LEARNING WHEN AND HOW TO INTERVENE: A HINDSIGHT-DISTILLED SENTINEL FOR CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents solve repository-level tasks through sequences of actions, where a single erroneous action can misdirect subsequent decisions and increase recovery costs. Existing approaches use execution feedback for recovery or specialized checks to block errors, but deciding before execution whether intervention will benefit eventual task completion remains challenging. To address this challenge, we propose HISENTINEL, a hindsight-distillation framework that trains lightweight 0.6B and 1.7B sentinels to select pre-execution interventions aimed at improving task completion rather than correcting every imperfect action. A privileged teacher uses recorded execution outcomes as evidence for intervention judgments, which are distilled into a causal student that receives only the pre-action context and proposed action. Beyond identifying whether and when to intervene, the sentinel must also provide actionable feedback that helps the coding agent recover or obtain necessary human input. To support these capabilities, we introduce SWE-INTERVENE, an action-level dataset constructed from software-engineering trajectories that annotates whether an action should be allowed, autonomously redirected, or paused for human assistance, together with corresponding intervention feedback. Across SWE-bench Verified Mini and Ask or Assume, HISENTINEL consistently improves task completion across Sentinel scales and coding-agent families, with gains of up to 14% and 10%, respectively, while maintaining competitive token consumption. These results demonstrate that lightweight pre-execution intervention can effectively prevent error propagation and improve the reliability of autonomous coding agents. All datasets and code are available at `https://anonymous.4open.science/r/Hisentinel/`.

1 INTRODUCTION

Large language models now power coding agents that solve repository-level software-engineering tasks by navigating codebases, editing files, executing commands, and running tests (Jimenez et al., 2024; Yang et al., 2024; Wang et al., 2025). Because these agents operate through multi-step interaction loops, a poor action can alter the environment, mislead subsequent decisions, and propagate into failed repairs, redundant exploration, or ultimately task failure (Chen et al., 2025; Majgaonkar et al., 2025; Gandhi et al., 2025). Critically, an early mistake may remain latent for several steps and become apparent only after recovery has become difficult or impossible (Zhao et al., 2026c).

Evidence from Terminal-Bench highlights this gap: among 1,184 failed CLI-agent trajectories, half committed their decisive error by step 7 and became unrecoverable around step 12, while the first observable failure appeared only around step 16; prefix monitoring detected just 3.7–8.7% of failures before lock-in (Merrill et al., 2026; Zhao et al., 2026c). Existing process critics review recent steps and provide periodic corrective feedback (Gandhi et al., 2025), while trajectory diagnosis can locate decisive errors after a run (Wang et al., 2026; Zhang et al., 2026c). Yet feedback may arrive after a harmful action has executed, and retrospective diagnosis cannot prevent it. This leaves open how to judge an action when it is first proposed.

As illustrated in Figure 1, this motivates moving intervention earlier: after the agent proposes an action but before the environment executes it. At this boundary, a Sentinel can prevent a harmful

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.

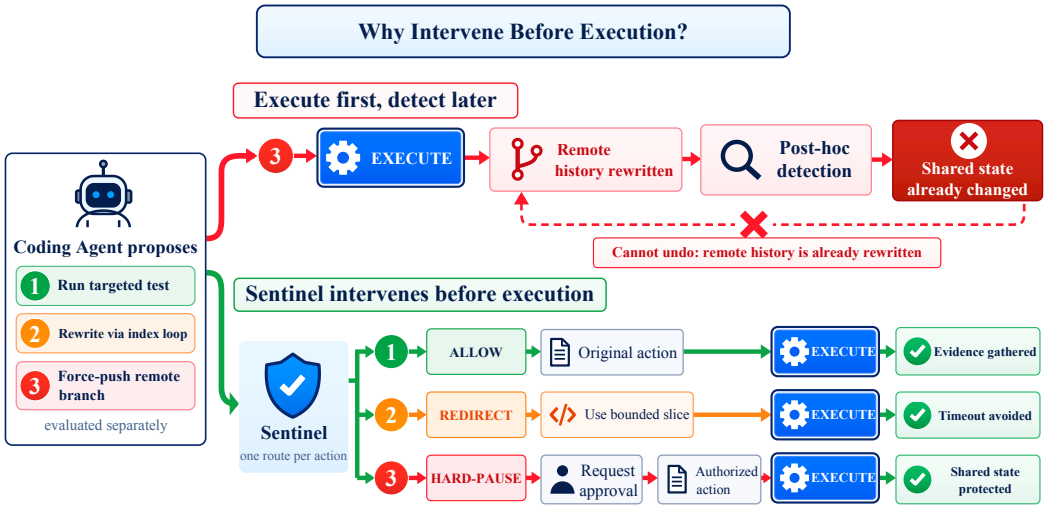


Figure 1: Pre-execution intervention for coding agents. The Sentinel evaluates each proposed action before execution and selects ALLOW, REDIRECT, or HARD-PAUSE.

action from changing the environment, rather than attempting to recover from its consequences afterward. We formulate this as a three-way decision among ALLOW, REDIRECT, and HARD-PAUSE: allowing appropriate actions, redirecting errors the agent can correct autonomously, and pausing when progress requires information, authorization, or access available only from a human.

However, moving intervention before execution creates a fundamental information gap. The execution result provides the most direct evidence of whether a proposed action is appropriate, yet this evidence is inherently unavailable at the time of intervention. The Sentinel must therefore anticipate whether an action threatens eventual task completion using only the pre-execution context and the proposed action.

To bridge this gap, we propose HISENTINEL, a hindsight-distilled framework that transfers knowledge available after execution into a model that acts before execution. During training, a privileged teacher observes both the proposed action and its recorded execution result, allowing it to judge the action using evidence of what actually occurred. Its judgments are then distilled into a lightweight causal Sentinel that receives only the pre-execution context and proposed action. At deployment, the Sentinel can therefore intervene without access to future observations while benefiting from supervision informed by execution hindsight.

Beyond deciding whether to intervene, the Sentinel must communicate how the agent should proceed. We therefore optimize feedback generation separately, training the Sentinel to provide concise, evidence-grounded guidance that suggests a minimal correction or requests the specific human input needed to continue.

To train both intervention decisions and feedback, we introduce SWE-INTERVENE, an action-level dataset constructed by selecting decision nodes from natural software-engineering trajectories. Each node preserves the task, complete pre-action context, and exact proposed action, and is annotated as a three-way decision.

We evaluate HISENTINEL at two levels: intervention decisions and feedback quality on frozen in-domain tests and external trajectory-diagnosis benchmarks, and end-to-end performance with coding agents on SWE-BENCH VERIFIED MINI and ASK-OR-ASSUME. We measure task completion, unnecessary interventions, and inference overhead (Hobbhahn, 2025; Edwards & Schuster, 2026).

In summary, our contributions are as follows:

- We formulate real-time intervention at the boundary between action proposal and execution as a three-way decision among ALLOW, REDIRECT, and HARD-PAUSE, targeting eventual task completion rather than merely detecting imperfect actions.
- We introduce SWE-INTERVENE, an action-level dataset for learning when and how to intervene in natural software-engineering trajectories.

- We propose HISENTINEL, which distills execution hindsight from a privileged teacher into a lightweight pre-execution Sentinel and trains it to generate concise corrective feedback that guides subsequent agent behavior.

2 RELATED WORK

Trajectory-based Agent Analysis. Agent evaluation has evolved from final task success and progress-based metrics in AGENTBOARD (Ma et al., 2024) to trace-level failure attribution and localization in WHO&WHEN and TRAIL (Zhang et al., 2025b; Deshpande et al., 2025). Recent methods further analyze realistic trajectories through learned or structured diagnosis, including AGENTRACER, TRAJAUDIT, FALAT, TRAJDEBUG, and LONGRCA (Zhang et al., 2026a; Ou et al., 2025; Liu et al., 2026; Barke et al., 2026; Wang et al., 2026; Rafi et al., 2026; Shu et al., 2026; Zhao et al., 2026c; Qi et al., 2026; Zhang et al., 2026c). Unlike these primarily diagnostic approaches, HISENTINEL asks whether intervention at the current node has sufficient evidence to improve eventual task completion over autonomous recovery.

Learning with Privileged Information. Learning with privileged information transfers training-only evidence from an informed teacher to a restricted student (Vapnik & Izmailov, 2015; Chen et al., 2020; Cai et al., 2024). Recent work extends this paradigm to LLMs: LEAP uses privileged environment states to provide corrective feedback for LLM agents (Choudhury & Sodhi, 2025), while OPSD conditions a self-teacher on verified solutions and π-DISTILL jointly optimizes privileged and unprivileged policies (Zhao et al., 2026b; Penaloza et al., 2026). Related methods exploit future-conditioned, short-context, or trajectory-derived privilege (Fang et al., 2026; Zhang et al., 2026b; Dat et al., 2026; Chen et al., 2026), although privileged context can be harmful when unrealizable from the student view (Kaur et al., 2026; Shrestha & Tessier, 2026). In contrast, HISENTINEL uses realized software-engineering continuations as privileged evidence for intervention utility, rather than supervising task actions or reasoning trajectories.

3 SWE-INTERVENE: DATASET CONSTRUCTION

SWE-INTERVENE draws action-level instances from OPEN-SWE-TRACES, SWE-HERO, and SWE-CHAT (Ahmad et al., 2026; Ludwig et al., 2026; Baumann et al., 2026). Using human-defined guidelines, GPT-5.5 (OpenAI, 2026) labels decision points as ALLOW, REDIRECT, or HARD-PAUSE; SWE-CHAT `AskUserQuestion` (AUQ) interactions supply natural HARD-PAUSE cases. A two-stage human audit rejects 25.3% of intervention candidates. The resulting data comprise 5,680 training and 1,243 held-out test instances (82.0%/18.0%); the test set includes 1,122 natural and 121 AUQ instances. We de-duplicate training data against every downstream benchmark in Section 5.1 using instance identities and normalized task, repository, and trajectory signatures. Appendix A.1 details the sources, audit, and de-duplication.

4 HISENTINEL

Figure 2 presents the overall framework of HISENTINEL. During training, a future-aware teacher uses recorded outcomes to provide privileged intervention supervision, which is distilled into a lightweight causal sentinel, while AUQ-derived samples supply supervision for HARD-PAUSE. A separate feedback adapter is optimized to generate actionable guidance or human-assistance requests. At deployment, the sentinel observes only the trajectory prefix and proposed action, and intervenes before execution when necessary.

4.1 PROBLEM FORMULATION

We consider a coding agent solving a software-engineering task u through a sequence of actions and environment observations. At step t, the agent has observed a trajectory prefix h_t and proposes an action a_t. Before the action is executed, HISENTINEL evaluates the pre-action context

$$x_t = (u, h_t, a_t), \tag{1}$$

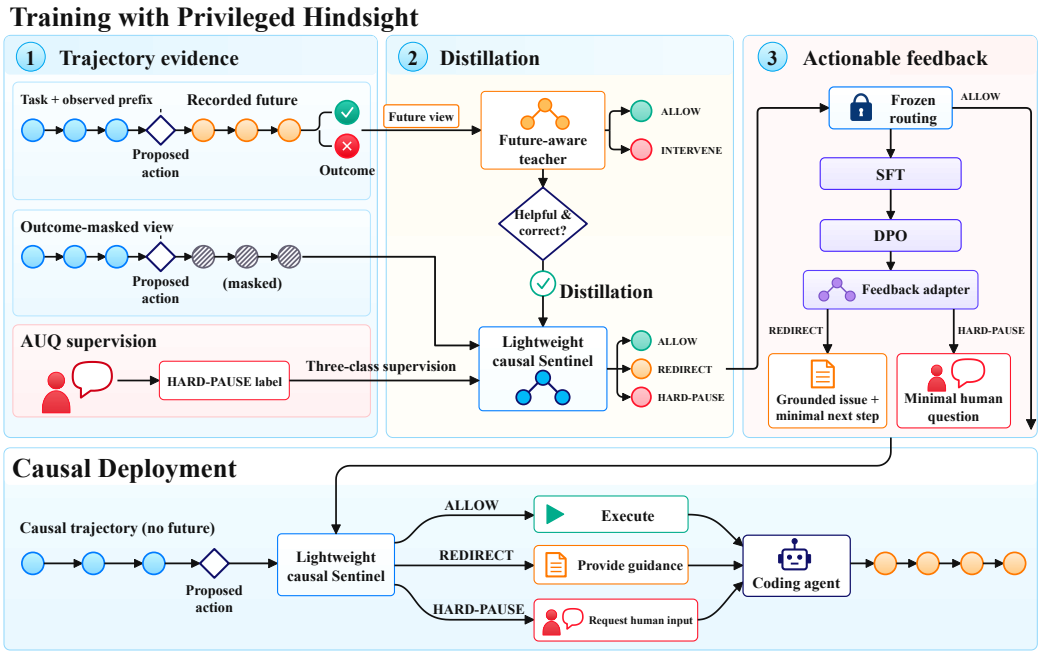


Figure 2: Overview of HISENTINEL. Privileged hindsight is used only during training, while the deployed causal sentinel selects an intervention before the proposed action is executed.

and returns an intervention route $\hat{y}_t$ together with feedback r_t when intervention is required. The route and feedback determine how the coding agent proceeds. Our objective is to intervene when there is sufficient reason to expect an improvement in eventual task completion relative to autonomous continuation, rather than to correct every locally imperfect action.

Table 1: Intervention actions used by HISENTINEL.

Action	Operational definition
ALLOW	Execute the proposed action without intervention because there is insufficient evidence that intervening would improve task completion.
REDIRECT	Withhold the proposed action and provide corrective feedback, prompting the agent to revise its plan and propose a new action autonomously.
HARD-PAUSE	Suspend execution and request missing information, authorization, or preferences from a human before the agent continues.

The three routes induce different runtime control flows. Under ALLOW, $r_t = \varnothing$ and a_t is executed unchanged. Under REDIRECT, r_t provides corrective guidance; the original action is withheld, and the agent proposes a revised action conditioned on the feedback. Under HARD-PAUSE, r_t specifies a question for the human, and execution remains suspended until the answer is returned to the agent.

For training, let e_t denote the retrospective evidence available at a decision point, such as its recorded continuation and task outcome or an observed agent–human interaction. Following the annotation criteria in Section 3, this evidence informs the intervention label

$$y_t = \mathcal{A}(x_t, e_t) \in \mathcal{Y}, \qquad \mathcal{Y} = \{\text{ALLOW}, \text{REDIRECT}, \text{HARD-PAUSE}\}. \tag{2}$$

The annotation assesses whether intervention has a concrete reason to help task completion, including whether the agent is already able to recover autonomously. Each logged decision point reveals only its realized continuation, so y_t is a hindsight-informed judgment rather than a measured difference between outcomes under alternative routes. HISENTINEL learns $p_\theta(y_t \mid x_t)$; at deployment, neither e_t nor any subsequent outcome is available.

4.2 HINDSIGHT-GUIDED INTERVENTION DISTILLATION

Our goal is to transfer the evidence contained in completed trajectories into a lightweight sentinel that must operate before action execution. We therefore introduce **H**indsight-**G**uided **I**ntervention **D**istillation (HGID). A privileged teacher observes the future and task outcome, while the causal student receives only the task, trajectory prefix, and proposed action. Distillation is applied only when the future improves the teacher's prediction over the corresponding outcome-masked view.

Because natural HARD-PAUSE events are scarce, the teacher predicts only ALLOW or INTERVENE. AUQ-derived HARD-PAUSE examples are excluded from teacher training because their suffixes occur after the missing information has already been supplied, but they remain available for three-class student supervision. To support subsequent feedback generation, we formulate routing as generative classification: the model predicts the route through its native language-modeling head rather than a separate classification head (Raffel et al., 2020; Zhang et al., 2025a; He et al., 2025). The generated route can consequently serve as the prefix for producing corrective guidance or a request for human assistance.

$$b_t = g(y_t) = \begin{cases} \textsc{Allow}, & y_t = \textsc{Allow}, \\ \textsc{Intervene}, & y_t \in \{\textsc{Redirect}, \textsc{Hard-Pause}\}. \end{cases} \tag{3}$$

Given the causal input x_t and its recorded future f_t, the teacher is optimized by

$$\mathcal{L}_{\text{teacher}} = -\mathbb{E}_{(x_t, f_t, b_t)\sim\mathcal{D}_F} \log p_\phi(b_t \mid x_t, f_t), \tag{4}$$

where $\mathcal{D}_F$ contains trajectories with valid future evidence. The student retains the full three-class output space and is trained on the complete dataset, including AUQ-derived examples:

$$\mathcal{L}_{\text{route}} = -\mathbb{E}_{(x_t, y_t)\sim\mathcal{D}} \log p_\theta(y_t \mid x_t). \tag{5}$$

Both models express their decisions through the native generative output space, without introducing a separate classification head. To align the binary teacher with the three-class student, we aggregate the student probabilities as

$$\begin{aligned} \bar{p}_\theta(\textsc{Allow} \mid x_t) &= p_\theta(\textsc{Allow} \mid x_t), \\ \bar{p}_\theta(\textsc{Intervene} \mid x_t) &= p_\theta(\textsc{Redirect} \mid x_t) + p_\theta(\textsc{Hard-Pause} \mid x_t). \end{aligned} \tag{6}$$

This decomposition allows the teacher to supervise whether an intervention is warranted, while the three-class labels determine whether that intervention should be autonomous redirection or human assistance.

We use helpfulness gate to further prevent uninformative teacher predictions from dominating student learning. For each valid trajectory, we compare the teacher's future-aware prediction with a control view in which the privileged outcome is removed. Distillation is enabled only when the future-aware teacher predicts the correct binary label and assigns it a higher probability than the control view:

$$m_t = \mathbb{I}\left[\arg\max_b p_\phi(b \mid x_t, f_t) = b_t \ \wedge \ p_\phi(b_t \mid x_t, f_t) > p_\phi(b_t \mid x_t, \varnothing)\right]. \tag{7}$$

The final student objective combines direct three-class supervision with selective future distillation:

$$\begin{aligned} \mathcal{L}_{\text{HGID}} = \mathcal{L}_{\text{route}} \\ + \lambda\, \mathbb{E}_{(x_t, f_t)\sim\mathcal{D}_F} \left[m_t \tau^2 D_{\text{KL}}\big(p_\phi^\tau(\cdot \mid x_t, f_t) \,\|\, \bar{p}_\theta^\tau(\cdot \mid x_t)\big)\right], \end{aligned} \tag{8}$$

where τ is the distillation temperature and λ controls the strength of privileged supervision. After training, the teacher and all future information are removed, leaving a lightweight three-class sentinel that operates solely on the causal pre-action context.

4.3 From Intervention Decisions to Actionable Feedback

A route label alone does not specify how the agent should recover. We therefore activate a feedback module after the routing decision. For REDIRECT, it generates a grounded problem description and a minimal actionable suggestion; for HARD-PAUSE, it asks for the missing information, authorization, or preference. ALLOW requires no feedback.

We first perform supervised fine-tuning (SFT) on the annotated feedback in SWE-INTERVENE. Given the causal context x_t, fixed route y_t, and annotated response r_t^+, the feedback module is optimized with teacher forcing:

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}\left[\frac{1}{|r_t^+|}\sum_{k=1}^{|r_t^+|}\log \pi_\phi\left(r_{t,k}^+ \mid x_t, y_t, r_{t,<k}^+\right)\right]. \tag{9}$$

This stage teaches the basic structure and content of intervention feedback.

Starting from the SFT policy, we further refine feedback quality using DPO (Rafailov et al., 2023) over preferred and rejected responses (r_t^+, r_t^-). Let

$$\Delta_\psi = \log\frac{\pi_\psi(r_t^+ \mid x_t, y_t)}{\pi_{\mathrm{ref}}(r_t^+ \mid x_t, y_t)} - \log\frac{\pi_\psi(r_t^- \mid x_t, y_t)}{\pi_{\mathrm{ref}}(r_t^- \mid x_t, y_t)}, \tag{10}$$

where π_{ref} is a frozen copy of the SFT policy. We retain an NLL term on the preferred response while optimizing the preference objective:

$$\mathcal{L}_{\mathrm{feedback}} = \mathcal{L}_{\mathrm{NLL}}(r_t^+) - \eta\,\mathbb{E}\left[\log\sigma(\beta\Delta_\psi)\right]. \tag{11}$$

Both SFT and DPO update only the feedback module. We freeze the routing parameters learned through HGID and condition feedback generation on the fixed predicted route. Consequently, feedback training cannot change whether the sentinel selects ALLOW, REDIRECT, or HARD-PAUSE; it only improves the response produced after an intervention is selected.

5 Experiments

5.1 Experimental Setup

Models. We use Qwen3-Coder-30B-A3B-Instruct as the privileged teacher and instantiate the lightweight sentinels with Qwen3-0.6B and Qwen3-1.7B (Team, 2025). At deployment, we use MINI-SWE-AGENT (Yang et al., 2024) as the coding-agent harness and evaluate the sentinels with two open-weight coding agents: Qwen3-Coder-30B-A3B-Instruct and Devstral-Small-2-24B-Instruct-2512 (Rastogi et al., 2025). The latter is a comparably sized model specialized for agentic software-engineering tasks, allowing us to evaluate whether the learned intervention policy transfers across coding-agent families. We additionally evaluate Claude-Sonnet-4.6 (Anthropic, 2026) as a closed-source reference to assess transfer to a frontier proprietary coding agent.

Benchmarks. We evaluate **HiSentinel** on static intervention recognition and end-to-end task completion. For static recognition, **SWE-Intervene** evaluates three-way action classification among ALLOW, REDIRECT, and HARD-PAUSE; **RootSE** (Wang et al., 2026) localizes root causes in failed coding trajectories; and the PROGRAM subset of **R-Judge** (Yuan et al., 2024) classifies program-related reasoning trajectories as safe or unsafe. For end-to-end evaluation, **SWE-bench Verified Mini** (Hobbhahn, 2025) is a 50-task subset of **SWE-bench Verified** (Jimenez et al., 2024) preserving its performance, test-pass-rate, and difficulty distributions, with results closely matching the full benchmark across 16 models. **Ask or Assume** (Edwards & Schuster, 2026) masks task-critical details to test clarification behavior; we stratify 100 of its 500 tasks by repository and difficulty to approximate the full-set distribution. We cap each task at 30 minutes and 100 agent steps.

Baselines. For static intervention recognition, **Step-by-Step** makes predictions from the causal prefix by examining actions sequentially. Post-hoc baselines observe the complete trajectory: **All-at-Once** diagnoses it in a single prompt; **AgentRx** identifies failures by synthesizing and checking constraints in multi-agent trajectories (Barke et al., 2026); **TrajAudit** uses an investigator agent with context reduction to locate decisive errors (Wang et al., 2026); and **RCTA** traces candidate errors through segment summaries, dependencies, and agent handoffs (Zhang et al., 2026c).

Table 3: **Results on static intervention-recognition benchmarks.** No data from RootSE or R-Judge are used for training. M-F1 and I-F1 denote Macro-F1 and Intervention F1. BAcc denote Balanced accuracy. For RootSE, T@k counts predictions within k steps of the annotated earliest decisive error. Cost denotes the average total number of input and output tokens per trajectory.

Backbone	Method	In-Domain			OOD					
		SWE-Intervene			RootSE			R-Judge		
		M-F1 ↑	I-F1 ↑	Cost ↓	T@0 ↑	T@1 ↑	Cost ↓	BAcc ↑	M-F1 ↑	Cost ↓
Qwen3-0.6B	Step-by-Step	27.23	21.21	8.08K	3.92	10.78	0.12M	60.68	60.61	0.58K
	All-at-Once	28.88	29.79	8.08K	3.92	6.86	0.07M	60.02	60.77	0.66K
	TrajAudit	30.23	19.13	10.16K	0.00	3.92	0.10M	50.66	34.92	1.80K
	RCTA	28.61	19.75	11.28K	2.94	10.78	0.03M	50.00	32.09	1.30K
	AgentRx	25.78	25.51	9.16K	0.98	7.84	2.28M	49.17	31.72	3.40K
	HiSentinel (Ours)	**79.99**	**75.22**	5.91K	**9.80**	**16.67**	0.07M	**65.09**	**61.68**	1.71K
Qwen3-1.7B	Step-by-Step	29.10	36.46	8.10K	3.92	12.75	0.06M	52.76	35.99	0.37K
	All-at-Once	27.31	31.38	8.10K	3.92	10.78	0.07M	57.48	56.04	0.66K
	TrajAudit	29.94	38.42	10.26K	2.94	8.82	0.10M	57.48	55.77	1.30K
	RCTA	27.94	35.89	11.42K	2.94	11.76	0.04M	49.61	36.89	1.29K
	AgentRx	27.94	33.92	9.23K	2.94	12.74	3.09M	**66.14**	**65.60**	3.51K
	HiSentinel (Ours)	**81.77**	**78.97**	5.91K	**9.80**	**15.69**	0.08M	**66.14**	65.09	1.24K

For end-to-end evaluation, **Step-by-Step** is a prompt-only baseline that inspects each proposed action before execution. **SWE-PRM** provides taxonomy-guided corrective feedback every five executed steps (Gandhi et al., 2025), while **Steer, Don't Solve** uses an SFT- and DPO-trained critic to provide high-level guidance at the same interval (Gandhi et al., 2026). Because its DPO training data are not publicly available, we directly use the strongest publicly released checkpoint.

Table 2: **End-to-end results.** SWE-V Mini denotes SWE-bench Verified Mini; SDS denotes STEER, DON'T SOLVE. Res.: resolved (%). Cost: average input and output tokens per trajectory (millions).

Agent	Backbone	Method	SWE-V Mini		Ask or Assume	
			Res. ↑	Cost ↓	Res. ↑	Cost ↓
Sonnet-4.6	–	Base	60.00	0.26	58.00	0.47
	Qwen3-1.7B	HISENTINEL	**66.00**	0.41	**61.00**	0.82
Qwen3-Coder	–	Base	30.00	0.29	23.00	0.26
		Reflexion	34.00	0.31	20.00	0.36
	SDS-4B	SDS	28.00	0.57	24.00	0.44
	Qwen3-0.6B	Step-by-Step	18.00	0.12	20.00	0.30
		SWE-PRM	32.00	0.38	22.00	0.32
		HISENTINEL	**36.00**	0.38	**28.00**	0.34
	Qwen3-1.7B	Step-by-Step	32.00	0.45	21.00	0.33
		SWE-PRM	20.00	0.35	25.00	0.36
		HISENTINEL	**44.00**	0.43	**33.00**	0.39
Devstral	–	Base	20.00	1.16	19.00	1.25
		Reflexion	24.00	1.31	21.00	1.40
	SDS-4B	SDS	24.00	1.46	20.00	1.53
	Qwen3-0.6B	Step-by-Step	18.00	1.21	18.00	1.31
		SWE-PRM	22.00	1.28	21.00	1.35
		HISENTINEL	**26.00**	1.30	**24.00**	1.39
	Qwen3-1.7B	Step-by-Step	22.00	1.25	20.00	1.34
		SWE-PRM	24.00	1.30	22.00	1.39
		HISENTINEL	**30.00**	1.34	**27.00**	1.44

5.2 EXPERIMENT RESULTS

Static intervention recognition. HISENTINEL substantially outperforms prompt-based and post-hoc baselines on the in-domain SWE-INTERVENE test set, while requiring fewer tokens per trajectory. The advantage also transfers to RootSE and R-Judge, neither of which is used for training: HISENTINEL provides the strongest failure-localization results on RootSE and achieves the best or competitive classification performance on R-Judge. These results show that the learned intervention policy generalizes beyond the annotation distribution used for training.

End-to-end agent performance. More importantly, the improved intervention decisions translate into consistently higher task completion rates. Across Qwen3-Coder, Devstral, and Sonnet-4.6, both HISENTINEL variants improve over the corresponding unassisted agents and competing intervention

methods on SWE-bench Verified Mini and Ask or Assume. The largest gains are obtained with the 1.7B model: for Qwen3-Coder, it improves resolution from 30% to 44% on SWE-bench Verified Mini and from 23% to 33% on Ask or Assume. Similar improvements with Devstral and the stronger Sonnet-4.6 agent indicate that HISENTINEL is not tied to a particular coding model. Although monitoring introduces additional token cost, the cost remains comparable to other feedback-based baselines while yielding substantially larger improvements in task completion.

5.3 ABLATION STUDY

Table 4: Ablations of HISENTINEL. (a) Routing on SWE-INTERVENE: M-F1/I-F1 denote three-class/Intervention F1; R-F1/HP-F1 denote class-wise F1. (b) Feedback groundedness (Ground.) and actionability (Action.) pass rates. (c) Resolved rate and cost on SWE-bench Verified Mini. F1, pass rates, and resolved rates are percentages.

(a) Intervention routing

Variant	M-F1	I-F1	R-F1	HP-F1
w/ 3-Class Teacher	75.53	74.42	62.87	80.00
w/ Classification Head	66.92	67.22	58.91	88.36
w/o Distillation	76.42	73.31	60.30	86.74
w/ Causal Teacher	77.16	75.08	**68.22**	84.63
w/o Route SFT	68.91	61.84	54.73	84.26
w/o Helpfulness Gate	78.72	77.41	61.06	91.18
w/ Self-Distillation	76.30	74.46	61.21	88.67
w/ Joint Feedback	78.67	77.96	57.30	92.49
HISENTINEL (Full)	**81.77**	**78.97**	66.30	**92.90**

(b) Feedback quality

Variant	Ground.	Action.
w/ Joint Feedback	66.9	63.2
w/o DPO	69.4	65.8
HISENTINEL (Full)	**76.8**	**73.5**

(c) SWE-bench Verified Mini

Variant	Resolved↑	Cost↓
w/o Distillation	34.00	0.43
w/ Causal Teacher	36.00	0.42
HISENTINEL (Full)	**44.00**	0.43

Ablation setup. We ablate HISENTINEL on teacher supervision, routing, and feedback learning. All variants use the 1.7B backbone with the same data and optimization settings. Due to the cost of interactive evaluation, we select two representative routing ablations for the end to end study.

Routing. On SWE-INTERVENE, the full model achieves 81.77 M-F1 and 78.97 I-F1. Ablating distillation, replacing the future-aware teacher with a causal or self-distilled alternative, or removing the helpfulness gate consistently degrades performance, supporting the contribution of each component in HGID. Alternative routing implementations, including a classification head and removing route SFT, lead to substantially larger drops.

Feedback learning. With routing held fixed, the full model reaches 76.8% groundedness and 73.5% actionability. Joint training lowers these rates to 66.9% and 63.2%, while removing DPO yields 69.4% and 65.8%, respectively.

End to end impact. On SWE-bench Verified Mini, removing distillation and using a causal teacher reduce resolution from 44% to 34% and 36%, despite their smaller gaps in static F1 and nearly identical token costs. Both variants make fewer false alarms but also miss more necessary interventions. A coding agent can sometimes recognize a low-risk false alarm and continue productively; when an intervention is missed, the flawed action proceeds. This asymmetry makes intervention recall especially consequential during execution.

5.4 ANALYSIS

5.4.1 DOES HISENTINEL BENEFIT FROM PRECISE OR FREQUENT INTERVENTION?

Effective oversight depends on intervening well, not often: mistimed feedback can derail an otherwise successful trajectory. We judge each intervention solely from the task, proposed action, and pre-intervention trajectory, since even rare stochastic continuations could make poor guidance appear successful; final rescues and regressions are reported separately. On SWE-BENCH VERIFIED MINI, 109 of HISENTINEL's 242 interventions (45.0%) are beneficial, 106 neutral, and 27 harmful; eight rescues and one regression raise the resolved count from 15/50 to 22/50. STEP-BY-STEP has one beneficial intervention among 24 (4.2%) and one rescue without regression; SWE-PRM

has three among 284 (1.1%), one rescue, and six regressions. Thus, HISENTINEL's gains reflect intervention quality rather than frequency.

5.4.2 WHY WE RECOMPUTE THE CAUSAL PREFIX

Since each monitoring input largely extends the preceding causal prefix by one step, cross-step KV reuse appears to be a natural optimization. We evaluate per-task BF16 exact-prefix caching for the 1.7B HISENTINEL, but do not adopt it in the final system. In the cached run, untruncated calls reuse 86.4% of their input tokens; however, 906/1,869 calls require left truncation, for which reuse falls to 3.2%, limiting the token-weighted overall reuse ratio to 29.8%. Compared with full-prefix recomputation, the observed mean number of processed Sentinel tokens decreases only from 382K to 299K per task (22%), while mean end-to-end time remains similar (757 s vs. 763 s). A matched audit of 896 token-identical routing calls further identifies 13 routing flips and three effective-intervention flips. On SWE-BENCH VERIFIED MINI, these differences reduce resolution from 22/50 with full recomputation to 18/50 with cross-step caching. We therefore recompute the complete causal prefix at every monitoring step, while retaining ordinary within-generation KV caching.

5.4.3 CASE STUDY: REDIRECTING AN INEFFECTIVE PATCH

Figure 3 shows a representative REDIRECT episode from the 1.7B HISENTINEL run on `sphinx-doc__sphinx-10673`. The task requires Sphinx to suppress missing-document warnings for documents generated by extensions while preserving the warnings for genuinely missing documents. The coding agent attempted to implement this behavior with a nested branch containing an empty `pass`, which still allowed execution to reach the existing warning logic. HISENTINEL blocked the edit and requested a focused, evidence-grounded correction. The agent then inspected the relevant block and moved the generated-document check before the missing-document branch. The resulting run passed `test_toctree_index` and all nine regression tests, whereas the paired no-intervention run remained unresolved. This episode illustrates how a REDIRECT steers the agent toward a concrete corrective action rather than merely providing a post-hoc warning.

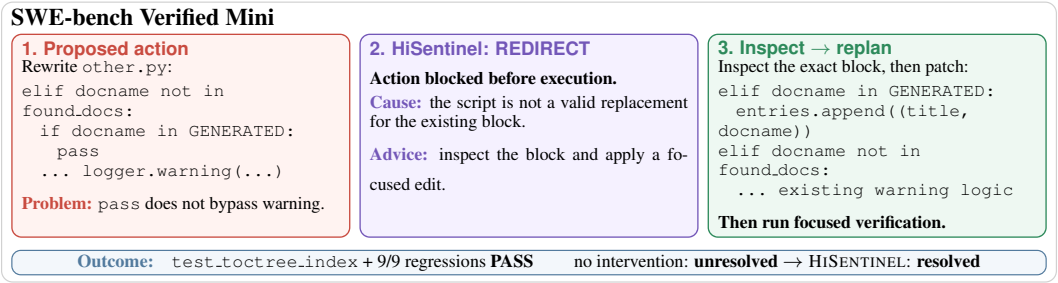


Figure 3: A representative causal intervention. The displayed REDIRECT is one local episode in the trajectory; the final resolved/unresolved labels are produced by the official SWE-bench evaluator.

6 CONCLUSION

We introduced HISENTINEL, a pre-execution intervention framework that distills outcome-aware judgments from a privileged teacher into a lightweight causal Sentinel. Seeing only the observed trajectory and proposed action, it chooses among ALLOW, REDIRECT, and HARD-PAUSE, and provides actionable feedback when intervention is needed. We also introduced SWE-INTERVENE to study these decisions on natural coding trajectories. The 1.7B model achieves 81.77 macro-F1 and 78.97 intervention F1 on this benchmark and transfers to RootSE and R-Judge. In end-to-end evaluation with Qwen3-Coder, it improves resolution from 30% to 44% on SWE-bench Verified Mini and from 23% to 33% on Ask or Assume, with gains also observed for Devstral and Sonnet-4.6. These results show that hindsight supervision can improve coding agents through timely, useful intervention.

AI USE STATEMENT

Generative AI tools were used in this work to assist with language editing, the preparation and refinement of figures and tables, data annotation, and evaluation as LLM judges. All AI-assisted materials and annotations were reviewed and verified by the authors to ensure accuracy and faithful representation of the research. These tools were not used to independently produce experimental results, conduct scientific analyses, or formulate research conclusions. The authors retain full responsibility for all content presented in this paper.

REFERENCES

Wasi Uddin Ahmad, Nikolai Ludwig, Somshubra Majumdar, and Boris Ginsburg. Open-swe-traces: Advancing dual-mode multilingual distillation for software engineering agents. *arXiv preprint arXiv:2606.16038*, 2026.

Anthropic. Claude sonnet 4.6. Anthropic, 2026. URL `https://www.anthropic.com/claude/sonnet`. Accessed: 2026-09-20.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. Agentrx: Diagnosing ai agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

Joachim Baumann, Vishakh Padmakumar, Xiang Li, John Yang, Diyi Yang, and Sanmi Koyejo. Swe-chat: Coding agent interactions from real users in the wild. *arXiv preprint arXiv:2604.20779*, 2026.

Yang Cai, Xiangyu Liu, Argyris Oikonomou, and Kaiqing Zhang. Provable partially observable reinforcement learning with privileged information. *Advances in Neural Information Processing Systems*, 37:63790–63857, 2024.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on robot learning*, pp. 66–75. PMLR, 2020.

Yutong Chen, Guangfu Guo, Zhichao Xu, and Kunpeng Liu. Dualopsd: Adaptive privileged teachers for on-policy self-distillation. *arXiv preprint arXiv:2608.26019*, 2026.

Zhi Chen, Wei Ma, and Lingxiao Jiang. Beyond final code: A process-oriented error analysis of software development agents in real-world github scenarios. *arXiv preprint arXiv:2503.12374*, 2025.

Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: Llm agents that learn from privileged ai feedback. In *International Conference on Learning Representations*, volume 2025, pp. 10457–10493, 2025.

Phuong Tuan Dat, Qi Li, and Xinchao Wang. dopsd: On-policy self-distillation for diffusion language models. *arXiv preprint arXiv:2607.04428*, 2026.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization. *arXiv preprint arXiv:2505.08638*, 2025.

Nicholas Edwards and Sebastian Schuster. Ask or assume? uncertainty-aware clarification-seeking in coding agents. *arXiv preprint arXiv:2603.26233*, 2026.

Pengcheng Fang, Hongli Chen, and Xiaohao Cai. Privileged foresight distillation: Zero-cost future correction for world action models. *arXiv preprint arXiv:2604.25859*, 2026.

Yujian Gan, Changling Li, Jinxia Xie, Luou Wen, Matthew Purver, and Massimo Poesio. Clarq4llm: A benchmark for models clarifying and requesting information in task-oriented dialog. *IEEE Transactions on Audio, Speech and Language Processing*, 2026.

Shubham Gandhi, Jason Tsay, Jatin Ganhotra, Kiran Kate, and Yara Rizk. When agents go astray: Course-correcting swe agents with prms. *arXiv preprint arXiv:2509.02360*, 2025.

Shubham Gandhi, Yiqing Xie, Atharva Naik, Ruichen Zhu, and Carolyn Rose. Steer, don't solve: Training small critic models for large code agents. *arXiv preprint arXiv:2606.21811*, 2026.

Mingqian He, Fei Zhao, Chonggang Lu, Ziyan Liu, Yue Wang, and Haofu Qian. Gencls++: Pushing the boundaries of generative classification in llms through comprehensive sft and rl studies across diverse datasets. *arXiv preprint arXiv:2504.19898*, 2025.

Marius Hobbhahn. SWE-bench Verified Mini. `https://github.com/mariushobbhahn/SWEBench-verified-mini`, 2025. GitHub repository, accessed September 18, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *arXiv preprint arXiv:2607.05184*, 2026.

Belinda Li, Been Kim, and Zi Wang. Questbench: Can llms ask the right question to acquire information in reasoning tasks? *Advances in Neural Information Processing Systems*, 38, 2026.

Shuyang Liu, Yang Chen, Rahul Krishna, Saurabh Sinha, Jatin Ganhotra, and Reyhaneh Jabbarvand. Process-centric analysis of agentic software systems. *Proceedings of the ACM on Programming Languages*, 10(OOPSLA1):1961–1988, 2026.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. 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.

Nikolai Ludwig, Wasi Uddin Ahmad, Somshubra Majumdar, and Boris Ginsburg. From swe-zero to swe-hero: Execution-free to execution-based fine-tuning for software engineering agents. *arXiv preprint arXiv:2604.01496*, 2026.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

Oorja Majgaonkar, Zhiwei Fei, Xiang Li, Federica Sarro, and He Ye. Understanding code agent behaviour: An empirical study of success and failure trajectories. *arXiv preprint arXiv:2511.00197*, 2025.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

OpenAI. GPT-5.5 System Card. `https://openai.com/index/gpt-5-5-system-card/`, April 2026.

Tianyue Ou, Wanyao Guo, Apurva Gandhi, Graham Neubig, and Xiang Yue. Agentdiagnose: An open toolkit for diagnosing llm agent trajectories. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 207–215, 2025.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Yunjia Qi, Zehua Yin, Xintong Shi, Hao Peng, Songyuanyi Lu, Yixian Liu, Richeng Xuan, Yuhong Liu, Zhichao Hu, Xiaozhi Wang, et al. Trajdebug: Tracing error lifecycle to identify critical failures in long-horizon agent trajectories. *arXiv preprint arXiv:2608.06346*, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Md Nakhla Rafi, Md Ahasanuzzaman, Dong Jae Kim, Zhijie Wang, and Tse-Hsun Chen. Falat: Tracing failures in llm agent trajectories via dependency-guided search. *arXiv preprint arXiv:2606.00765*, 2026.

Abhinav Rastogi, Adam Yang, Albert Q Jiang, Alexander H Liu, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Anmol Agarwal, Andy Ehrenberg, Andy Lo, et al. Devstral: Fine-tuning language models for coding agent applications. *arXiv preprint arXiv:2509.25193*, 2025.

Samyak Shrestha and Alexander Tessier. Rethinking privileged information in on-policy self-distillation. *arXiv preprint arXiv:2608.18271*, 2026.

Rui Shu, Chun Yong Chong, Xin Zhou, Yun Peng, Zihan Wu, Xu Han, Zeyang Zhuang, Guowen Yuan, and Yuan Wang. What resolve rate hides: Trajectory structure diagnostics for coding agents. *arXiv preprint arXiv:2607.06184*, 2026.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *The Journal of Machine Learning Research*, 16(1):2023–2049, 2015.

Minxing Wang, Xiaofei Xie, and Yintong Huo. Trajaudit: Automated failure diagnosis for agentic coding systems. *arXiv preprint arXiv:2605.26563*, 2026.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. R-judge: Benchmarking safety risk awareness for llm agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1467–1490, 2024.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems? In *International Conference on Learning Representations*, volume 2026, pp. 11377–11399, 2026a.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Seyed Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations*, volume 2025, pp. 12476–12505, 2025a.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025b.

Xinsen Zhang, Zhenkai Ding, Tianjun Pan, Run Yang, Chun Kang, Xue Xiong, and Jingnan Gu. Opsdl: On-policy self-distillation for long-context language models. *arXiv preprint arXiv:2604.17535*, 2026b.

Yunfei Zhang, Boyu Feng, Changhua Pei, Zexin Wang, Zhihuang Peng, Xinlong Liu, Hengyue Jiang, Difeng Ma, Jiayi Zhang, Yongzhou Yao, et al. Longrca bench: Diagnosing responsible roles and root causes in long-horizon agent failures. *arXiv preprint arXiv:2608.15242*, 2026c.

Jiale Zhao, Ke Fang, and Lu Cheng. When and what to ask: Askbench and rubric-guided rlvr for llm clarification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17120–17140, 2026a.

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. *arXiv preprint arXiv:2601.18734*, 2026b.

Xiangxin Zhao, Han Li, Shuaiting Li, Tianyi Zhao, Earl T Barr, Federica Sarro, and He Ye. Failure as a process: An anatomy of cli coding agent trajectories. *arXiv preprint arXiv:2607.09510*, 2026c.

A Appendix

A.1 SWE-Intervene: Construction and Quality Audit

A.1.1 Data Sources and Instance Construction

SWE-Intervene is constructed from three complementary sources of software-engineering agent trajectories. Open-SWE-Traces (Ahmad et al., 2026) provides repository-level trajectories generated by coding agents, while SWE-Hero (Ludwig et al., 2026) contributes trajectories containing diverse development and debugging behaviors. We further use SWE-chat (Baumann et al., 2026), which contains naturally occurring interactions between coding agents and developers. The resulting dataset contains 6,923 action-level instances, as summarized in Table 5.

Table 5: Composition of SWE-Intervene.

Source	Instances	Proportion
Open-SWE-Traces	3,451	49.85%
SWE-Hero	1,089	15.73%
SWE-chat	2,383	34.42%
Total	**6,923**	**100%**

The unit of annotation is a decision point immediately before the execution of a proposed action. Each instance contains the task description u, the relevant trajectory prefix h_t, the complete proposed action a_t, and an intervention label y_t. The recorded continuation and task outcome are retained as privileged evidence for annotation and teacher training, but are excluded from the causal input available to the deployed sentinel.

Rather than labeling every step uniformly, we prioritize decision-critical nodes. In failed trajectories, these include points at which an error begins to propagate, the agent overlooks decisive evidence, or intervention could prevent an incorrect completion claim. We also retain Allow instances in which the agent is making reasonable progress, including imperfect actions from which it is already recovering autonomously. This distinction prevents the dataset from reducing intervention detection to local error detection.

A.1.2 Annotation Protocol

GPT-5.5 (OpenAI, 2026) assigns one of the three labels defined in Section 4.1, following criteria and annotation guidelines written by human researchers. The annotator is shown the task, relevant history, complete proposed action, and recorded continuation. It is explicitly instructed to judge whether intervention at the current node would improve the probability of eventual task completion relative to allowing the agent to continue, rather than whether the action could merely be made more standardized or efficient.

For every candidate intervention, the annotation process considers three questions:

1. Does the identified problem materially threaten task completion, rather than merely reflect an imperfect or nonstandard operation?

2. Has the agent already recognized the problem and begun a reasonable recovery process?

3. What concrete loss would an intervention prevent, or what current obstacle would it remove, compared with autonomous continuation?

A node is labeled REDIRECT only when there is concrete evidence that the proposed action may impede task completion and corrective guidance can enable autonomous recovery. A node is labeled HARD-PAUSE when progress depends on task-specific information, authorization, or preferences unavailable to the agent. Otherwise, including cases in which the agent is already correcting its own mistake, the node is labeled ALLOW.

A.1.3 RECONSTRUCTING NATURAL HARD-PAUSE INSTANCES

Naturally occurring HARD-PAUSE events are rare in standard coding-agent trajectories. We therefore extract additional examples from SWE-CHAT interactions containing `AskUserQuestion` (AUQ) calls. These interactions provide direct evidence that the agent encountered information, authorization, or a user preference that it could not infer autonomously.

For each retained AUQ interaction, we reconstruct the decision point immediately before the question is issued. The current AUQ call and its corresponding human response are removed from the causal input, so the sentinel must infer the need for assistance solely from information available at that decision point. Earlier AUQ exchanges may remain in the history only when they occurred strictly before the reconstructed decision point; their responses are therefore part of the legitimately observed state rather than privileged future information. Multiple valid HARD-PAUSE nodes may be extracted from one trajectory, but all nodes from the same trajectory are assigned to the same data split. We perform splitting and de-duplication at the trajectory level, yielding zero trajectory overlap between training and held-out evaluation data.

We retain only AUQ cases in which the requested information is necessary for current progress. Questions that are optional, premature, already answered by the task context, or unrelated to a present obstacle are excluded. Because the suffix following an AUQ call is conditioned on the human response, reconstructed AUQ instances provide only causal three-class supervision for the student and are never used to construct privileged-future targets for the teacher.

A.1.4 TWO-STAGE QUALITY AUDIT

We conduct a two-stage human audit focused on provisional REDIRECT and HARD-PAUSE instances, since unnecessary intervention can disrupt an otherwise successful trajectory. In the first stage, reviewers inspect the complete task context, proposed action, existing label, and recorded continuation using the three criteria above. The original annotation is retained when it is well supported; revision is required only when the trajectory provides clear evidence that the intervention is unnecessary or temporally misplaced.

The second stage independently cross-checks proposed removals and ambiguous cases. Particular attention is paid to nodes involving superficially suspicious operations, such as commands that obscure exit codes, repeated searches, or repeated builds. Such patterns are not treated as intervention-worthy by themselves: they are retained only when the surrounding task state establishes a concrete threat to completion. Semantic judgments are made from the trajectory content; automatic scripts are used only for deterministic extraction and schema validation, not for assigning or revising labels.

Among 2,252 provisional intervention candidates, the audit removes 570 instances that lack sufficient evidence for intervention, corresponding to a rejection rate of 25.3%. The remaining 1,682 candidates exhibit a concrete completion-relevant failure risk or an obstacle that cannot be resolved without external assistance.

The most common rejection cases involve locally imperfect but recoverable actions, problems that the agent has already recognized and is actively addressing, and HARD-PAUSE labels placed before external information is actually required. This audit aligns the dataset with the central objective of HISENTINEL: intervening when doing so is likely to improve final task completion, rather than correcting every detectable imperfection.

Table 6: Results of the intervention-candidate audit.

Audit outcome	**Instances**
Provisional intervention candidates	2,252
Retained after audit	1,682
Rejected after audit	570
Rejection rate	25.3%

A.2 TRAINING DETAILS.

We use the same training hyperparameters for the 0.6B and 1.7B HISENTINEL models unless otherwise specified. Training consists of causal routing supervision with future-view distillation, feedback SFT, and feedback preference optimization. The routing model is kept fixed during feedback optimization. Table 7 summarizes the complete configuration.

A.3 EVALUATION-TIME HARD-PAUSE PROTOCOL

Following the evaluation design of ASK-OR-ASSUME and established protocols for agents operating under missing or underspecified information, including ASKBENCH, CLARQ-LLM, TOOLSANDBOX, and QUESTBENCH (Edwards & Schuster, 2026; Zhao et al., 2026a; Gan et al., 2026; Lu et al., 2025; Li et al., 2026), we implement HARD-PAUSE as a controlled interaction with a proxy user rather than granting the evaluated agent direct access to the complete task specification. The evaluated agent initially observes only the underspecified task input. When the sentinel emits HARD-PAUSE, execution is suspended and the agent's clarification question is forwarded to a proxy-user API.

Each proxy-user request contains only three components: a fixed role instruction defining the information boundary, the agent's exact clarification question, and the withheld complete task specification as private evidence available exclusively to the proxy user. The proxy API is not given the reference solution or patch, hidden tests, verifier outputs, evaluation labels, sentinel route or risk score, future trajectory, or any other indication of the action expected from the evaluated agent. Consequently, the complete specification cannot enter the agent context except through an admissible answer to a clarification question.

The proxy user is instructed to return only the minimum information that directly answers the submitted question and is explicitly supported by the withheld specification. It must not provide chain-of-thought reasoning, implementation advice, code, patches, test outcomes, unsolicited requirements, additional hints, or information merely inferred from the intended solution. If the requested information is not present in the withheld specification, the proxy must abstain rather than speculate. This knowledge-boundary restriction follows the general provider-agent practice used in missing-information benchmarks, in which the simulated user possesses private task facts but is prevented from revealing a complete solution.

The resulting clarification is inserted verbatim into the interaction as the user's response, after which execution resumes from the paused state. Each admitted HARD-PAUSE produces at most one proxy-user response and remains subject to the same fixed intervention budget and cooldown used throughout evaluation. We log every query and response for auditing. Proxy-user latency and token usage are reported separately and are excluded from the evaluated model's inference cost, while the wall-clock task time includes the complete interaction.

The same proxy-user API is available to every baseline in all end-to-end benchmarks, under the information boundary and response restrictions described above. Baselines may request clarification when they determine that human input is needed; access to the API is not exclusive to HISENTINEL. In our runs, the other baselines rarely invoked it, consistent with their lack of training to recognize missing information and initiate clarification.

Table 7: **Training hyperparameters of HiSentinel.** The settings are shared by the Qwen3-0.6B and Qwen3-1.7B variants. The feedback learning rate decays to its floor after the first epoch.

Hyperparameter	Value
General configuration	
Backbones	Qwen3-0.6B and Qwen3-1.7B
Future-view teacher	Frozen Qwen3-Coder-30B-A3B-Instruct
Numerical precision	BF16
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$
Weight decay	0.01
Gradient clipping	1.0
LoRA configuration	Rank 32, $\alpha = 64$, dropout $= 0.05$
LoRA target modules	`q_proj`, `k_proj`, `v_proj`, and `o_proj`
Maximum causal input length	16,384 tokens
Causal routing and future-view distillation	
Training records	5,680
Epochs / optimizer steps	2 / 710
Effective batch size	16
Epoch seeds	42 and 43
Learning rate	5×10^{-5}
LR schedule	12-step linear warmup followed by cosine decay
Routing objective	Three-class route-token CE over ALLOW, REDIRECT, and HARD-PAUSE
Class-prior adjustment	$\tau = 0.85$
Distillation objective	Two-view future-to-causal binary-logit distillation
KD temperature / maximum weight	$T = 2.0$ / $\lambda_{\max} = 0.25$
KD-weight warmup	20 optimizer steps
Feedback supervised fine-tuning	
Training records	1,345
Epochs / optimizer steps	5 / 425
Batch size	16
Epoch seeds	42, 43, 44, 45, and 46
Learning rate	1×10^{-4}, decayed to a floor of 1×10^{-5}
LR schedule	5-step linear warmup followed by cosine decay
Objective	Mean next-token CE over feedback tokens and EOS
Route-token loss	None; routing parameters remain fixed
Feedback preference optimization	
Training pairs	800
Optimizer steps / maximum batch size	50 / 16 pairs
Random seed	42
Learning rate	5×10^{-6}
LR schedule	5-step linear warmup followed by cosine decay
DPO coefficient	$\beta = 0.1$
Loss weights	Preferred-response NLL $= 1.0$; DPO loss $= 1.0$
Reference policy	Frozen feedback-SFT checkpoint
Dropout	Disabled during preference optimization