patterns, and $\mathcal{K}_m$ the extracted reflection skills. The resulting offline library is $\mathcal{K} = \bigcup_{m \in \mathcal{M}} \mathcal{K}_m$, so each sample in $\mathcal{D}_{\text{pair}}$ is associated with a skill $\kappa \in \mathcal{K}_m$. Representative environment-specific skills are shown in Appendix E.3.

3.2 Stage 2: Reflective Supervised Fine-Tuning (ReSFT)

ReSFT converts explicit reflection skills into policy behavior. Starting from a frozen behavior policy π_0, we construct three complementary forms of reflective supervision for local repair, global recovery, and user-guided correction.

Self-Reflection Sampling (SR). For an erroneous trajectory τ^-, let $t^\star$ be its first erroneous turn, $s_{t^\star}$ the environment-state snapshot taken before that turn is executed, $h_{t^\star}$ the deployment-visible history up to that turn, and κ the matched reflection skill. Let $x \oplus z$ denote appending z to context x, and let $\text{Ver}_s(x, y)$ indicate whether continuation y successfully completes the task after restoring the environment to state s. To teach failure awareness and self-reflection, we restore the environment to $s_{t^\star}$ and continue after appending the corrective skill:

$$y^{\text{SR}} \sim \pi_0(\cdot \mid h_{t^\star} \oplus \kappa), \qquad \mathcal{D}_{\text{SR}} = \{(h_{t^\star}, y^{\text{SR}}) : \text{Ver}_{s_{t^\star}}(h_{t^\star}, y^{\text{SR}}) = 1\}. \tag{2}$$

The skill triggers diagnosis and replanning during generation but is removed from the stored input, so the policy learns to recover from its own error state.

Skill Recovery Sampling (KR). Many failures originate from an incorrect initial plan rather than a local tool error. To teach global recovery, we reset the environment to its initial state s_0 and restart from the original task with the matched skill:

$$y^{\text{KR}} \sim \pi_0(\cdot \mid q \oplus \kappa), \qquad \mathcal{D}_{\text{KR}} = \{(q, y^{\text{KR}}) : \text{Ver}_{s_0}(q, y^{\text{KR}}) = 1\}. \tag{3}$$

As in SR, the privileged skill is removed before storage, encouraging recovery without an explicit skill prior.

User-Centric Reflection Sampling (UR). When an agent deviates from the user's intent, users may intervene and correct its reasoning. To teach user-centric reflection, a user simulator U converts the matched skill into a natural-language correction: $u_\kappa = U(\kappa, h_{t^\star}, s_{t^\star})$. We then restore the environment to $s_{t^\star}$, insert the correction after the erroneous turn, and continue:

$$y^{\text{UR}} \sim \pi_0(\cdot \mid h_{t^\star} \oplus u_\kappa), \qquad \mathcal{D}_{\text{UR}} = \{(h_{t^\star} \oplus u_\kappa, y^{\text{UR}}) : \text{Ver}_{s_{t^\star}}(h_{t^\star} \oplus u_\kappa, y^{\text{UR}}) = 1\}. \tag{4}$$

Because the user message is deployment-visible, u_κ remains in the stored input, whereas the privileged skill κ is removed.

Let $\mathcal{D}^{\pi_0}_{\text{correct}} \subseteq \mathcal{D}_{\text{correct}}$ denote correct trajectories generated by the target policy π_0. We merge them with the three reflective datasets:

$$\mathcal{D}_{\text{ReSFT}} = \mathcal{D}^{\pi_0}_{\text{correct}} \cup \mathcal{D}_{\text{SR}} \cup \mathcal{D}_{\text{KR}} \cup \mathcal{D}_{\text{UR}}, \qquad \mathcal{L}_{\text{ReSFT}} = -\sum_{(x,y) \in \mathcal{D}_{\text{ReSFT}}} \log \pi_\theta(y \mid x). \tag{5}$$

3.3 Stage 3: On-Policy Reflective Policy Optimization (ORPO)

ReSFT provides an offline initialization by internalizing explicit reflection skills. To further elicit proactive reflection from the current policy on its own on-policy rollouts, ORPO supplements outcome-based agentic reinforcement learning with token-level, reflection-guided on-policy distillation (OPD) (Agarwal et al., 2024; Zhao et al., 2026; Wu et al., 2026), providing dense supervision for failure-causing decisions.

Noisy Agentic Reinforcement Learning. At update u, we draw a task q from the EnvScaler RL task pool $\mathcal{Q}_{\text{RL}}$ (Song et al., 2026), which ORPO directly reuses instead of RSD-synthesized tasks, and

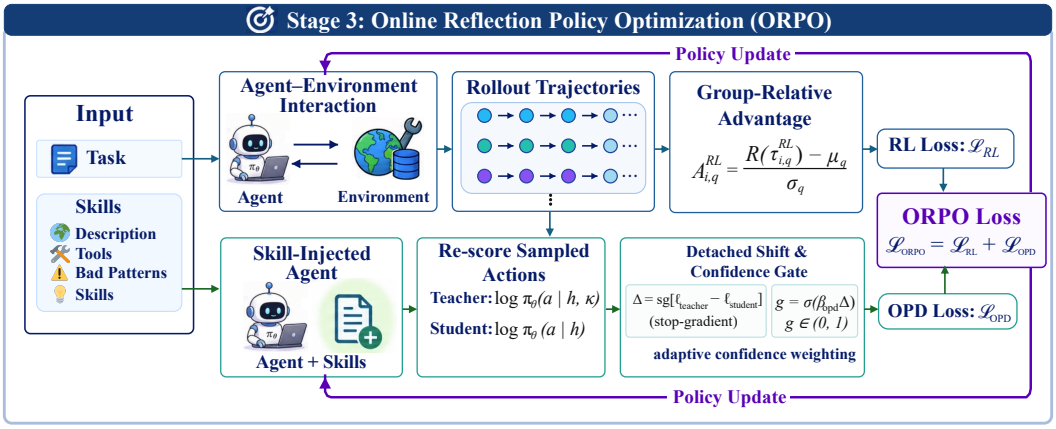


Figure 3: **ORPO.** A mixed rollout group of mostly clean and a few perturbed trajectories shares one group-normalized advantage. Re-scoring the same tokens with and without the skill card of their environment yields the detached, confidence-weighted OPD signal.

construct one mixed on-policy group to expose the behavior policy to noisy environment feedback. Specifically, the policy $\pi_{\theta_{\text{old}}}$ generates G trajectories $\mathcal{G}_q = \{\tau_q^{(i)}\}_{i=1}^{G}$, with a fraction ν rolled out in noisy environments and the remainder in clean ones. All trajectories retain the complete tool interaction and final answer, and none is given a reflection-skill prompt. After rollout, the environment verifier assigns each trajectory a return $R_{q,i}$; we normalize these returns within the mixed group:

$$\mu_q = \frac{1}{G}\sum_{i=1}^{G} R_{q,i}, \qquad \sigma_q^2 = \frac{1}{G}\sum_{i=1}^{G}\big(R_{q,i} - \mu_q\big)^2, \qquad \widehat{A}_{q,i} = \frac{R_{q,i} - \mu_q}{\sigma_q + \epsilon}, \tag{6}$$

where normalizing over the mixed group rather than within its clean and perturbed strata keeps genuine failures inside the on-policy distribution without letting them form a contrast of their own. For an agent action token $a_{q,i,t,k}$ at position k of turn t under history $h_{q,i,t}$, we define the token log-probability of the current policy and its ratio to the behavior policy:

$$\ell^{\theta}_{q,i,t,k} = \log \pi_\theta\big(a_{q,i,t,k} \mid h_{q,i,t}, a_{q,i,t,<k}\big), \qquad \rho_{q,i,t,k}(\theta) = \exp\big(\ell^{\theta}_{q,i,t,k} - \ell^{\text{old}}_{q,i,t,k}\big), \tag{7}$$

with ℓ^{old} the same score under $\pi_{\theta_{\text{old}}}$. Letting $\chi_{q,i,t,k} \in \{0,1\}$ mask the valid agent-generated tokens, which excludes tool returns and simulated user messages, the clipped Agent RL objective is

$$\mathcal{L}_{\text{RL}}(\theta) = -\,\mathbb{E}_{q,i,t,k}\Big[\chi_{q,i,t,k}\min\big(\rho_{q,i,t,k}\widehat{A}_{q,i},\ \text{clip}(\rho_{q,i,t,k}, 1-\epsilon_c, 1+\epsilon_c)\,\widehat{A}_{q,i}\big)\Big] + \beta_{\text{KL}}\,\mathbb{E}_{h\sim\mathcal{G}_q}\big[D_{\text{KL}}\big(\pi_\theta(\cdot \mid h)\|\pi_{\text{ref}}(\cdot \mid h)\big)\big], \tag{8}$$

with $\epsilon, \epsilon_c, \beta_{\text{KL}}$ hyperparameters and $\mathbb{E}_{q,i,t,k}$ the mean over all sampled token positions of the group. This component propagates only the outcome supervision of the verifier; the reflection-specific token-level signal comes next.

Reflection On-Policy Distillation. To turn the sparse outcome signal into dense supervision over reflective decisions, we condition on the reflection meta-skills $\kappa_{q,i} \in \mathcal{K}_m$ of the environment m that task q instantiates. No retrieval or extra matching step is involved: each EnvScaler RL task already specifies its environment, and the skill library is organized by environment, so the corresponding skill card is loaded directly with the environment and never depends on a benchmark answer. Prompting the same policy π_θ with this skill prepended to the task turns it into a teacher that re-scores the student rollout sampled without any skill prior: a deterministic function H builds the skill-augmented history $\widetilde{h}_{q,i,t} = H(h_{q,i,t}, \kappa_{q,i})$, and the current policy re-scores the already sampled action token under it,

$$\ell^{\text{skill}}_{q,i,t,k} = \log \pi_\theta\big(a_{q,i,t,k} \mid \widetilde{h}_{q,i,t}, a_{q,i,t,<k}\big), \tag{9}$$

which is contrasted with the ordinary-context score $\ell^{\theta}_{q,i,t,k}$ of Equation 7 on exactly the same token. OPD therefore needs a single environment rollout stream: the skill branch only scores tokens as a training-time privileged teacher, and gradients flow through the ordinary branch alone. Following SEED, the detached skill-induced shift and its confidence weight are

$$\Delta_{q,i,t,k} = \text{sg}\big[\ell^{\text{skill}}_{q,i,t,k} - \ell^{\theta}_{q,i,t,k}\big], \qquad \gamma_{q,i,t,k} = \text{sigmoid}\big(\beta_{\text{opd}}\Delta_{q,i,t,k}\big), \tag{10}$$

Table 1: **Main results on five tool-use benchmarks.** Overall is the unweighted mean of the five benchmark averages; Avg. of BFCL V4 and ACEBench follows each benchmark's official scoring rather than a simple mean. Best and second-best among 7B–8B methods are **bolded** and underlined (ties share a mark). *Reproduced by us because the checkpoint is not open-sourced

Method	VitaBench					τ^2-Bench				BFCL V4							ACEBench			Tool Sandbox	Overall
	Cro.	Del.	In.	OTA.	Avg.	Ret.	Air.	Tel.	Avg.	Web.	Mem.	Mul.	NoL.	Liv.	Irr.	Avg.	Ste.	Tur.	Avg.		
Foundation Models																					
GPT-5.2	24.3	52.5	45.8	28.3	37.7	81.6	62.5	95.8	80.0	75.5	45.8	48.5	81.9	70.4	88.7	62.9	95.0	50.0	68.0	73.3	64.4
Claude Sonnet 4.5	23.5	52.0	48.3	28.3	38.0	86.2	70.1	98.0	84.7	81.0	65.0	61.4	88.7	81.1	86.6	73.2	100.0	70.0	82.0	85.0	72.6
Gemini 3 Pro	31.5	62.0	56.5	40.3	47.6	85.3	72.7	98.0	85.4	80.0	61.7	60.8	90.7	83.1	85.6	72.5	100.0	50.0	70.0	88.4	72.8
Qwen3-235B-A22B	14.5	45.0	32.0	15.8	26.8	71.9	45.6	58.0	58.5	54.0	23.9	45.4	37.4	68.9	81.7	48.0	100.0	63.3	78.0	77.1	57.7
Qwen3-8B	1.2	15.5	24.8	4.0	11.4	34.0	26.5	18.0	26.2	7.0	17.6	35.4	90.2	80.9	77.2	40.4	80.0	36.7	54.0	69.1	40.2
Reflection Methods (8B)																					
Reflexion-8B	1.0	13.0	11.0	2.0	6.8	43.0	36.0	24.6	34.5	3.5	20.9	46.5	86.8	80.2	72.1	42.7	75.0	56.7	64.0	75.4	44.7
ExpeL-8B	0.0	14.0	11.0	4.0	7.3	45.6	34.0	21.1	33.6	2.5	23.0	39.5	88.8	78.2	80.2	41.7	55.0	36.7	44.0	63.0	37.9
Agent-R-8B*	2.0	15.0	8.0	4.0	7.3	37.7	32.0	21.9	30.6	1.0	17.9	35.4	87.6	80.2	74.7	38.7	60.0	43.3	50.0	73.7	40.1
NoisyAgent-8B*	5.0	22.0	27.0	7.0	15.3	47.6	40.0	40.8	42.8	2.5	13.3	34.8	65.4	79.9	74.6	35.6	50.0	53.3	52.0	71.5	43.4
Environment-Scaling Methods (7B–8B)																					
Simulator-8B	0.0	1.2	4.8	1.0	1.8	32.2	34.0	29.2	31.8	17.5	6.0	4.1	47.6	44.6	**87.3**	23.9	35.0	30.0	32.0	61.7	30.2
TOUCAN-7B	0.5	6.9	3.0	0.8	2.8	22.8	20.0	10.5	17.7	21.0	18.5	17.8	81.0	73.9	78.6	36.6	70.0	26.7	44.0	61.9	32.6
AWM-8B	1.5	16.0	19.2	4.2	10.2	41.2	23.5	38.5	34.4	9.5	15.7	34.9	**90.2**	80.5	73.9	40.0	60.0	46.7	52.0	71.5	41.6
EnvACE-8B*	**6.0**	24.0	27.0	7.0	16.0	48.9	44.0	17.3	36.7	12.3	**24.0**	45.3	87.6	81.2	83.2	46.0	55.0	43.3	48.0	71.6	43.7
EnvScaler-8B	**6.0**	20.8	**29.5**	7.0	15.8	49.6	31.5	32.7	37.9	23.0	21.9	**47.1**	88.5	**82.2**	74.6	47.6	**85.0**	60.0	72.5	80.5	50.9
Agent-World-8B*	**6.0**	21.0	11.0	**20.0**	14.5	**72.8**	40.0	50.9	54.6	**47.0**	21.7	44.5	83.3	79.6	80.2	51.4	**85.0**	63.3	72.0	75.6	53.6
Agent-Reflex-8B (ours)	**6.0**	**27.0**	29.0	8.0	**17.5**	64.6	**46.0**	**65.8**	**58.8**	40.0	23.5	**47.1**	88.8	81.4	81.4	**52.0**	**85.0**	**73.3**	**78.0**	**81.8**	**57.6**

and we optimize

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathbb{E}_{q,i,t,k}\Big[\chi_{q,i,t,k}\,\gamma_{q,i,t,k}\big(\mathrm{sg}[\ell^{\mathrm{skill}}_{q,i,t,k}] - \ell^{\theta}_{q,i,t,k}\big)\Big], \tag{11}$$

where $\mathrm{sg}[\cdot]$ blocks gradients through the teacher branch, so minimizing $\mathcal{L}_{\mathrm{OPD}}$ raises the student score on teacher-endorsed on-policy tokens. The confidence weight $\gamma \in (0, 1)$ is a soft weighting rather than a hard filter: it emphasizes tokens with a positive skill-induced shift and down-weights tokens whose shift is weak or negative.

Joint Objective. We jointly optimize the outcome-based Agent RL and reflection OPD objectives:

$$\mathcal{L}_{\mathrm{ORPO}} = \mathcal{L}_{\mathrm{RL}} + \lambda_{\mathrm{opd}}\mathcal{L}_{\mathrm{OPD}}. \tag{12}$$

The objective combines trajectory-level outcome learning with token-level reflection supervision, further sharpening the proactive reflection of the policy.

4 Experiments

4.1 Experimental Setups

Datasets. We evaluate our method on two groups of benchmarks. (1) **Long-horizon tool use** covers general function calling and interactive workflows: BFCL-v4 (Patil et al., 2025), ACEBench (Chen et al., 2025), 200 stratified ToolSandbox tasks (Lu et al., 2025), τ^2-Bench (Barres et al., 2025), and VitaBench (He et al., 2025). (2) **Noisy interaction** uses AgentNoiseBench (Wang et al., 2026a), instantiated on the τ^2-Bench and VitaBench tool- and user-noise settings. Following the EnvScaler protocol (Song et al., 2026), all evaluations use the standard MCP format and task templates, and each benchmark is scored with its official metric. Appendix B specifies every benchmark.

Baselines. Every trained method here, including AGENT-REFLEX, starts from the same Qwen3-8B backbone under one fair experimental setting. We compare three families: *reflection methods* (Shinn et al., 2023; Zhao et al., 2024; Yuan et al., 2025; Bi et al., 2026; Chen et al., 2026), *environment-scaling methods* (Li et al., 2025; Xu et al., 2025; Song et al., 2026; Wang et al., 2026b; Xu et al., 2026;

Table 2: **Results on AgentNoiseBench.** We report performance under tool and user noise. Avg. is the mean of the tool-noise and user-noise scores, and overall is the mean of the two benchmark.

Method	AgentNoiseBench-τ^2				AgentNoiseBench-Vita				Overall
	Clean	Tool Noise	User Noise	Avg.	*Clean*	Tool Noise	User Noise	Avg.	
Foundation Models									
Qwen3-8B	26.2	14.1	26.2	20.2	11.4	3.3	5.5	4.4	12.3
Gemini-2.5-Pro	60.0	37.0	55.0	46.0	49.0	34.0	38.0	36.0	41.0
Claude-4.5-Sonnet	73.0	57.0	65.0	61.0	51.0	41.0	41.0	41.0	51.0
GPT-5.2	81.0	**61.0**	**72.0**	**66.5**	53.0	**45.0**	**45.0**	**45.0**	**55.8**
Reflection and Environment Scaling (Qwen3-8B)									
Reflexion-8B	34.5	17.2	29.3	23.3	6.8	7.8	0.2	4.0	13.7
Agent-R-8B*	30.6	14.5	25.4	20.0	7.3	3.9	0.8	2.4	11.2
NoisyAgent-8B	42.8	–	–	37.6	15.3	–	–	**14.2**	25.9
AWM-8B	34.4	14.1	22.6	18.4	10.2	3.2	0.6	1.9	10.2
EnvScaler-8B	37.9	20.8	33.4	27.1	15.8	5.0	13.6	9.3	18.2
Agent-Reflex-8B (ours)	58.8	**38.4**	**53.4**	**45.9**	17.5	**9.4**	**15.2**	12.3	**29.1**

Dong et al., 2026), and *trajectory-level RL algorithms* (Shao et al., 2024; Yu et al., 2025; Hu et al., 2025; Wu et al., 2026). Appendix C specifies every baseline and Appendix D the implementation.

4.2 Main Results

4.2.1 Results on Long-Horizon Tool Use.

Table 1 shows that Agent-Reflex performs well on five long-horizon agent benchmarks, achieving the highest overall score among strong baselines. We further make the following observations.

(1) Internalizing Reflection beyond Inference-Time Guidance. Agent-Reflex outperforms Reflexion-8B and ExpeL-8B on VitaBench, ACEBench, and ToolSandbox. On VitaBench, Agent-Reflex achieves 17.5%, compared with 6.8% for Reflexion-8B. While these methods rely on self-generated critiques or retrieved experience at inference time, Agent-Reflex trains directly on verified recovery behavior. This may help the policy translate reflection into corrective actions more reliably, reducing its reliance on interpreting textual advice during execution.

(2) Reusable Recovery Knowledge beyond Individual Trajectories. Agent-Reflex surpasses Agent-R-8B and NoisyAgent-8B on VitaBench, BFCL, ACEBench, and ToolSandbox. On ACEBench, it reaches 78.0%, compared with 52.0% for NoisyAgent-8B. These gains suggest that environment-level meta-skills provide useful recovery guidance beyond individual correction trajectories or episode-level rewards.

(3) Reflective Supervision Complements Environment Scaling. Agent-Reflex improves over EnvScaler-8B and Agent-World-8B on VitaBench, ACEBench, and ToolSandbox averages. On ACEBench, the gain comes mainly from multi-turn interaction: Agent-Reflex reaches 73.3% at the turn level while matching the baselines at 85.0% step-level accuracy. This suggests that reflective supervision complements broader tool and task coverage.

4.2.2 Robustness under Interaction Noise.

To assess whether Agent-Reflex remains effective when tool feedback or user interactions are perturbed, we evaluate it on AgentNoiseBench. In detail, every foundation model degrades markedly.

Generalization of Agent-Reflex to Unseen Perturbations. Table 2 shows that Agent-Reflex achieves the highest Overall score among the compared 8B methods, reaching 29.1% compared with 25.9% for NoisyAgent-8B and outperforming the other strong baselines overall. These results indicate that its effectiveness extends to noisy interaction. Appendix E.1 provides a fine-grained attribution of the observed noise-robustness gains.

4.3 Does On-Policy Reflective Optimization Improve Robustness?

To further assess ORPO's effectiveness for reflection training, we analyze along two dimensions.

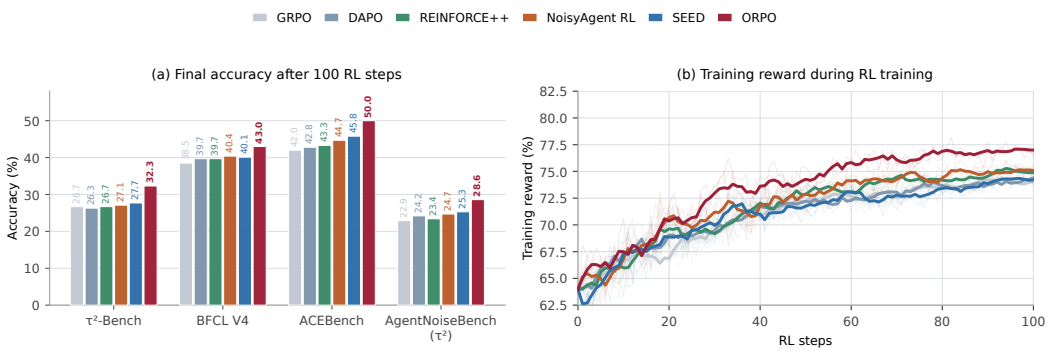


Figure 4: **Comparison of agent policy optimization algorithms.** All runs start from Qwen3-8B and train for 100 RL steps on the same EnvScaler task pool under matched budgets. **(a)** Final accuracy on held-out benchmarks. **(b)** Mean training reward on the same tasks.

(1) Performance Comparison. To evaluate online reflective learning, we train five representative algorithms and ORPO from Qwen3-8B for 100 RL steps on the same task pool. In Figure 4(a), ORPO holds a stable advantage over GRPO, DAPO, and REINFORCE++ on BFCL V4, τ^2-Bench, and real interaction noise. The advantage also holds against SEED, which already carries an OPD signal, and against noisy Agent RL, so reflection meta-skills summarized from real failures transfer better than experience estimated from rollouts alone.

(2) Training Dynamics. To assess ORPO's stability, Figure 4(b) tracks mean rollout reward over the same 100 steps. ORPO separates from GRPO and SEED early and keeps rising after both flatten. We attribute this stability to the confidence weighting, which emphasizes tokens with a positive skill-induced shift and down-weights those with weak or negative shifts, so skill-neutral tokens do not dominate the update.

4.4 Ablation Study

Table 3 reports an ablation study of AGENT-REFLEX on the two AgentNoiseBench settings. Each row removes one component while keeping the remaining training and evaluation conditions fixed, and the complete-model scores match those in Table 2. We make three observations. **(1)** Removing any component degrades both settings, showing that the components are complementary. **(2)** The supervision types follow a consistent ordering: removing SR causes the largest drop, followed by KR and UR, suggesting that failure awareness and local repair form the foundation of reflective behavior. **(3)** The skill library alone is insufficient: with the library retained, removing OPD clearly hurts performance, while removing only its confidence weighting also causes a decline. Thus, the gains arise from the token-level reflection signal rather than skill exposure alone.

Table 3: **Component ablation.** Agent-NoiseBench averages per variant; shading is the within-column normalized drop, so darker marks a more load-bearing module.

Variant	ANB-τ^2	ANB-Vita
AGENT-REFLEX	**45.9**	**12.3**
w/o SR	41.3	8.9
w/o KR	42.4	10.2
w/o UR	43.0	10.9
w/o OPD	43.7	9.5
w/o weight	44.5	10.4

5 Conclusion

In this paper, we present AGENT-REFLEX, a general training framework for incentivizing proactive reflective thinking in agents. In detail, RSD mines reusable reflection meta-skills from matched success–failure trajectories under grounded tool-feedback perturbations. Building on this skill library, RESFT turns each skill into complementary supervision for self-reflection, skill recovery, and user-centric correction, and ORPO further elicits proactive reflection by combining agentic reinforcement learning under noisy interaction with reflection-guided on-policy distillation. Across six challenging agent benchmark suites, AGENT-REFLEX improves long-horizon tool use and robustness under tool and user noise with stable training dynamics, supporting scalable agent training.

AI USE STATEMENT

We used generative AI tools solely to assist with literature reading and to proofread and polish author-written text. These tools were not used to develop research ideas, design experiments, conduct analyses, generate results or citations, formulate survey or interview questions, or create or modify figures. We reviewed all AI-assisted content and take full responsibility for the final manuscript, including its claims and artifacts.

REPRODUCIBILITY STATEMENT

Readers can find information supporting the reproducibility of this work in Appendix D. It includes detailed information on environment construction, failure mining and noise injection, trajectory sampling and skill extraction, ReSFT and ORPO training procedures, model and optimization settings, evaluation protocols, and contamination controls.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://mlanthology.org/iclr/2024/agarwal2024iclr-onpolicy/.

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

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

Jinhe Bi, Chennan Zhou, Zengjie Jin, Aniri, Shuo Lu, Wenke Huang, Hu Cao, Xun Xiao, Zhihong Zhu, Volker Tresp, Fei Shen, Yunpu Ma, and Tat-Seng Chua. Reflectrl: Learning from golden negative trajectories via reflective-to-direct reasoning, 2026. URL https://arxiv.org/abs/2608.03972.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, Wulong Liu, Xinzhi Wang, Defu Lian, Baoqun Yin, Yasheng Wang, and Wu Liu. Acebench: Who wins the match point in tool usage?, 2025. URL https://arxiv.org/abs/2501.12851.

Yuxin Chen, Xiaodong Cai, Junfeng Fang, Zhuowen Han, Yu Wang, Yaorui Shi, Yi Zhang, Qi Gu, Xunliang Cai, Xiang Wang, An Zhang, and Tat-Seng Chua. Learning to act under noise: Enhancing agent robustness via noisy environments, 2026. URL https://arxiv.org/abs/2605.27209.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic entropy-balanced policy optimization, 2025a. URL https://arxiv.org/abs/2510.14545.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning, 2025b. URL https://arxiv.org/abs/2505.16410.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization, 2025c. URL https://arxiv.org/abs/2507.19849.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence, 2026. URL https://arxiv.org/abs/2604.18292.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu

Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Jiacheng Guo, Ling Yang, Peter Chen, Qixin Xiao, Yinjie Wang, Xinzhe Juan, Jiahao Qiu, Ke Shen, and Mengdi Wang. Genenv: Difficulty-aligned co-evolution between llm agents and environment simulators, 2025. URL https://arxiv.org/abs/2512.19682.

Aayush Gupta. Reliabilitybench: Evaluating llm agent reliability under production-like stress conditions, 2026. URL https://arxiv.org/abs/2601.06112.

Harvey. Harvey's legal agent benchmark (LAB). https://github.com/harveyai/harvey-labs, 2026.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, Man Gao, Xi Su, Xiaodong Cai, Xunliang Cai, Yu Yang, and Yunke Zhao. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications, 2025. URL https://arxiv.org/abs/2509.26490.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL https://arxiv.org/abs/2501.03262.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.792.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026a. URL https://arxiv.org/abs/2602.12670.

Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training, 2025. URL https://arxiv.org/abs/2511.01824.

Yuetai Li, Yichen Feng, Zhangchen Xu, Zixian Ma, Kaiyuan Zheng, Fengqing Jiang, Xinghua Sun, Rulin Shao, Zichen Chen, Yue Huang, Xinyang Han, Brian Lee, Kayla Xu, Shenglai Zeng, Hang Hua, Xiangliang Zhang, Basel Alomair, Ranjay Krishna, Luke Zettlemoyer, Pang Wei Koh, Bhaskar Ramasubramanian, Luyao Niu, Xiang Yue, and Radha Poovendran. Jobbench: Aligning agent work with human will, 2026b. URL https://arxiv.org/abs/2605.26329.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey,

Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. Officeqa pro: An enterprise benchmark for end-to-end grounded reasoning, 2026. URL https://arxiv.org/abs/2603.08655.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

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

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. Userrl: Training interactive user-centric agent via reinforcement learning, 2025. URL https://arxiv.org/abs/2509.19736.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework, 2024. URL https://arxiv.org/abs/2409.19256.

Daniel Shepard and Robin Salimans. Automationbench, 2026. URL https://arxiv.org/abs/2604.18934.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Sierra Research. τ^3-bench: Advancing agent evaluation to knowledge and voice. https://github.com/sierra-research/tau2-bench, 2026.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis, 2026. URL https://arxiv.org/abs/2601.05808.

Alane Suhr and Yoav Artzi. Continual learning for instruction following from real-time feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/666cccc6376058e251315b4de7e085b9-Abstract-Conference.html.

Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Yukai Wu, Weizheng Wang, Hongzhang Huang, Wei Zhou, Jun Zhou, Jiachen Song, Shaoli Yu, Jinqi Wang, Zihang Zhou, Hongyi Zhou, Yuting Lv, Jinyang Li, Jiashuo Liu, Ruoyu Chen, Chunwei Liu, GuoLiang Li, Jihua Kang, and Fan Wu. Workspace-bench 1.0: Benchmarking ai agents on workspace tasks with large-scale file dependencies, 2026. URL https://arxiv.org/abs/2605.03596.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim P oder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents, 2024. URL https://arxiv.org/abs/2407.18901.

Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, Qi Gu, Hui Su, and Xunliang Cai. Scaleenv: Scaling environment synthesis from scratch for generalist interactive tool-use agent training, 2026. URL https://arxiv.org/abs/2602.06820.

Ruipeng Wang, Yuxin Chen, Yukai Wang, Chang Wu, Junfeng Fang, Xiaodong Cai, Qi Gu, Hui Su, An Zhang, Xiang Wang, Xunliang Cai, and Tat-Seng Chua. Agentnoisebench: Benchmarking robustness of tool-using llm agents under noisy condition, 2026a. URL https://arxiv.org/abs/2602.11348.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning, 2026b. URL https://arxiv.org/abs/2602.10090.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2607.14777.

Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. Toucan: Synthesizing 1.5m tool-agentic data from real-world mcp environments, 2025. URL https://arxiv.org/abs/2510.01179.

Zishan Xu, Zhiyuan Yao, Yuxin Chen, Yifu Guo, Zhengxi Lu, Yuquan Lu, Jinyang Huang, Yan Xu, Yasheng Wang, Weinan Zhang, Xingshan Zeng, and Weiwen Liu. Envace: Internalizing environment dynamics via world rehearsal for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2608.06197.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training, 2025. URL https://arxiv.org/abs/2501.11425.

Zhixin Zhang, Xinke Jiang, Zhibang Yang, Weixuan Xu, Guohong Qiu, Xu Chu, Junfeng Zhao, and Yasha Wang. Loongreflect: Boosting long-horizon reflection in search agents via global perspective distillation, 2026. URL https://arxiv.org/abs/2608.11967.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models, 2024. URL https://arxiv.org/abs/2403.13372.

APPENDIX

A Supplement Related Work **18**

A.1 Agent Reflection Learning 18

A.2 Scalable Environment Synthesis for Agent Training 18

A.3 Agentic Reinforcement Learning 19

B Datasets **19**

B.1 General Function Calling 19

B.2 Long-Horizon Interactive Workflows 19

B.3 Noisy Interaction 20

B.4 Training Task Pools 20

C Baselines **20**

C.1 Foundation Models 20

C.2 Reflection Methods 20

C.3 Environment-Scaling Methods 21

C.4 Agent Policy Optimization Algorithms 21

D Implementation Details **21**

D.1 Training Guideline 21

D.2 Stage 1: Environment Construction and Skill Discovery 21

D.3 Stage 2: Reflective Supervised Fine-Tuning 22

D.4 Stage 3: On-Policy Reflective Policy Optimization 23

D.5 Evaluation Protocol and Contamination Control 24

E Supplement Experimental Results **24**

E.1 Does AGENT-REFLEX Act Reflectively Under Noise? 24

E.2 Why AgentNoiseBench Is Out of Distribution for AGENT-REFLEX 25

E.3 Representative Environment-Specific Skills 25

F Limitations **28**

G Broader Impacts and Future Directions **28**

H Ethics statement **29**

A Supplement Related Work

A.1 Agent Reflection Learning

Agent reflection studies how interaction feedback can reveal errors and guide subsequent behavior. The methods differ along one axis that matters for our claims: *where the corrective signal lives*: in the context at inference time, in an offline trajectory rewrite, or in the parameters.

Inference-time reflection. The first family leaves parameters untouched and reflects in context. Reflexion appends a verbal self-critique to an episodic buffer that later attempts read (Shinn et al., 2023), and Self-Refine iterates generation, self-feedback, and revision within a single episode (Madaan et al., 2023); ExpeL instead accumulates cross-task natural-language insights and retrieves similar past trajectories at test time (Zhao et al., 2024). These methods bound what prompting alone recovers, and that bound is informative: because the critique is produced by the same policy that failed, it is only as good as that policy's model of the environment. When a tool return is well-formed but wrong, there is nothing in the context to contradict the failed plan, so the critique tends to restate it. Our Table 1 shows this concretely: both Reflexion and ExpeL fall below their own backbone on the longest-horizon benchmark.

Learning from correction trajectories. The second family moves correction into training. Agent-R combines MCTS with model-guided selection of the revision point to construct recovery trajectories, then learns timely correction by iterative self-training (Yuan et al., 2025). ReflectRL transfers reflection elicited on golden negative trajectories into direct reasoning (Bi et al., 2026), and LoongReflect combines privileged reflection supervision with outcome-based RL (Zhang et al., 2026), the closest prior work to our ORPO stage in supervision *type*, though its privileged context is an episode-level hindsight rewrite rather than an environment-level artifact. The shared limitation is that the correction is synthesized per trajectory: each repair is fitted to the episode that produced it, so nothing accumulates into a statement about the environment that a later, differently-failing episode can reuse.

Robustness to perturbed interaction. A third line attacks the problem from the environment side. NoisyAgent perturbs user instructions and tool responses, mixes clean and noisy rollouts, and adaptively raises noise difficulty, giving a complementary route to recovery and ambiguity handling (Chen et al., 2026). This exposes the policy to failure states, which trajectory-level RL alone does not, but the supervision remains a scalar return: it can express that a perturbed episode was harder, not which recovery the specific failure admitted.

Positioning. Agent-Reflex shares the goal of learning from failure and differs in what it learns *from* and what it stores. Before any policy training, it consolidates cross-model behavioral contrasts (matched success/failure pairs on the same task under the same injected noise) into explicit, environment-indexed reflection meta-skills. A skill is therefore reusable across episodes and mined once per environment, rather than rewritten per trajectory. These skills then supply supervision for autonomous recovery and user-directed correction (ReSFT) and a privileged teacher context for token-level distillation (ORPO). Because the skill is privileged only during training and stripped from every stored input, inference needs no skill prompt and no auxiliary reflector, unlike the inference-time family above, where the critique must be produced online at deployment.

A.2 Scalable Environment Synthesis for Agent Training

Existing approaches split into two families. The first uses language models as implicit textual world models: Simia generates environment feedback conditioned on tool specifications and interaction histories, and GenEnv couples generative simulation with a capability-adapted curriculum (Li et al., 2025; Guo et al., 2025). This removes most environment-engineering cost, but the fidelity of a simulated observation is bounded by the simulator's ability to keep state consistent, so long rollouts can drift away from any realizable environment. The second family makes dynamics explicit through executable tools, databases, and programmatic verifiers. EnvScaler, AWM, and ScaleEnv construct environments whose transitions execute and whose outcomes are checkable (Song et al., 2026; Wang et al., 2026b; Tu et al., 2026); Agent-World mines databases from real service themes, validates generated tools against them, and lets environments and policies co-evolve through continuous task generation (Dong et al., 2026); EnvACE instead internalizes dynamics by *world rehearsal*, with one policy emitting both tool actions and the responses they induce (Xu et al., 2026); and TOUCAN takes a data-centric route, synthesizing tool-use trajectories from real MCP services (Xu et al., 2025).

These efforts are why a large pool of executable environments can be assumed today. What they do not address is the status of *failure* inside those environments: a verifier returning zero says an episode was hard, not which of dozens of tool calls broke the plan, what evidence exposed it, or which recovery the situation admitted. AGENT-REFLEX adopts the database-backed executable paradigm from this line and targets that residual.

A.3 AGENTIC REINFORCEMENT LEARNING

GRPO removes the value network by normalizing rewards within a sampled group (Shao et al., 2024), and DAPO and REINFORCE++ improve stability and sample efficiency through clip asymmetry, dynamic sampling, token-level loss aggregation, and variance reduction (Yu et al., 2025; Hu et al., 2025). Applying these to tool-using agents introduces a difficulty single-turn reasoning lacks: a long episode's return is one scalar attached to a decision sequence of unknown length. One line of work therefore reshapes the rollout or the credit assignment rather than the optimizer, through hierarchical rewards with a self-critic stage (Dong et al., 2025b), entropy-triggered partial rollouts with soft advantage attribution (Dong et al., 2025c), and balanced exploration across the agent and environment branches (Dong et al., 2025a). A complementary route leaves the reward untouched and densifies the gradient instead: on-policy distillation scores the student's own sampled tokens under a teacher context, which SEED instantiates for agents by re-scoring the same action token with and without privileged information and gating a stop-gradient log-likelihood shift through a sigmoid (Wu et al., 2026). ORPO is orthogonal to the first route and reuses SEED's estimator unchanged in the second; what it contributes is the content of the privileged slot, an environment-level skill card mined once from empirical tool failures rather than a larger model or a per-episode hindsight rewrite.

B DATASETS

All evaluations use the standard MCP interaction format and each benchmark's official metric, reported by domain.

B.1 GENERAL FUNCTION CALLING

- **BFCL-v4** (Patil et al., 2025) extends the Berkeley Function Calling Leaderboard from single-call accuracy toward agentic evaluation. We report Web, Memory, Multi-Turn, Non-Live, Live, and Irrelevance, and the average follows the official BFCL scoring procedure rather than a simple mean of the six categories. It is useful here because it separates call formatting from execution: a model can be near-saturated on Non-Live while failing the multi-turn categories.
- **ACEBench** (Chen et al., 2025) evaluates normal, special, and agentic tool use, emphasizing when to call a tool and how to handle underspecified requests. We report step-level and turn-level accuracy, and the average follows the official ACEBench scoring procedure rather than a simple mean of the two; the turn-level metric credits a turn only when every call in it is correct, making it sensitive to error propagation.
- **ToolSandbox** (Lu et al., 2025) is stateful and conversational, with implicit tool state dependencies, an LLM user simulator, and deliberately unsolvable tasks that require asking rather than acting. We use a stratified 200-task subset and release the task identifiers so the subset is reproducible.

B.2 LONG-HORIZON INTERACTIVE WORKFLOWS

- **τ^2-Bench** (Barres et al., 2025) is *dual-control*: agent and simulated user both act on shared state. We report Retail, Airline, and Telecom and their macro-average. Each domain supplies a policy document the agent must respect, and success requires the final database state to match the target, so partially correct trajectories score zero.
- **VitaBench** (He et al., 2025) covers delivery, in-store, and OTA domains plus composite *cross*-scenario tasks spanning domains within one episode. We report the four domains under official Avg@4 and take the benchmark average as the macro-average of the per-domain scores.

B.3 Noisy Interaction

AgentNoiseBench (Wang et al., 2026a) injects perturbations into both sides of the interaction loop on the τ^2-Bench and VitaBench environments. Tool noise covers faulty, incomplete, inducing, redundant, and wrong returns; user noise covers ambiguity, topic shift, conflicting instructions, redundancy, and boundary probing. We score with the official Stability-Gated Success metric, which credits a task only when the whole trajectory stays valid and therefore excludes episodes derailed by noise that answer correctly by chance. The benchmark is out of distribution for us: our perturbations are summarized entirely from large-scale real tool-call failures, were designed without any reference to AgentNoiseBench, and are injected by a different process; no AgentNoiseBench environment is used in training.

B.4 Training Task Pools

Two training pools must not be confused with the evaluation sets. **The RSD ecosystem** supplies the SFT data: 500 Smithery-anchored environments, each with a mined database, validated tools, executable verifiers, and 60 tasks, for 30,000 tasks (Section D.2), from which the ReSFT corpus is synthesized. **EnvScaler tasks** supply the RL data: ORPO directly reuses the 5,000 EnvScaler RL tasks (Song et al., 2026) rather than RSD-synthesized tasks, and every algorithm in Figure 4 trains on this same pool, isolating the objective from the Stage 1–2 data.

C Baselines

Foundation-model rows are reference points at a different scale; bold and underline in Table 1 are computed among 7B–8B methods only. Most baseline results are taken from the numbers reported in the corresponding papers. Only for methods that are not open-sourced, or for benchmarks that a paper does not report, do we reproduce the method locally, following the settings and technical route of the original paper.

C.1 Foundation Models

GPT-5.2, **Claude Sonnet 4.5**, and **Gemini 3 Pro** are the proprietary reference points, with reasoning-effort settings preserved in the row label wherever the source reports them. **Qwen3-235B-A22B** is a large open-weight mixture-of-experts model included to separate scale from recipe. **Qwen3-8B** is the backbone for every trained method here, including Agent-Reflex, and is the control that matters: any margin over it is attributable to the training recipe rather than to capacity.

C.2 Reflection Methods

- **Reflexion** (Shinn et al., 2023) stores verbal self-reflection in an episodic buffer consulted on later attempts, and **Self-Refine** (Madaan et al., 2023) iterates generation, self-feedback, and revision. Neither updates parameters, so together they bound what inference-time reflection recovers.
- **ExpeL** (Zhao et al., 2024) accumulates cross-task experience as natural-language insights and retrieves similar past trajectories at test time: parameter-free, but with cross-task rather than within-episode memory.
- **Agent-R** (Yuan et al., 2025) builds recovery trajectories by combining MCTS with model-guided selection of the revision point and learns timely correction by iterative self-training. It releases neither an 8B checkpoint nor its training data and does not report our benchmarks, so the rows marked * are our local reproduction following the original setting.
- **NoisyAgent** (Chen et al., 2026) trains on clean/perturbed mixtures with adaptively increasing noise difficulty. Its reported τ^2-Bench and VitaBench numbers come from RL on the evaluated domains themselves. Its rows in Tables 1 and 2 are quoted directly from the paper. Since no checkpoint is released, the optimization algorithm compared in Figure 4 is instead reproduced from the description in the paper.

C.3 Environment-Scaling Methods

- **Simulator** (Li et al., 2025) is the model-simulated-environment route and **TOUCAN** (Xu et al., 2025) the trajectory-synthesis route.
- **EnvScaler** (Song et al., 2026) constructs executable environments with verifiable outcomes and is the strongest environment-scaling baseline in our tables; because the ORPO stage of AGENT-REFLEX reuses its RL task pool, it is the reference for attributing our margin to reflective supervision rather than coverage.
- **AWM** (Wang et al., 2026b) learns an agent world model over transitions, and **ScaleEnv** (Tu et al., 2026) scales executable environment synthesis.
- **EnvACE** (Xu et al., 2026) internalizes dynamics through world rehearsal.
- **Agent-World** (Dong et al., 2026) synthesizes database-grounded environments of controllable difficulty over 2,000 environments.

C.4 Agent Policy Optimization Algorithms

These are the comparands in Figure 4, all trained from Qwen3-8B on the same 5,000 EnvScaler tasks under matched rollout, tool-call, and token budgets.

- **GRPO** (Shao et al., 2024) normalizes verifier rewards within a rollout group and is the base of our own outcome term.
- **DAPO** (Yu et al., 2025) adds decoupled clipping, dynamic sampling, token-level loss, and overlong-reward shaping.
- **REINFORCE++** (Hu et al., 2025) strengthens critic-free optimization with baseline-based variance reduction.
- **NoisyAgent RL** (Chen et al., 2026) appears here in its optimization role, as clean/noisy rollout mixing with an adaptive noise schedule, reproduced from the paper.
- **SEED** (Wu et al., 2026) adds same-action on-policy distillation to outcome-based Agent RL and is the most important comparand for ORPO, since it shares the estimator and differs only in what supplies the privileged context.

D Implementation Details

D.1 Training Guideline

Our aim is to test whether reflection can be mined, supervised, and internalized as a meta-skill, not to maximize a leaderboard number. All stages follow the cold-start-then-RL paradigm (Dong et al., 2025b) so RL does not begin from a policy that cannot complete a trajectory, and every trained method shares the Qwen3-8B backbone. All stages use **Qwen3-8B** as the backbone, with supervised fine-tuning implemented in LLaMA-Factory (Zheng et al., 2024) and reinforcement learning in VERL (Sheng et al., 2024). All RL baselines and ORPO use the same experimental settings and hyperparameters. Tokens belonging to tool returns and simulated user messages are excluded from the loss; only agent-generated tokens contribute gradient, which is the mask $\chi_{q,i,t,k}$ in Equations 8 and 11; the same exclusion applies to the RESFT loss of Equation 5.

D.2 Stage 1: Environment Construction and Skill Discovery

Environments. Environment topics are the 500 most-used deployed MCP servers in the Smithery registry, ranked locally by the registry's cumulative `useCount`; because that counter grows monotonically, we release the frozen list of identifiers so the set can be reconstructed exactly. Following Agent-World (Dong et al., 2026), a research agent with search, browser, code, and OS tools mines a topic-grounded database and expands it for $N = 3$ complexification rounds; a coding agent then generates candidate tools with unit tests, and a tool is retained only if it compiles, passes its tests, respects state-transition constraints, and admits an executable verifier. Each environment contributes 60 verifier-backed tasks, for 30,000 tasks in total. Both agents are GPT-OSS-120B (OpenAI et al.,

2025), matching the Agent-World setting so that environment construction is not advantaged by a stronger builder.

Failure mining and injection. Running agents across the nine public benchmarks listed below yields 177,700 tool returns of which 24,510 are errors. An analyzer model groups them by surface form into the eight families of Table 4; the families are multi-label, so one return may land in several counts. During construction each tool-return position is perturbed independently with probability $p_{\text{inj}} = 0.2$ per Equation 1. Both draws are balanced uniform: the family uniformly over the eight rather than in proportion to mined frequencies, and the realization uniformly among those compatible with the invoked tool. A clean task is kept only if it admits a verifiable solution, and a perturbed task only if it admits a verifiable recovery.

Benchmarks replayed for failure mining. The error returns above come from replaying agents across nine public agentic benchmarks, chosen to span interface types and task horizons rather than to maximize count. Most of them use mcp server and workspace for interaction and performance evaluation, so they are close to real deployment scenarios, including: MCP-Atlas (Bandi et al., 2026), τ^3-Bench (Sierra Research, 2026), AutomationBench (Shepard & Salimans, 2026), OfficeQA Pro (Opsahl-Ong et al., 2026), JobBench (Li et al., 2026b), Harvey LAB (Harvey, 2026), GDPval (Patwardhan et al., 2025), Workspace-Bench (Tang et al., 2026), and SkillsBench (Li et al., 2026a). Together they reach MCP servers, REST suites, retrieval backends, and file or OS toolchains. They serve *only* as a source of observed error returns for the taxonomy in Table 4; none is used for training or reported as a result.

Sampling, pairing, and extraction. Trajectories come from the nine-model pool of Table 5 under an identical tool-call limit, response budget, and sampling protocol, so differences reflect behavior rather than budget. DeepSeek-V4-Flash serves as both critique and analyzer model; together with the environment verifier it assigns the correctness label, and disagreements are discarded. Pairs form only within the same task, with at least five distinct tasks per environment; the analyzer then compares each pair and emits the environment record ζ_m, from which 1,170 reflection meta-skills are summarized in total.

Training-Trajectory Sources and Policy Initialization. The diverse pool produces behavioral contrasts for skill mining only; it never supplies a training target. Every trajectory the policy is trained on comes from Qwen3-8B itself: $\mathcal{D}_{\text{correct}}$ keeps only Qwen3-8B's verified successes, and all Stage-2 continuations are rolled out by the same backbone. A stronger model appears solely as the analyzer for error diagnosis and skill extraction, a task Qwen3-8B is not reliable enough to perform. At the level of the policy that generates trajectories we therefore stay at 8B throughout, which is what keeps the comparison against the 7B–8B baselines fair.

Upon publication, we will release all training data, the complete reflection skill library, the environment manifests, and the code needed to reproduce our results.

Skill record schema. A skill is a structured record rather than free prose: a trigger over visible history with its supporting and contradicting evidence, a diagnosis naming the failure family, a recovery template stating when to retry, switch endpoint, ask the user, or abort, and reversibility and partial-effect flags. Provenance and replay-validation fields travel with each record so that a skill remains auditable.

D.3 Stage 2: Reflective Supervised Fine-Tuning

The three sampling strategies of Section 3.2 share one actor for continuation rollouts, $\pi_0 =$ Qwen3-8B, the same backbone that is being trained, and one user simulator for UR, $U =$ GPT-4.1, following the τ^2-Bench setting (Barres et al., 2025). A sample enters the dataset only if the environment verifier accepts the resulting episode. The final corpus combines 30K verified Qwen3-8B successes with 20K reflective-correction samples spread over SR, KR, and UR, 50K trajectories in total. Two kinds of privileged information are kept out of every stored input. The simulator state snapshot is used to *restore* the environment before a rollout resumes and never appears as tokens in the context, so the policy sees only what it would see at deployment. The skill text is removed from the stored input in SR and KR, and in UR it never enters the context at all, since the simulator rewrites it into a natural user message u_κ; those synthesized user messages are retained, because a user turn is visible at deployment whereas a skill prompt is not. The UR simulator is prompted to cover direct correction,

Table 4: **Empirical tool-failure taxonomy used for noise injection.** Share is the fraction of non-exclusive pattern hits over 24,510 candidate error returns, matching Figure 1(d). Families differ in whether an unchanged retry is admissible at all.

Failure family	Share	Typical signals	Recovery implication
Server/gateway unavailable	33.6%	500/502/503/504	Retry with bounded backoff; switch endpoint after the stop rule.
Authentication/ permission	17.9%	401/403, permission denied	Refresh credentials, request authorization, or escalate; unchanged retry is insufficient.
Timeout	14.7%	timeout, read timeout	Retry or narrow the request after checking whether state changed.
Resource missing	10.5%	404, not found	Verify the identifier or discover the resource through search/list tools.
Argument/validation	10.2%	400/422, invalid argument	Repair schema, types, required fields, or enums before retrying.
Generic execution exception	8.3%	tool/operation failed	Inspect returned evidence and choose a typed recovery rather than blind repetition.
Rate limit	4.5%	429, too many requests	Back off, respect quota, batch requests, or switch an allowed provider.
Network connection	0.3%	reset/refused/network error	Treat as transient infrastructure failure unless repeated evidence suggests otherwise.

Table 5: **LLM agent pool for RSD trajectory sampling.** The pool only supplies behavioral contrasts for skill mining; all training trajectories come from Qwen3-8B.

Provider	Model family	Checkpoints for trajectory sampling
Alibaba	Qwen3 (Yang et al., 2025)	4B-Thinking, 8B, 14B, 32B, 235B-A22B
Meta	Llama (Grattafiori et al., 2024)	3.1-8B-Instruct
Google	Gemma (Team et al., 2025)	4-31B-it
OpenAI	GPT-OSS (OpenAI et al., 2025)	120B
DeepSeek-AI	DeepSeek	V4-Flash (284B)

partial hints, ambiguity, and deliberately incorrect advice, so compliance is not always the right response.

D.4 Stage 3: On-Policy Reflective Policy Optimization

ORPO starts from the ReSFT checkpoint with the skill library and reference policy frozen. ORPO directly reuses the 5K EnvScaler RL tasks rather than RSD-synthesized data, which keeps the comparison against EnvScaler and the other RL algorithms direct. For the OPD hyperparameters, we follow the standard settings of SEED (Wu et al., 2026) to ensure a fair comparison. Each update samples a task group, rolls out $G = 8$ ordinary trajectories of which a fraction $\nu = 0.25$ runs in perturbed environments, normalizes advantages over the whole mixed group, and re-scores the recorded action tokens under the skill-augmented context. The skill branch is a forward pass only: it never queries the environment and its gradient is detached, so the marginal cost over plain Agent RL is a forward pass rather than a rollout. The confidence weight γ is a soft token weight: it emphasizes tokens with a positive skill-induced shift and down-weights those whose shift is weak or negative, while every trajectory still contributes through the outcome term. We further conduct an ablation study on the perturbation fraction ν. Across the tested values, the performance remains relatively stable on the evaluated benchmarks, with only minor fluctuations and no consistent advantage for alternative settings. We therefore empirically set $\nu = 0.25$.

Meta-skill selection and environment alignment. The EnvScaler RL task pool is entirely distinct from the RSD task pool, but the two are aligned at the environment level: the more than 100 environ-

ments used to construct EnvScaler's RL tasks are covered by the 500 top-used MCP environments collected in Stage 1. Each environment and synthesized task is assigned a canonical environment identifier, allowing direct lookup of the corresponding skill:

$$\kappa_q = \text{Lookup}(\text{env_id}(q), \mathcal{K}).$$

Thus, skills mined from RSD trajectories are transferred from the environment level to different RL tasks instantiated within the same environment, without semantic retrieval or task-level overlap. This directly tests the reusability and transferability of the proposed environment-level skills. At inference time, the skill prompt is removed, so the deployed policy does not require access to the skill library.

D.5 Evaluation Protocol and Contamination Control

Every benchmark is scored with its official metric under the standard MCP format and task templates, following the EnvScaler protocol (Song et al., 2026). VitaBench uses official Avg@4 with the benchmark average recomputed from per-domain scores. AgentNoiseBench uses Stability-Gated Success; numbers we measure ourselves are marked † in Table 2, and AGENT-REFLEX is measured under exactly that protocol. Following EnvScaler and Agent-World (Song et al., 2026; Dong et al., 2026), every score we measure is the mean of three independent runs, and decoding settings, maximum turns, and tool-call limits follow each benchmark's standard setting. Most baseline numbers are taken from the results reported in the corresponding papers; only for methods that are not open-sourced or benchmarks that a paper does not report do we reproduce locally, following the original settings and technical route.

Noise taxonomy and row protocol in Table 2. *Tool Noise* aggregates fault, incomplete, induce, redundant and wrong returns; *User Noise* aggregates ambiguity, topic shift, conflict, redundancy and boundary probing. Unmarked rows are quoted from their source papers. †Measured by us as the mean of three runs (τ^2: Retail, Airline, and Telecom; Vita: Delivery, In-store, OTA); AGENT-REFLEX uses exactly the same protocol. *Reproduced locally because the paper releases no 8B checkpoint or training data. The NoisyAgent row reports only per-domain aggregates, so its two noise buckets are left blank.

Because agent training on synthesized environments is easy to contaminate, we state the controls explicitly. Splits are preserved at the *environment-family* level rather than the task level, since tasks within one environment share a database and tool set; generated instructions and skill text are checked for near-duplicates against benchmark instructions and tool schemas; and evaluation reference answers never appear in any analyzer, teacher, or user-simulator prompt. We release task IDs, environment hashes, versioned noise injectors and seeds, and the six prompt templates used by the pipeline.

E Supplement Experimental Results

This section reports the fine-grained noise attribution deferred from Section 4.

E.1 Does Agent-Reflex Act Reflectively Under Noise?

Task success alone cannot show whether an agent repaired for the right reason. We therefore split the AgentNoiseBench noise settings into finer diagnostic slices (seven tool-side and six user-side axes, which Table 2 aggregates into its tool- and user-noise buckets) and score each with the benchmark's Stability-Gated Success ($\text{SGA} = I_{\text{traj}} \times I_{\text{task}}$), which credits a rollout only if its trajectory was never derailed by the injected artifact, discarding turns pulled off course that answer correctly by luck. Each axis is normalized by its own three-model mean so noise-family difficulty cannot dominate the shape, with 95% paired bootstrap intervals over 165 tasks per axis. As shown in Figure 5, AGENT-REFLEX leads on all thirteen axes, with the margin widest exactly where each part of the design is meant to act.

Tool side. The two axes clearing the interval against *both* baselines are misleading lure (13.9, the largest tool-side margin) and incomplete information (10.9), the families whose returns are well-formed, so no schema check fires and only the evidence inside the payload reveals the fault. This is what RSD contributes and environment scaling cannot: more clean environments add successful

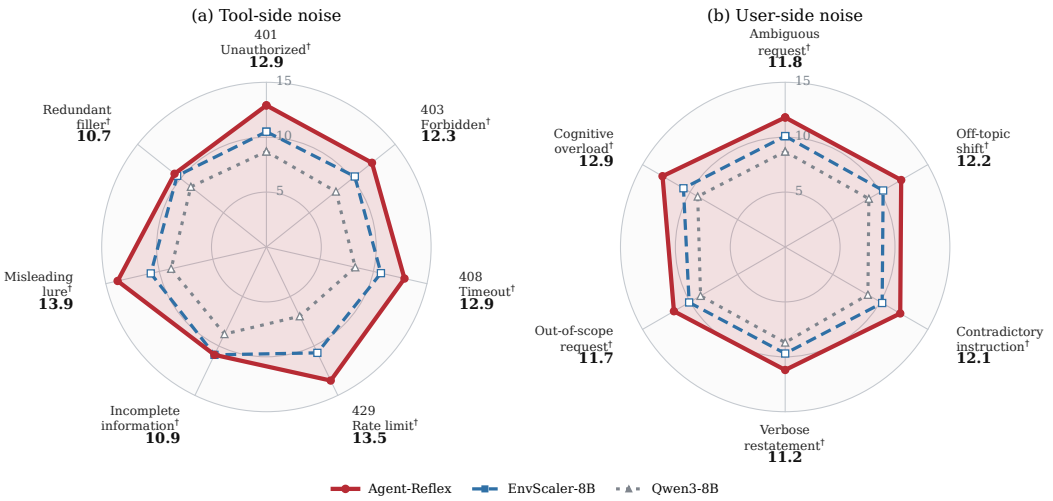


Figure 5: **Per-axis Stability-Gated Success on AgentNoiseBench's native taxonomy**, normalized by the three-model mean on each axis (10 = that average). **(a)** Seven tool-side axes. **(b)** Six user-side. * marks a lead clearing a 95% paired-bootstrap interval against *both* baselines, † the backbone alone.

trajectories, never a record of which well-formed return silently lied. On the four hard-failure templates, where the status code already names the fault, gains over the backbone are uniform (12.3–13.5).

User side. The largest margin is cognitive overload (12.9), the only user-side axis also clearing the interval against the stronger EnvScaler-8B, and the one RESFT's user-reflection supervision targets directly: when the instruction itself is the defect, the repair is to re-scope the request rather than re-plan the tools, a behavior no tool-side reward can teach. Pooled over both sides, AGENT-REFLEX is derailed on 28.5% of rollouts against 44.8% for EnvScaler-8B and 56.1% for the backbone. Because the metric gates on trajectory validity, none of this is reachable by recovering an answer after being derailed: the gains are repair, not tolerance.

E.2 WHY AGENTNOISEBENCH IS OUT OF DISTRIBUTION FOR AGENT-REFLEX

AgentNoiseBench is never consulted while RSD is built: neither its taxonomy nor its injector informs our perturbations, and the two noise designs are constructed in entirely different ways. The noise used by AGENT-REFLEX comes entirely from summarizing large-scale real tool-call failures: the eight families of Table 4 are counted from 24,510 error returns observed while replaying agents on nine public agentic benchmarks, and are injected into tool returns of our own synthesized environments during construction and training (Equation 1). AgentNoiseBench instead derives its categories from its own empirical analysis of interaction noise and uses a separate injector that rewrites tool returns and user utterances of the τ^2-Bench and VitaBench environments at evaluation time, covering seven tool-side and six user-side diagnostic axes. The two therefore differ in their source of noise, in the environments they act on, and in the injection process, and no AgentNoiseBench environment, task, or injector is used anywhere in training. Performance here therefore measures whether a reflective policy transfers to unseen perturbations, not whether a seen noise distribution is recalled.

E.3 REPRESENTATIVE ENVIRONMENT-SPECIFIC SKILLS

To make the mined library concrete, we show five records from substantially different domains in `environment_skills.jsonl`. Each capsule preserves the four fields used by RSD: environment, representative tools, bad pattern, and extracted skill.

AKShare One MCP Server

Environment

Market-data and instrument-registry service. It exposes historical and real-time quotes, news, financial statements, insider-trading records, and instrument metadata, with date, source, pagination, and symbol constraints.

Representative tools

- `resolve_symbol`: map a ticker to a normalized instrument record.
- `list_instruments`: filter and paginate the instrument registry.
- `get_hist_data`: return OHLCV bars with optional adjustment and indicators.
- `get_realtime_data`: return the latest quote snapshot.
- `get_balance_sheet` / `get_cash_flow`: return statement rows.

Bad patterns

- Fail to verify the target instrument and mix similar objects.
- Ignore the requested date, source, count, or field boundary and report an incomplete result as exhaustive.

Extracted skills

- Confirm the resource, object, and requested filter boundary before querying.
- Reconcile every returned row against symbol, range, count, and field types; defaults, summaries, and accepted requests are not evidence of returned data.

DeepResearch

Environment

Stateful research workflow: a query creates a session, iterative steps add notes and citations, and a final report is available only after the session reaches a reportable state. Session, step, and source records are separately addressable and may be paginated.

Representative tools

- `initialize-research`: create a session with a query and target depth.
- `execute-research-step`: advance exactly one research step.
- `get-research-session` / `list-research-steps`: expose status and progress.
- `list-sources`: enumerate discovered sources.
- `generate-report`: compile a report only from a reportable session.

Bad patterns

- Continue reasoning after a key return is missing or a call fails.
- Treat a default parameter or an old session snapshot as the current state, and confuse session, step, or report identifiers.

Extracted skills

- Map each input condition to the required target field and execute the workflow in state order.
- Re-check session, source, and report records after every transition; never fill missing values by guesswork or conflate stage identifiers.

Airtable Server

Environment

Structured-table environment with nested bases, tables, fields, views, and records. Operations use opaque identifiers, cursor pagination, schema constraints, soft deletion, and authorization checks.

Representative tools

- `list_bases`: discover visible bases.
- `list_tables`: enumerate tables within a selected base.
- `list_fields`: verify the table schema.
- `search_records` / `list_records`: return filtered or paginated records.
- `get_record`: read one record by its table-scoped identifier.
- `update_record` / `delete_record`: mutate records under schema and lifecycle constraints.

Bad patterns

- Act on a similar search hit without confirming the target base and table.
- Omit the record identifier or post-write read, so the operation can affect the wrong scope and remain unverifiable.

Extracted skills

- Resolve base, table, and record identifiers in that order and compare key fields with the user's constraints.
- Update only after an exact match, then re-read the record to verify committed values and lifecycle state.

Hotel Booking Server

Environment

Stateful hotel-search and booking workflow. A location query creates a search session, hotel results are paginated within that session, rate details are scoped to a returned hotel, and a booking attempt produces a lifecycle record and payment link.

Representative tools

- `find-place`: resolve free-text locations to coordinates.
- `search-hotels`: start a session with dates, guests, and filters.
- `load-more-hotels`: advance the session cursor.
- `get-search-session`: verify query, pagination, and lifecycle state.
- `get-hotel-details`: expand a hotel returned by the session.
- `book-hotel`: initiate a booking for a currently valid rate.

Bad patterns

- Mix hotels or rates from different search sessions, or treat a paginated summary as complete.
- Initiate booking without checking current availability and whether the rate belongs to the selected hotel.

Extracted skills

- Preserve the session identifier and verify query parameters, pagination, hotel identity, and room/rate constraints at each step.
- Re-read the returned hotel/rate and booking records; treat payment-link creation as an initiated booking, not completed payment.

Semantic Scholar Server

Environment

Academic-literature environment for searching papers and authors and traversing citation/reference edges. Search is truncated, while detailed metadata and citation edges require stable paper or author identifiers and are cached for inspection.

Representative tools

- `search_semantic_scholar`: find candidate papers.
- `get_semantic_scholar_paper_details`: resolve a full paper record.
- `get_semantic_scholar_author_details`: resolve a full author record.
- `get_semantic_scholar_citations_and_references`: retrieve verified citation edges.
- `list_semantic_scholar_papers` / `get_semantic_scholar_search_query`: audit cached records and query status.

Bad patterns

- Fail to keep a unique paper or author identifier attached to the retrieved record, then use a similar or truncated hit for citation statistics.
- Treat a missing result or service error as confirmed metadata, leaving the final claim without verifiable evidence.

Extracted skills

- Verify title, authors, year, DOI, and stable identifier before reading citation edges; retrieve the target record in a second step.
- Cite only fields present in the returned record; if the target is missing or the service fails, mark the claim unconfirmed and refine the query.

F LIMITATIONS

Our reflection skill library is mined offline from a fixed snapshot of a large collection of executable environments. Consequently, it does not automatically adapt to changes in tool interfaces, service behavior, or environment dynamics. Such changes may require periodic re-mining or lightweight maintenance to keep the library aligned with the current environments.

Online updating, however, does not necessarily yield broadly generalizable skills. When the available environments or tasks lack sufficient diversity, skills summarized online may remain specific to individual environments or local failure patterns. The broader environmental coverage used for offline skill discovery may therefore help explain the stronger generalization of AGENT-REFLEX relative to SEED, although isolating this factor would require further controlled experiments. This highlights the importance of balancing update frequency with sufficient environmental coverage for skill generalization.

Future work will investigate online skill summarization and incremental updating across larger and more diverse environment collections, allowing the library to incorporate new environments and emerging failure patterns while supporting cross-environment generalization.

G BROADER IMPACTS AND FUTURE DIRECTIONS

Scope of the current recipe. AGENT-REFLEX is validated on 500 executable environments anchored in live MCP services, eight empirically mined tool-failure families, and six evaluation suites spanning long-horizon tool use and noisy interaction, so the recipe is exercised well beyond a single benchmark. Its three stages are also deliberately modular: the skill library, the reflective supervision, and the token-level objective can each be replaced without touching the others, which makes the extensions below incremental rather than redesigns.

Future directions. Three extensions follow naturally. First, the failure taxonomy is textual, mined from what an agent *sees* when a service fails; extending it to temporal signatures such as latency, concurrency, and partial writes would widen the range of deployment faults a reflective policy can

anticipate. Second, the library is mined once at 8B scale, and testing whether it transfers across capacities, and whether skills compose across environments rather than only within them, would turn it from a per-recipe artifact into a reusable resource. Third, the privileged slot in ORPO accepts any environment-level artifact, so targets richer than reflection rules, verified sub-plans or tool contracts for instance, can be distilled with exactly the same estimator.

Broader impacts. An agent that recovers from failure is more useful and also more consequential when wrong, since it takes more actions in the world than one that halts. Our skill records therefore carry explicit reversibility and partial-effect fields, and recovery templates distinguish asking the user from acting unilaterally. The RESFT user-reflection data deliberately includes incorrect advice so the policy learns evidence-sensitive replanning rather than obedience. All environments are synthesized from public service specifications and contain no real user data, and because training perturbations are mined from real error logs, we release failure counts and families rather than per-service identities.

H ETHICS STATEMENT

This work does not involve any issues related to the Code of Ethics.