HINT-SD: Targeted Hindsight Self-Distillation for Long-Horizon Agents

Anonymous ACL submission

Abstract

Training long-horizon LLM agents with reinforcement learning is challenging because sparse outcome rewards reveal whether a task succeeds, but not which intermediate actions caused the outcome or how they should be corrected. Recent methods alleviate this issue by generating rewards or textual hints from turn-level action-output signals, or by using feedback-conditioned self-distillation. However, generating feedback at every turn is inefficient when many intermediate turns are already successful or neutral, and applying feedback at a fixed or misaligned turn often fails to supervise the actions that contributed to the failure. To bridge this gap, we propose HINT-SD, a targeted self-distillation framework that uses full-trajectory hindsight to select failure-relevant actions and applies feedback-conditioned distillation only to targeted action spans. Experiments on BFCL v3 and AppWorld show that our method improves over the dense per-turn feedback baseline by up to 18.80 percentage points while achieving a 2.26× reduction in time per training step, suggesting that selecting where to distill is a key factor for both effective and efficient long-horizon agent training.

1 Introduction

Large language model (LLM) agents are widely adopted to automate complex workflows through long-horizon interactions with tools, APIs, and web interfaces (Yao et al., 2023; Zhou et al., 2024; Trivedi et al., 2024; Patil et al., 2025). Reinforcement learning (RL) has become an increasingly common post-training paradigm for improving agents from verifiable task results. However, long-horizon agent tasks typically provide only sparse, binary rewards that indicate whether the task succeeded, offering limited guidance on which intermediate decisions contributed to success or failure or how agent behavior should be improved.

Recent work has addressed this sparsity by providing denser learning signals. AgentEvolver (Zhai et al., 2025) augments GRPO with LLM-based self-attribution, assigning process-level contribution rewards to intermediate actions. SDPO (Hübotter et al., 2026) and RLTF (Song et al., 2026) use rich feedback as privileged training context, distilling feedback-conditioned teacher distributions from textual critiques, runtime errors, or failed tests into a feedback-free student. OpenClaw-RL (Wang et al., 2026b) extends this idea to agentic interactions by using next-state signals such as user replies, tool outputs, and environment transitions to generate turn-level rewards and textual hints for on-policy distillation.

While providing denser feedback, these approaches leave its localization unresolved, making corrective supervision highly ineffective for long-horizon agents. For instance, although self-attribution can identify failure-causing actions, its signal remains a scalar reward, making the agent still rely on sparse successful rollouts to learn the correct alternative. Feedback-conditioned distillation provides token-level teacher supervision, but applying hindsight feedback before the first action or distilling the full trajectory can misalign the teacher and student. After the erroneous turn identified by feedback, the student's subsequent trajectory may already deviate from the trajectory supported by the feedback-conditioned teacher, making later token targets unreliable and dominated by accumulated mismatch rather than the intended local correction. OpenClaw-RL localizes feedback to each action, but it must evaluate every turn and remains tied to immediate action-output transitions, making delayed failures difficult to attribute. The remaining bottleneck is therefore not only how to obtain richer feedback, but also how to place it on the action spans where it is relevant.

We view this as a *relevance-sparsity* problem: in a failed trajectory, only a small subset of actions

may require correction, since most turns are either correct or consequences of earlier mistakes. Supervising all turns wastes training budget and can introduce noisy updates. Moreover, in long-horizon interactions, errors often become visible after their causal actions. Applying feedback at the observed failure can therefore supervise downstream consequences instead of the original mistakes. Effective hindsight learning should first identify where feedback is relevant before using it for policy updates.

We propose HINT-SD, a self-distillation framework that converts hindsight feedback into targeted token-level supervision. Given a failed trajectory, HINT-SD analyzes the full rollout to produce a sparse set of failure-relevant steps together with corrective feedback for each step. For each selected step, the same policy serves as a hindsight-conditioned teacher by observing the original prefix plus the generated feedback, while the student observes only the original prefix. HINT-SD then applies a distillation loss only to the selected action spans, encouraging the student to internalize corrective behavior.

Our contributions are as follows: (i) we identify relevance-sparsity as a key obstacle in long-horizon agent training and formulate hindsight distillation as a target-selection problem; (ii) we propose HINT-SD, a self-distillation framework for long-horizon agent training that distills a feedback-conditioned teacher only at selected failure-relevant actions; (iii) we evaluate HINT-SD on BFCL v3 and AppWorld, improving over the dense per-turn feedback baseline by up to 18.80 percentage points with a 2.26× reduction in time per training step.

2 Method

Problem Setup. We consider a multi-turn agent policy π_θ interacting with an environment over a trajectory

$$\tau = (s_1, a_1, \cdots, s_T, a_T), \tag{1}$$

where s_t denotes the environment state observed at step t, which may include tool outputs, error messages, and other interaction feedback. At each step, the agent samples an action $a_t \sim \pi_\theta(\cdot \mid h_t)$ conditioned on the interaction history $h_t = (s_1, a_1, \ldots, s_t)$. Our goal is to improve task outcomes by applying supervision only to action spans responsible for failure while preserving useful behavior elsewhere.

Overview. To address the challenges of sparse trajectory-level rewards and the inefficiency of uniform process supervision, we propose HINT-SD, a targeted self-distillation framework guided by self-generated hindsight. Given a failed rollout, HINT-SD analyzes the complete trajectory in hindsight to isolate a set of failure-relevant actions, and then applies feedback-conditioned self-distillation only to the token spans of these selected actions.

Hindsight Feedback Generation. Identifying the true source of failure is fundamentally challenging in long-horizon trajectories as local evidence can be misleading. A tool call may be syntactically valid and return a plausible observation while encoding an assumption that becomes harmful only several turns later, whereas an incorrect late-stage action may simply be the consequence of an earlier wrong decision. Evaluating each intermediate step in isolation therefore gives an incomplete basis for supervision; reliable attribution requires reasoning over the full sequence of decisions, observations, and the final outcome.

We address this by generating feedback from the *complete* failed rollout. Given a failed trajectory τ, we instantiate the current policy π_θ as a hindsight analyzer $\mathcal{H}_\theta$, prompting it with the task, full trajectory, and instruction (as shown in Figure 4) to return a sparse set of failure-relevant steps together with corresponding corrective feedback:

$$\mathcal{H}_\theta(\tau) \mapsto \{(i, f_i)\}_{i \in \mathcal{I}}, \quad \mathcal{I} \subseteq \{1, \ldots, T\}, \tag{2}$$

where $\mathcal{I}$ denotes the set of selected failure-relevant steps, and f_i is natural-language feedback describing why the action at step i contributes to the failure and how it should have been corrected. Because the selection is made with global trajectory context, it can avoid supervising turns that are locally noisy or merely consequences of the root cause. The feedback generation stage therefore serves two roles at once: it produces corrective feedback and determines the target spans to which the correction should be applied.

Targeted Self-Distillation. With the selected failure-relevant steps $\mathcal{I}$, the remaining challenge is how to apply each correction to its corresponding action span without updating unrelated parts of the rollout. We resolve this by exploiting the information asymmetry inherent in *self-distillation*. Rather than relying on an external supervisor or uniform reward signals, we leverage the policy itself as a localized expert by exposing it to hindsight

feedback. For each identified step $i \in \mathcal{I}$, we augment the original interaction history h_i with the generated feedback f_i, and query the teacher policy under this augmented context. Conditioning the policy on this privileged hindsight induces a locally improved teacher distribution, $\pi_\theta(\cdot \mid h_i, f_i, a_{i,<t})$, while the student remains conditioned only on the original history. We then minimize the reverse KL divergence between the two distributions only on the identified failure-relevant action spans:

$$\sum_{i \in \mathcal{I}} \sum_{t=1}^{|a_i|} D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_i, a_{i,<t}) \,\|\, \mathtt{sg}(\pi_\theta(\cdot \mid h_i, f_i, a_{i,<t}))\big), \tag{3}$$

where $\mathtt{sg}(\cdot)$ denotes the stop-gradient. By deliberately narrowing the optimization landscape to these precise regions, the policy is encouraged to absorb high-quality feedback exactly where it erred. This targeted mechanism effectively provides dense, token-level supervision for long-horizon tasks with sparse rewards, enhancing learning efficiency while preserving original task performance by avoiding unnecessary updates to successful trajectories.

3 Experiments

3.1 Experimental Setup

Benchmarks & Metrics. We evaluate HINT-SD on two complementary long-horizon agent benchmarks: BFCL v3 (Patil et al., 2025) and AppWorld (Trivedi et al., 2024). BFCL v3 evaluates executable multi-turn function calling under schema and dialogue constraints; we use only the BASE and LONG CONTEXT categories from the multi-turn split. AppWorld evaluates stateful application workflows through Task Goal Completion, where agents interact with app APIs and are scored by unit tests over the final environment state. We run each task four times and report Avg@4 and Best@4.

Baselines. We compare HINT-SD against five baselines. **Initial** denotes the zero-shot policy before any intervention. **SFT** performs supervised fine-tuning on high-reward trajectories generated by GPT-5.4-mini (OpenAI, 2026). **GRPO** (Shao et al., 2024) optimizes terminal task rewards (without textual feedback) under the same rollout budget. **SDPO** (Hübotter et al., 2026) conditions the teacher on hindsight feedback but distills the entire failed trajectory without target-turn selection. **OpenClaw-RL** (Wang et al., 2026b) uses next-state signals to derive scalar rewards and textual hints at each turn, testing dense local feedback without full-trajectory hindsight attribution. We report two variants of HINT-SD: **HINT-SD-Single**, which distills the first failure-relevant step, and **HINT-SD-Multi**, which distills multiple selected failure-relevant steps.

	BFCL v3		AppWorld	
Method	Avg@4	Best@4	Avg@4	Best@4
Initial	25.94	36.25	5.98	13.85
SFT	28.44	38.13	6.82	13.16
GRPO	31.56	41.25	7.49	15.21
SDPO	30.78	40.00	9.74	19.32
OpenClaw-RL	28.28	45.00	7.65	12.31
HINT-SD-Single	36.25	43.13	16.54	29.40
HINT-SD-Multi	**41.88**	**48.75**	**18.46**	**31.11**

Table 1: Comparison of HINT-SD and baselines on BFCL v3 and AppWorld. Best results are in **bold**.

Implementation Details. We evaluate HINT-SD with Qwen3-4B-Instruct-2507 (Yang et al., 2025) as the backbone model. Across all rollout-based optimization methods, we use four rollouts per task and train for 15 epochs. For HINT-SD, we restrict hindsight feedback generation to at most three failure-relevant steps per failed trajectory. Additional details are provided in Section A.

3.2 Experimental Results

Main Results. Table 1 shows that HINT-SD-Multi achieves the best overall performance on both BFCL v3 and AppWorld. On BFCL v3, it improves Avg@4 from the strongest baseline score of 31.56 to 41.88 and Best@4 from 45.00 to 48.75. On AppWorld, it improves Avg@4 from 9.74 to 18.46 and Best@4 from 19.32 to 31.11. The baseline trends highlight the role of localization: GRPO improves on the initial policy but remains limited by sparse terminal rewards, full-trajectory SDPO benefits from hindsight feedback but can dilute corrective supervision across irrelevant or already-correct actions, and OpenClaw-RL achieves competitive Best@4 on BFCL v3 but lower Avg@4, suggesting that dense local hints are less stable across samples. In contrast, HINT-SD localizes feedback-conditioned distillation to failure-relevant turns. Even HINT-SD-Single, which distills only the first failure-relevant step, shows substantial gains over the baselines, demonstrating the efficacy of localized hindsight supervision. The additional gains from HINT-SD-Multi suggest that supervising multiple selected failure points extracts a richer corrective signal from the same rollout budget.

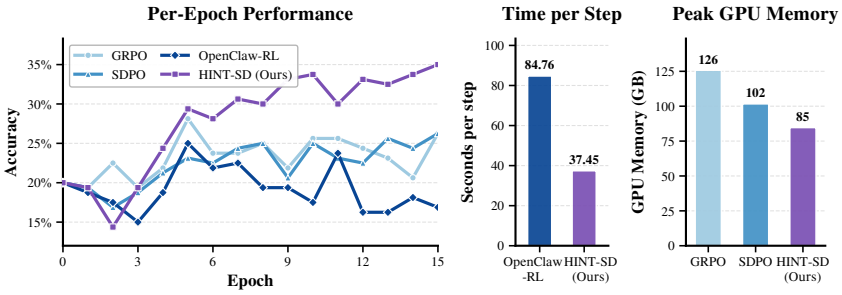

Figure 1: (Left) Per-epoch accuracy scores on the BFCL v3 eval split. (Middle) Time per training step. (Right) Peak GPU memory usage during the first epoch.

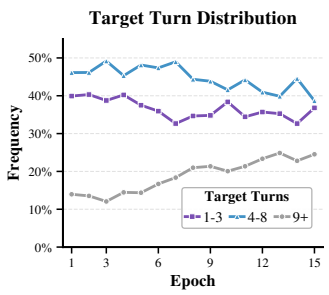

Figure 2: Distribution of selected target turns for BFCL v3.

Benchmark	Start FB	Target FB	Δ(Target–Start)
BFCL v3	2.68	8.67	+5.99
AppWorld	0.44	2.16	+1.72

Table 2: Comparison of feedback placements, showing percentage-point gains vs. the no-feedback rollout.

	BFCL v3		AppWorld	
Feedback Source	Avg@4	Best@4	Avg@4	Best@4
Teacher (w/EMA)	41.88	48.75	18.46	31.11
Environment	36.25	42.50	15.90	27.86
Initial Teacher	37.50	45.63	14.40	28.89
Larger Teacher	**48.59**	**52.50**	**20.81**	**35.04**

Table 3: Comparison of different feedback sources.

Training Dynamics and Efficiency. In addition to effectively providing dense supervision for relevant actions, HINT-SD is significantly more efficient than approaches that either supervise the entire trajectory or rely on per-step feedback or rewards. Figure 1 (Left) shows that HINT-SD improves more rapidly and reaches the highest evaluation accuracy across training epochs, whereas GRPO and SDPO saturate at lower accuracies and OpenClaw-RL exhibits weaker stability. At the same time, because HINT-SD supervises only selected turns rather than every action or the full trajectory, it avoids much of the feedback-generation and distillation overhead of dense-feedback methods. Figure 1 (Middle, Right) shows that this localization reduces time per training step from 84.76s to 37.45s and peak GPU memory from 126GB to 85GB, yielding a 2.26× lower step time and a 1.48× lower memory footprint than the strongest dense-feedback baseline. By limiting distillation to failure-relevant actions, HINT-SD improves optimization while substantially reducing training cost.

Analysis on Target Turn Distribution. Figure 2 shows that feedback targets are distributed across the trajectory rather than concentrated at the beginning. Across the first 15 epochs, 36.7% of targets fall in turns 1–3, 44.8% in turns 4–8, and 18.5% in turn 9 or later. Notably, later targets (9+) increase from 14.0% to 24.5% over training, suggesting that feedback shifts toward later-stage corrections as early-stage errors are reduced. Since these corrections are often distant from the initial prompt, this motivates targeting feedback to the selected failure-relevant turn rather than treating it as a global trajectory-level hint.

Analysis on Feedback Placement. We examine whether applying hindsight feedback at the selected target turn improves rollout success. For each failed base-policy trajectory, the hindsight analyzer produces one feedback message and a target turn. We then run paired interventions with the same feedback: feedback is either inserted at the beginning of a fresh rollout or immediately before the target action after replaying the failed prefix. Each condition is compared against its no-feedback rollout, and feedback remains persistent in the context. Table 2 shows that target-turn feedback yields larger success gains on both benchmarks, with Target – Start gains of +5.99 points on BFCL v3 and +1.72 points on AppWorld. This suggests that the selected target turns are actionable and provide a stronger feedback-conditioned teacher signal than applying the same feedback globally from the start.

Analysis on Feedback Source. To further analyze how the source of hindsight feedback affects HINT-SD, we compare different feedback sources in Table 3. Environmental feedback uses environment outputs directly rather than generating hindsight feedback, but it underperforms teacher-generated feedback variants. The EMA-updated teacher also consistently outperforms the fixed initial teacher, indicating that feedback generation benefits from tracking the improving policy. A larger teacher (GPT-5.4-mini) further improves performance, suggesting that stronger feedback can provide additional gains. Nevertheless, our EMA-updated teacher yields strong results without relying on an external large model, supporting the self-contained design of HINT-SD.

Limitations

While HINT-SD effectively generates hindsight feedback for targeted self-distillation, its training signal still depends on whether the generated feedback correctly identifies actionable failures and proposes corrections that improve task completion. This requires the initial model to have sufficient instruction-following and task-solving capability to reason about failed trajectories. Nevertheless, our results with Qwen3-4B-Instruct-2507 show that a small model can serve as an effective feedback generator. Future work can further guide feedback generation with additional supervision or constraints to improve feedback quality.

Ethical Considerations

HINT-SD leverages the model itself as both a hindsight feedback generator and a feedback-conditioned teacher, enabling the agent to improve through self-generated feedback without relying on stronger external supervision. By focusing hindsight feedback on failure-relevant action spans rather than every step in a trajectory, the approach can make long-horizon agent training more efficient and accessible. However, when connected to real-world tools, the agent may execute incorrect, unauthorized, or harmful actions. Because HINT-SD relies on the model's own feedback signals, such harmful behaviors may also reinforce existing biases or failure modes, highlighting the need for appropriate safeguards before real-world deployment.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2026. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jonas Hübotter, Frederike Lübeck, Lejs Deen Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. 2026. Reinforcement learning via self-distillation. In *The 1st Workshop on Scaling Post-training for LLMs*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

OpenAI. 2026. Introducing gpt-5.4 mini and nano.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yuda Song, Lili Chen, Fahim Tajwar, Rémi Munos, Deepak Pathak, Drew Bagnell, Aarti Singh, and Andrea Zanette. 2026. Expanding the capabilities of reinforcement learning via text feedback. In *The 1st Workshop on Scaling Post-training for LLMs*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. 2026. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1195–1204, Red Hook, NY, USA. Curran Associates Inc.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, Bangkok, Thailand. Association for Computational Linguistics.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. TRL: Transformers Reinforcement Learning.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and 1 others. 2026a. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. 2026b. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Junkeun Yi, Damon Mosk-Aoyama, Baihe Huang, Ritu Gala, Charles Wang, Sugam Dipak Devare, Khushi Bhardwaj, Abhibha Gupta, Oleksii Kuchaiev, Jiantao Jiao, and 1 others. 2026. Pivotrl: High accuracy agentic post-training at low compute cost. *arXiv preprint arXiv:2603.21383*.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, and 1 others. 2025. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*.

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

A Additional Experimental Details

A.1 Additional Details on Baselines

SFT. The SFT baseline is trained on demonstrations generated by GPT-5.4-mini. For each training task, we sample up to 10 candidate trajectories from the teacher model and execute them in the corresponding environment. If at least one candidate satisfies the benchmark-specific success criterion, we retain a successful trajectory as the demonstration. Otherwise, we select the trajectory with the highest reward, providing the strongest available supervision for that task. We then fine-tune the student model using step-wise teacher forcing. At each interaction step t, the student conditions on the trajectory prefix h_t and is trained to predict the corresponding assistant action a_t.

GRPO. The GRPO (Shao et al., 2024) baseline follows the standard GRPO objective, where rewards are computed solely from the final environment state. Because intermediate interaction steps cannot be directly evaluated or assigned step-level rewards, we optimize a trajectory-level objective using the global terminal reward. Concretely, the same scalar reward is propagated uniformly across all action spans $a_1, \ldots, a_T$ within the trajectory.

SDPO. Although SDPO (Hübotter et al., 2026) was originally proposed for single-turn settings, we extend it to the multi-turn setting. After rollout, we use a teacher model to generate natural-language feedback conditioned on the final trajectory and outcome. This global feedback is then prepended to the initial prompt as privileged context, and the feedback-conditioned teacher distills all assistant action spans in the failed trajectory without target-turn selection.

OpenClaw-RL. OpenClaw-RL (Wang et al., 2026b) trains agents from next-state signals observed immediately after each action. A judge converts each next state into an evaluative scalar reward and textual feedback. The reward is used for policy optimization, while the feedback is inserted only into the teacher context for on-policy distillation. Unlike HINT-SD, OpenClaw-RL provides dense local supervision rather than using full-trajectory hindsight.

A.2 Additional Implementation Details

For BFCL v3, we split the full task set into train/eval/test partitions with a ratio of 5:1:4. For optimization, we use AdamW (Loshchilov and Hutter, 2019) with a base learning rate of 5×10^{-6} for BFCL v3 and 3×10^{-6} for AppWorld, together with a linear scheduler and warm-up over the first 5% of training steps. We apply LoRA (Hu et al., 2022) to the query and value projection layers with rank $r = 32$, scaling factor $\alpha = 64$, and dropout rate (Srivastava et al., 2014) of 0.05. Teacher parameters are initialized from the student and updated via EMA (Tarvainen and Valpola, 2017) with an update rate of 0.001. We select the checkpoint with the highest reward. We implement all optimization-based methods with TRL (von Werra et al., 2020) and use vLLM (Kwon et al., 2023) for efficient on-policy generation. All experiments are conducted on a single NVIDIA H200 GPU. The feedback-generation prompts for the HINT-SD-Multi and HINT-SD-Single variants are shown in Figures 4 and 5.

B Additional Experimental Results

B.1 Distribution of Selected Hindsight Targets

Figure 3 aggregates selected hindsight target turns, complementing the epoch-wise regions in Figure 2. Targets concentrate in early and middle interactions (mean turn 5.32), but are not restricted to the start: 10.0% occur after turn 10. This supports the relevance-sparsity view behind HINT-SD: hindsight supervision should be attached to selected failure-relevant turns rather than uniformly to the full trajectory or always from the beginning.

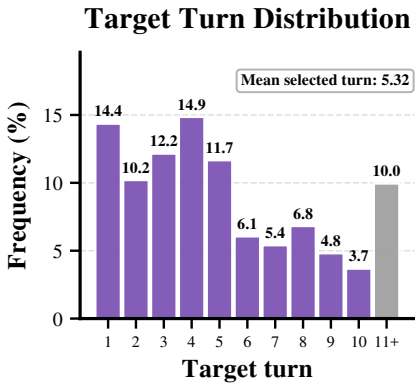

Figure 3: Count distribution of selected hindsight target turns on BFCL v3 over the first 15 training epochs. Turns after 10 are aggregated into the 11+ bin.

B.2 Qualitative Example of Targeted Hindsight Feedback

Figure 6 illustrates how target-turn selection makes hindsight feedback action-specific: rather than giv-

ing a generic episode-level summary, the analyzer specifies how each selected action should change. For example, it identifies the missing Spotify access token at Turn 14 and the undefined token variable at Turn 15, making the feedback directly actionable at the selected action spans. Figures 7 and 8 further compare global hindsight feedback with HINT-SD multi-step feedback, showing that global feedback summarizes the episode-level failure while selected-turn feedback attaches corrections to the concrete actions where the failure becomes actionable.

C Related Work

Credit assignment and selective training. Long-horizon agent training requires assigning sparse outcome signals to intermediate decisions. Verifier and process-supervision works (Cobbe et al., 2021; Lightman et al., 2024) show that intermediate labels can be more informative than final outcomes alone. AgentEvolver (Zhai et al., 2025) uses LLM-based self-attribution to score each action's contribution to the final outcome and converts these scores into process-level rewards for GRPO optimization. Other long-horizon methods select or reweight informative states. PivotRL (Yi et al., 2026) trains on informative pivots from expert trajectories, GiGPO (Feng et al., 2026) targets fine-grained credit assignment in group-based RL, and HCAPO (Tan et al., 2026) uses hindsight reasoning to refine step-level credit for policy optimization. These methods can identify failure-causing actions, but the resulting signal is typically scalar or policy-gradient based, so learning the correct alternative action still depends on sparse successful rollouts.

Feedback-Conditioned Distillation. Natural-language and environment feedback has been used to revise model outputs and provide privileged training signals. Reflexion (Shinn et al., 2023), Self-Refine (Madaan et al., 2023), and CRITIC (Gou et al., 2024) use verbal or tool-grounded feedback for iterative correction. SDPO (Hübotter et al., 2026) and RLTF (Song et al., 2026) instead internalize such feedback by distilling feedback-conditioned behavior into a feedback-free policy. In agent settings, OpenClaw-RL (Wang et al., 2026b) converts next-state signals into turn-level rewards or textual hints, and Skill-SD (Wang et al., 2026a) conditions the teacher on retrieved skill descriptions. However, these methods either treat feedback as trajectory-level supervision or analyze feedback at every turn, leaving open which agent action should receive the corrective signal and introducing unnecessary cost when most turns are already correct or irrelevant. HINT-SD instead uses full-trajectory hindsight to select failure-relevant action spans and applies feedback-conditioned distillation only at those targeted turns.

```
SYSTEM:
You analyze failed AppWorld tool-use trajectories. Identify up to {
max_steps} problematic steps where the agent made mistakes (between 1 and
{max_steps} steps total, ordered earliest first). For EACH problematic
step, write a distinct correction in less than three sentences that
targets only that step's mistake. Output valid JSON only, no other text.

USER:
Review this failed AppWorld attempt and identify the problematic steps.
Return one (step, feedback) pair per problematic step.

{trajectory}

Output format (JSON only, no other text):
{"failures": [{"step": <1-indexed step number>, "feedback": "<correction
for this step>"}, ...]}
```

Figure 4: Prompt template for multi-step hindsight feedback generation in HINT-SD-Multi. Given a complete failed AppWorld trajectory, the analyzer selects up to {max_steps} failure-relevant steps and returns localized corrective feedback for each selected step.

```
SYSTEM:
You analyze failed AppWorld tool-use trajectories. Identify the FIRST step
 where the agent made a mistake. Write the feedback in less than three
sentences. Output valid JSON only, no other text.

USER:
Review this failed AppWorld attempt and identify the first problematic
step.

{trajectory}

Output format (JSON only, no other text):
{"failure_step": <1-indexed step number>, "feedback": "<correction
sentence(s)>"}
```

Figure 5: Prompt template for single-step hindsight feedback generation in HINT-SD-Single. Given a complete failed AppWorld trajectory, the analyzer identifies the earliest failure-relevant step and returns a concise correction for the step.

Task

Start playing a playlist on Spotify that has enough songs for my workout today. I do not want to have to change the playlist in the middle of my workout. The workout plan is in Simple Note.

Abbreviated Execution Trajectory

```
[Turns 1-13 (Summarized)]
The agent retrieves account information, logs into Simple Note, searches for the workout note, and
parses the workout duration for the current day.  It then moves to Spotify and obtains a login
response stored as login_result.

[Selected Turn 14 Action]
playlists = apis.spotify.search_playlists(query="workout")

[Observation]
Exception: Response status code is 401:
{"message": "You are either not authorized to access this spotify API endpoint
or your access token is missing, invalid or expired."}

[Selected Turn 15 Action]
playlists = apis.spotify.search_playlists(
query="workout",
access_token=spotify_access_token["access_token"]
)

[Observation]
NameError: name 'spotify_access_token' is not defined

[Turns 16-20 (Summarized)]
The rollout continues after the invalid Spotify state, but no valid playlist playback is completed
before termination.
```

Selected Hindsight Feedback

Turn 14. The API call is missing the access token; add it as a parameter in the `search_playlists` call.
Turn 15. The variable `spotify_access_token` is undefined; use `login_result` from the previous step instead.

Figure 6: Qualitative example of selected hindsight target turns from an AppWorld training rollout. The abbreviated trajectory context shows that the analyzer localizes feedback to the actions where the agent loses the authenticated Spotify state, rather than applying the same feedback globally at the beginning of the trajectory.

Task

BFCL travel booking. Book an economy flight from Crescent Hollow to New York on November 12, 2026 using the user's card and access token. The conversation then asks for travel insurance, an invoice, and customer-support handling for the same booking.

Abbreviated Execution Trajectory

```
[Turns 1-2 (Summarized)]
The agent maps Crescent Hollow to CRH and New York to JFK.

[Turn 3 Action]
get_flight_cost(
travel_from="CRH",
travel_to="JFK",
travel_date="2026-11-12",
travel_class="economy"
)
[Observation]
{"travel_cost_list": [300.0]}

[Selected Turn 4 Action]
The assistant explains the mismatch between the expected $850 fee and the observed $300 cost, asks
for confirmation, and does not call book_flight.
[Next User]
The user refers to the trip as post-booking and asks to purchase $100 travel insurance.

[Selected Turn 5 Action]
purchase_insurance(
access_token=...,
booking_id="bk_12345",
card_id="card_6789",
insurance_cost=100
)
[Observation]
{"insurance_status": false, "error": "Booking not found"}

[Selected Turn 7 Action]
retrieve_invoice(
access_token=...,
booking_id="bk_12345",
insurance_id="ins_98765"
)
[Observation]
{"error": "Booking not found"}
```

Feedback Comparison

Global hindsight. The episode-level feedback identifies one root cause: the assistant never created a valid booking, so the later insurance, invoice, and support calls all used a non-existent booking id. It recommends calling `book_flight` before any booking-dependent action.
HINT-SD multi-step. Turn 4: initiate the booking with `book_flight`. **Turn 5:** do not buy insurance with the fabricated booking id `bk_12345`. **Turn 7:** retrieve the invoice only after a successful booking creates a valid booking id.

Figure 7: Qualitative comparison on a BFCL task. The task-matched rollouts share the same failure pattern: the booking is never created, so later booking-dependent tool calls fail. Global hindsight gives one episode-level correction, while HINT-SD attaches the same root cause to the concrete turns where it first appears and then propagates.

Task

AppWorld file system. Compress each vacation-spot subdirectory under `~/photos/` into a tar file under `~/photos/vacations/`, then delete the original vacation-spot subdirectories.

Abbreviated Execution Trajectory

```
[Selected Turn 1 Action]
import os
import tarfile
Use local filesystem calls to list vacation folders under ~/photos/.
[Observation]
Usage of the following module is not allowed: tarfile.
[Turns 2-4 (Summarized)]
The agent inspects available apps and file_system API descriptions, which list show_directory rather
than list_directory.
[Selected Turn 3 Action]
vacation_spots = apis.file_system.list_directory("~/photos/")
[Observation]
Exception: No API named list_directory found in the file_system app.
[Selected Turn 5 Action]
vacation_spots = apis.file_system.show_directory("~/photos/")
[Observation]
TypeError: positional argument error. The API specification later shows that show_directory expects
named arguments, including access_token.
```

Feedback Comparison

Global hindsight. The task-matched episode feedback focuses on a later login failure and recommends using the full email address when calling `apis.file_system.login`.
HINT-SD multi-step. **Turn 1:** avoid disallowed system modules and use environment APIs. **Turn 3:** replace `list_directory` with the documented `show_directory` API. **Turn 5:** authenticate first and pass `access_token` as a named parameter.

Figure 8: Qualitative comparison on an AppWorld task. The trajectory excerpt is from the selected-target run. The selected-turn feedback exposes early actionable errors in API use and authentication, instead of only summarizing a later episode-level failure.