COUNTERFACTUAL SUPERVISION FOR CREDIT ASSIGNMENT IN LANGUAGE MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We develop a shared turn-credit selector using counterfactual supervision from 4B and 9B policies before and after reinforcement learning. Successful coding trajectories can contain harmful actions, while failed ones can contain useful decisions. To distinguish these local roles, we restore the state before each action and repeatedly execute both the observed action and same-policy alternatives in the environment. Their difference in average correctness–efficiency utility defines a gold credit label. These labels guide development of a parameter-frozen retrospective selector, which supplies the actor's complete turn-level advantage without online counterfactual branching. Each score is shared by the turn's action tokens in a clipped policy objective; training uses no learned value critic, GAE, or additional outcome advantage. Across four evaluation sets and both actor scales, our method exceeds all four trained baselines. SWE-bench Pro gains over PPO are 4.52 and 4.79 percentage points at 4B and 9B, respectively. At 9B, it matches the Base model on LiveCodeBench while exceeding the other trained methods. Credit-removal experiments distinguish effective learning from simple trajectory compression. Matched-policy probes further support improved local action selection under a shared continuation policy.

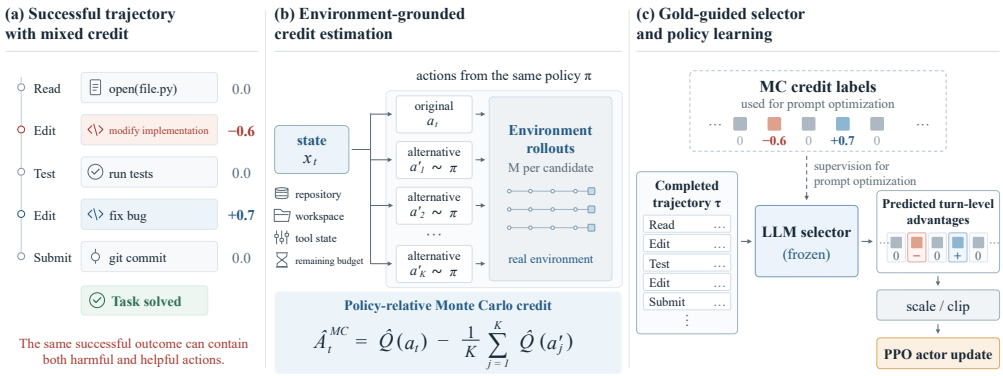


Figure 1: **Measure within a policy, learn across policies.** Each counterfactual experiment fixes its source policy; mixed-policy gold develops a shared selector. Its turn scores drive critic-free actor learning using PPO-style clipping, without a learned value critic. The motivating scores are illustrative.

1 INTRODUCTION

A coding agent can damage a file, repair it, and still pass the checker. Learning should preserve the recovery without reinforcing the mistake. Conversely, a failed trajectory can contain a useful diagnosis. The challenge is to assign credit to local decisions rather than broadcast the final outcome.

Dense feedback can be obtained from rollouts, learned process models, or generative critiques (Kazemnejad et al., 2025; Choudhury, 2025; Zhou et al., 2025; Xie et al., 2025; Zhang et al., 2026;

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

Tan et al., 2026). Yet a plausible critique need not estimate an action's value for the agent being trained. An apparently useful step may be no better than the agent's usual choices, while a failed test may reveal information that improves subsequent execution. We ask whether credit measured through counterfactual execution can become a reusable learning signal for coding agents.

We define turn-level gold credit through real counterfactual execution. Before an observed action, we restore the environment state and sample alternatives from the same source policy. Repeatedly continuing both the observed action and its alternatives estimates whether the action improves on that policy's usual choices. Our utility combines correctness with efficient successful completion, distinguishing direct progress from detours that are eventually repaired. These measurements provide a testable supervision reference for the source policy and utility, rather than an evaluator's verbal assessment alone.

The corpus combines 4B and 9B checkpoints before and after RL. The source policy is fixed *within* a comparison but varies *across* labels. These labels develop one shared selector's instructions and aggregation, with separate development and evaluation partitions. The resulting configuration is reused without online counterfactual branching.

The fixed selector supplies the actor's complete turn-level advantage. Each turn's credit is shared by its generated action tokens in a PPO-style clipped objective. Actor training uses no learned value critic, GAE, or additional outcome-advantage term. This retains clipping as an optimization safeguard while replacing learned value estimation with gold-guided local credit.

We evaluate the complete method through held-out credit prediction, downstream performance, and controlled interventions on its outputs. Across both actor scales and four evaluation sets, our method exceeds every trained baseline. Gains over PPO on SWE-bench Pro are 4.52 points at 4B and 4.79 points at 9B; on Terminal-Bench they are 2.25 and 10.11 points. Transfer is not uniformly large: the 9B model matches Base on LiveCodeBench and improves only slightly on BFCL. Interventions on the developed credit signal examine its use in learning, separately from the full configuration's gains.

Our contributions are:

- **Mixed-policy gold supervision.** Same-state, same-policy counterfactual experiments measure local credit across model scales and training stages.
- **A shared retrospective selector.** Development against these labels turns expensive environment measurements into a reusable scoring rule, evaluated against execution-derived credit.
- **Critic-free turn-credit learning.** Gold-guided predictions provide the actor's complete advantage signal, replacing learned value estimation while retaining a clipped policy update.

2 Related Work

Execution-based credit and advantage learning. VinePPO replaces learned value estimates with Monte Carlo estimates (Kazemnejad et al., 2025); AgentPRM trains process reward models from environment rollouts (Choudhury, 2025). Rewarding Progress formulates process feedback through advantage under a prover policy (Setlur et al., 2024), while SWEET-RL learns turn-wise advantages from trajectory preferences (Zhou et al., 2025). We build on the connection between relative action value and policy learning, using same-source-policy alternatives to define labels across model scales and training stages for one shared retrospective evaluator.

Generative and hindsight critics. CAPO shapes outcome-based rewards using generated step critiques (Xie et al., 2025). CriticSearch combines retrospective turn judgments with trajectory-level advantages (Zhang et al., 2026); HCAPO combines hindsight credit with group-relative outcome advantages (Tan et al., 2026). Our method instead links execution-derived gold credit to development of the evaluator and critic-free actor learning. Its predictions supply the full turn-level advantage rather than supplementing a separate outcome term. The contribution is this supervision-to-learning pipeline, not the use of a clipped objective alone.

Feedback-driven prompt development. OPRO optimizes prompts using evaluated candidates (Yang et al., 2024), and GEPA uses reflection and feedback (Agrawal et al., 2026). Our feedback is

agreement with environment-derived local credit. With fixed evaluator weights, prompt development seeks a reusable policy-learning signal rather than only a fluent critique.

3 METHOD

Our pipeline measures gold credit across source policies, develops one shared selector, and uses its predictions for actor learning (Figure 1). Within each annotation experiment, π_0 is the frozen source policy, x_t the restored state, h_t the visible history, and a_t the observed assistant turn. The source policy varies across the corpus; π_θ denotes the optimized actor. We distinguish gold $\widehat{A}_t^{\mathrm{MC}}$, prediction $\widehat{A}_t^{\mathrm{sel}}$, and training advantage A_t^{train}.

3.1 ENVIRONMENT-GROUNDED GOLD ANNOTATION

Controlled comparisons across a policy mixture. Sources include 4B and 9B checkpoints before and after RL. For each turn, we restore the messages, workspace, tool state, and remaining budget; retain the observed action; and sample K alternatives $a'_{t,j} \sim \pi_0(\cdot \mid h_t)$. The original and every alternative are executed and repeatedly continued in the real environment under that same π_0. Thus, policies are mixed across labels, not within a label's branches. The reference is the source policy's usual choices, not an expert replacement or the best sampled action.

Alternative sampling estimates what the agent would choose; repeated continuation estimates what happens after each choice. Repeating the original action's continuations also separates its value from the single observed outcome. Actual checker results, rather than imagined consequences, supply the supervision.

Correctness and efficient success. When many continuations eventually succeed, correctness alone cannot distinguish direct progress from avoidable detours. Let $Y(\tau)$ denote checker success, $C(\tau)$ voluntary completion, and $T(\tau)$ total assistant turns, including the prefix. With a reference length $\bar{T}$ estimated for the task and source policy, we define

$$U(\tau) = \begin{cases} -1, & Y(\tau) = 0, \\ +1, & Y(\tau) = 1,\ C(\tau) = 1,\ T(\tau) \le \bar{T}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Zero marks correct completion without the efficient-success bonus. The objective jointly rewards correctness and efficiency: $\mathbb{E}[U] = P(\text{success}) + P(\text{efficient success}) - 1$. It does not constrain success probability to remain fixed, and a zero-utility continuation need not yield zero advantage.

(a) A fixed observed edit

`showyourwork-456`, turn 4. State restored before editing `clean.py`.

```
for dir_path in
  paths.user().figures
  .rglob("*")[::-1]:
  if dir_path.is_dir():
    dir_path.rmdir()
```

(b) Repeated same-policy execution

The observed action and every alternative are followed by repeated continuations under the same frozen policy.

Fixed action(s)	$U=-1$	$U=0$	$U=+1$	Mean U
Observed action	11	2	3	-0.5000
12 alternatives	6	6	180	$+0.90625$

The baseline is the *average over sampled alternatives*, not the best branch.

$$\widehat{A}_t^{\mathrm{MC}} = \widehat{Q}_t(a_t) - \frac{1}{K}\sum_{j=1}^{K} \widehat{Q}_t(a'_{t,j}) = -0.5000 - 0.90625 = \mathbf{-1.40625}$$

Figure 2: **A gold label computed from real execution.** Line-wrapped excerpt and recorded outcome counts. The original action has 16 continuations; the 12 alternatives have 192 in total. Equal per-action counts make the pooled mean equal to the action-average baseline. This selected example illustrates label construction, not its frequency across the dataset.

Monte Carlo gold. Let $\widehat{Q}_t(a)$ be mean utility over valid continuations following fixed action a. We compute

$$\widehat{A}_t^{\mathrm{MC}} = \widehat{Q}_t(a_t) - \frac{1}{K}\sum_{j=1}^{K} \widehat{Q}_t(a'_{t,j}). \tag{2}$$

Positive credit means better expected utility than the source policy's alternatives; negative credit favors resampling. The gold label provides an execution-based, testable supervision reference for the specified source policy and utility. Averaging sampled returns follows Monte Carlo value estimation (Sutton & Barto, 2018); repetition stabilizes the finite-sample estimate of that target. Figure 2 gives a worked calculation. Sampling and invalid-execution conventions appear in Appendix A.

3.2 Developing a Shared Retrospective Selector

To avoid online counterfactual branching, we develop a reusable scoring rule with DeepSeek v4.1 Flash. It receives the task/sample identifier, terminal state, turn IDs, trajectory, and tool observations, and scores every assistant turn. It receives neither counterfactual branches nor gold labels, and no source-model name, scale, checkpoint, or training step. The recorded future supplies evidence for predicting the MC expectation, not a replacement for it. Each target is source-policy relative, while one selector is developed across these sources and held fixed during actor training. Its predictions approximate the target; exact current-policy advantages are not assumed.

Scoring-configuration development compares predictions with gold direction, magnitude, and within-trajectory ranking, subject to complete turn coverage. It optimizes both the scoring instructions and the use of multiple prompts. Behavioral descriptions can guide reasoning but do not define fixed category scores. Model weights remain fixed during development; the selected scoring configuration is then fixed during actor training. Full uses three evolved prompts. For each turn, we retain the signed member score with maximum absolute value and apply a fixed positive scale. This is neither majority voting nor turn filtering. Prompt texts, selection details, and scaling appear in Appendices B and D; the alternative 35B distillation study is kept separate.

3.3 Critic-Free Policy Learning with Turn-Level Credit

The actor learns directly from selector predictions rather than a learned value function. For each trajectory, we set

$$A_t^{\mathrm{train}} = \mathrm{clip}(g\widehat{A}_t^{\mathrm{sel}}, -A_{\max}, A_{\max}), \tag{3}$$

with fixed $g > 0$. Every generated action token in turn t shares this coefficient; tool observations and padding receive no direct actor loss. For trajectory i, let $\mathcal{I}_i$ contain action-token positions, $t(i,k)$ identify their turns, and $A_{ik}^{\mathrm{train}} = A_{i,t(i,k)}^{\mathrm{train}}$. Using recorded generating probabilities p_{ik}^{roll} and $\rho_{ik} = \pi_\theta(y_{ik} \mid h_{ik})/p_{ik}^{\mathrm{roll}}$, the PPO-style clipped surrogate (Schulman et al., 2017) is

$$\ell_\epsilon(\rho, A) = -\min\{\rho A, \mathrm{clip}(\rho, 1-\epsilon, 1+\epsilon)A\},$$
$$L_{\mathrm{clip}} = \frac{1}{B}\sum_i \frac{1}{|\mathcal{I}_i|}\sum_{k\in\mathcal{I}_i} \ell_\epsilon(\rho_{ik}, A_{ik}^{\mathrm{train}}). \tag{4}$$

Following turn-level action modeling (Zhou et al., 2025), all tokens within a turn share credit; probability ratios are clipped per token, not at the level of the joint turn probability. We retain negative-advantage dual clipping (Ye et al., 2020); exact implementation details are in the appendix. There is no learned value critic, generalized advantage estimation (GAE; Schulman et al., 2016), separate outcome advantage, or batch whitening. The terminal outcome informs scoring without subsequently overwriting local signs. These coefficients are a practical training surrogate, not guaranteed unbiased current-policy advantages.

4 Experimental Setup

Policies and supervision. The actors are Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026). The mixed-policy gold corpus contains 300 trajectories and 5,192 turns from checkpoints before and after RL. We randomly allocate 250 trajectories to development and 50 to evaluation; the latter contain 1,179 turns and are shared by both selector configurations. Behavioral and locality analyses use a separate 185-trajectory collection.

Training and comparisons. We train on SWE-rebench (Badertdinov et al., 2025) with the `mini_swe` scaffold. All generative credit evaluators use DeepSeek v4.1 Flash (DeepSeek, 2026). The initial selector makes one call; Full combines three evolved prompts by signed maximum-absolute selection. This evaluates scoring-configuration optimization, not prompt wording alone. CriticSearch (Zhang et al., 2026) and CAPO (Xie et al., 2025) are task-adapted process-feedback baselines (Appendix D.1). Negative-credit interventions share Full's settings and run for 64 actor updates. The 4B Base entry is a proxy reference.

Budget and checkpoint selection. Candidates are saved every 16 actor updates through update 128. For each method and scale, we select the highest unsmoothed batch-mean training environment score at those candidates: update 64 for 9B Ours and 128 for the other trained entries. One checkpoint serves all four benchmarks; benchmark results are not used for selection. PPO adds 128 critic-only warm-up updates, so matched actor updates do not imply equal total compute.

Evaluation sets and metrics. We use SWE-bench Pro (Deng et al., 2025), Terminal-Bench 2.1 (The Terminal-Bench Team, 2026), BFCL (Patil et al., 2025), and LiveCodeBench (Jain et al., 2025), with fixed evaluation sets of 730, 89, 800, and 131 tasks. The latter three use the `mini_swe_v2`, BFCL v4 Agent v3, and code-generation subset identifiers, respectively. All methods use the full set within each suite as the denominator. Subsets and execution budgets are specified in Appendix D.2. Selector evaluation reports negative precision/recall, high-impact weighted sign agreement, and NDCG@3 (Järvelin & Kekäläinen, 2002). Training statistics use updates 57–64: eight equally weighted batches and 512 accepted trajectories per run.

5 Results

5.1 Downstream Performance Across Scales and Benchmarks

	4B				9B			
Method	SWE-Pro (730)	TB 2.1 (89)	BFCL (800)	LCB (131)	SWE-Pro (730)	TB 2.1 (89)	BFCL (800)	LCB (131)
Base	19.04	23.60	4.63	24.43	29.04	25.84	18.75	**41.22**
PPO	25.89	25.84	5.25	27.48	31.37	28.09	19.00	38.93
Selector (unoptimized)	23.97	26.97	5.25	31.30	33.42	30.34	18.25	38.17
CriticSearch	21.64	22.47	5.13	31.30	30.27	29.21	18.50	35.11
CAPO	21.78	24.72	5.38	29.77	29.04	30.34	19.00	37.40
Ours	**30.41**	**28.09**	**6.13**	**35.11**	**36.16**	**38.20**	**19.13**	**41.22**

Table 1: **Task success rates (%) across model scales.** Parentheses give the number of tasks in each evaluation set; higher is better. Best scores, including ties, are bold. The 4B Base entry uses the proxy reference.

Consistent improvements over trained baselines. Our method has the highest score among the five trained methods in all eight scale–benchmark comparisons (Table 1). On SWE-bench Pro, 4B performance rises from PPO's 25.89% to 30.41%, a 4.52-point gain. At 9B, it rises from 31.37% to 36.16%, a 4.79-point gain. The unoptimized selector scores 23.97% and 33.42%, leaving gaps of 6.44 and 2.74 points to the gold-guided configuration. Both CriticSearch and CAPO adaptations remain below our method at both scales.

Transfer across evaluation tasks. On Terminal-Bench, Ours reaches 28.09% at 4B and 38.20% at 9B, compared with PPO's 25.84% and 28.09%. On LiveCodeBench, Ours scores 35.11% and 41.22%, exceeding the strongest trained competitor by 3.82 and 2.29 points. The 9B score matches the untrained Base reference, rather than improving it. BFCL gains are smaller: 6.13% at 4B and 19.13% at 9B, versus the strongest other trained scores of 5.38% and 19.00%. Thus, transfer benefits vary substantially across tasks; the 0.13-point 9B BFCL difference is a small point-estimate advantage.

Value of developing the scoring configuration. The developed configuration exceeds the unoptimized selector on all four tasks at both scales. Its SWE-bench Pro gains are 6.44 and 2.74 points, and its Terminal-Bench gains are 1.12 and 7.87 points. These results evaluate the complete developed scoring configuration.

5.2 Effect of Scoring-Configuration Optimization

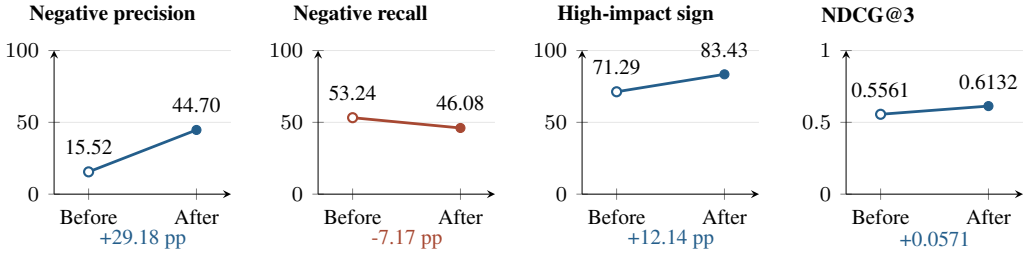


Figure 3: **Scoring-configuration optimization with DeepSeek v4.1 Flash.** Before/after evaluation on 50 trajectories (1,179 turns). The initial configuration uses one call; the optimized configuration uses three evolved prompts and signed max-absolute aggregation. High-impact sign agreement uses turns with $|\widehat{A}^{\mathrm{MC}}| \geq 0.25$, weighted by $|\widehat{A}^{\mathrm{MC}}|$. Deltas are percentage points for the first three panels and absolute NDCG change.

We evaluate complete scoring configurations: one initial call versus three evolved prompts with signed max-absolute aggregation, using the same evaluator and 50 held-out trajectories.

More selective negative credit. MC-negative turns constitute 24.85% (293/1,179) of the evaluation set. Precision rises from 15.52% to 44.70%, above this prevalence, while recall falls from 53.24% to 46.08% (Figure 3). Predicted negatives decrease from 1,005 to 302: a larger share is MC-negative, but fewer MC-negative turns are recovered (Appendix C).

Improved direction and ranking. High-impact weighted sign agreement rises from 71.29% to 83.43%, a 12.14-point gain. NDCG@3 increases from 0.5561 to 0.6132, an absolute gain of 0.0571. These measure credit direction and absolute importance, respectively. On a separate analysis set, sign disagreements concentrate at small MC magnitudes (Section 6.2). The 35B distillation study is in Appendix B.1.

5.3 Ablations and Learning Dynamics

Method	Natural (%)	Cap-pass (%)	Turns	Context tokens
PPO	57.62	2.54	50.25	31,497.84
Full selector	**66.21**	0.59	32.96	19,347.71
Remove success negatives	61.52	3.71	46.79	26,452.12
Remove all negatives	41.60	0.00	15.37	8,402.32

Table 2: **Training success and execution length.** Actor updates 57–64 (eight equally weighted batches, 512 accepted samples per run). Natural: voluntary checker-passing completion; cap-pass: checker success at the turn limit. Context tokens include tool output. Full and both interventions use the same evolved selector configuration. These are training metrics, not benchmark evaluations.

Figure 4 shows Full and PPO through 128 updates and both interventions through 64; Table 2 compares their common updates 57–64. The interventions zero negative advantages either everywhere or only within voluntary successes. Failure/cap-pass negatives remain in the latter. Positive values, masks, and loss normalization are unchanged; no samples are removed.

Higher training success with fewer turns. Full reaches 66.21% voluntary success versus PPO's 57.62%, with 32.96 rather than 50.25 turns (Table 2). These are accepted training rollouts, not evaluation-time cost estimates.

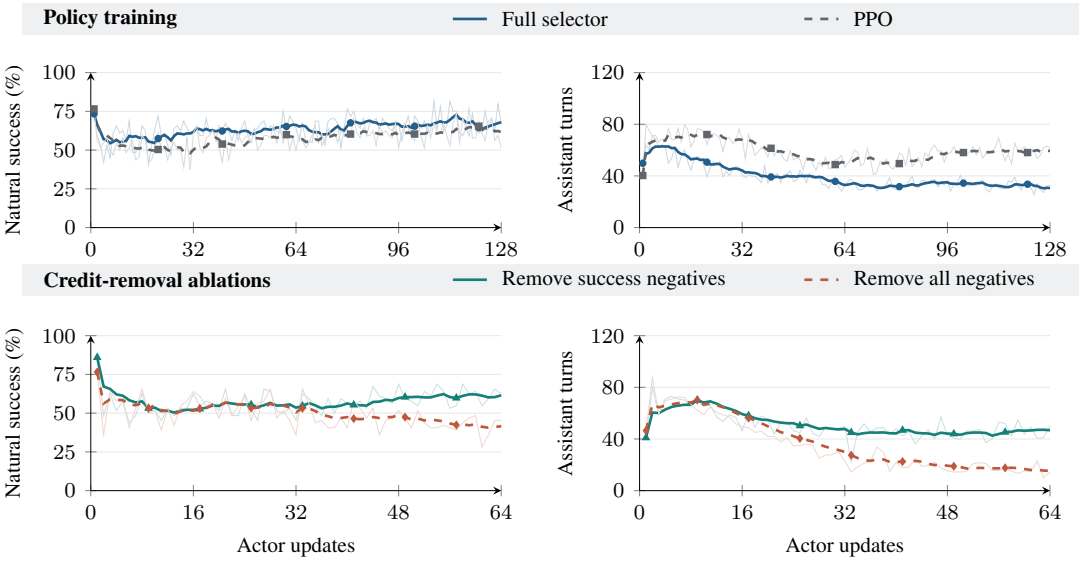


Figure 4: **Learning dynamics and execution length.** Top: Full selector versus PPO. Bottom: removing negative credit only from voluntarily successful trajectories (teal), or from every trajectory (vermilion). Natural success denotes voluntary checker-passing completion. Faint lines are accepted-rollout metrics at each update; dark lines are trailing eight-update means, not confidence intervals. Full and both interventions use the same evolved selector configuration. The main comparison shows the 128-update budget; the interventions show 64 updates.

Success-negative credit and timely completion. Full reaches 66.21% voluntary success versus 61.52% after removing success-negative credit, while total checker success (Natural plus Cap-pass) is 66.80% versus 65.23%. Mean length decreases from 46.79 to 32.96 turns. The larger gap in voluntary completion than in checker success indicates a change in how and when solutions are completed, not only whether they pass. Removing all negative credit yields even shorter trajectories (15.37 turns) but only 41.60% success: shortening alone does not characterize effective learning.

6 Analysis

6.1 Interpreting the Measured Gold Signal

In the gold-analysis collection, 136 of 143 successful trajectories contain at least one MC-negative turn; 567 of their 1,761 turns (32.20%) have negative credit. Final success therefore does not settle local value. These statistics concern the analysis collection, not the 300-trajectory mixed-policy corpus.

In Figure 2, success-probability and efficient-success probability differences contribute -0.65625 and -0.75000 to the -1.40625 label. The observed action is worse on both components, not only length. This selected example illustrates the target rather than corpus-wide attribution.

6.2 Local Correspondence Beyond Outcome and Position

Within-trajectory permutation preserves score values and sign counts but removes their action correspondence. Weighted sign agreement falls from 67.1479% to 58.6562%, and NDCG@3 from 0.534319 to 0.417728. Permuting within each trajectory half retains a coarse early/late prior, yet yields only 59.6206% agreement and 0.452920 NDCG. Credit therefore contains information beyond outcome and coarse position.

The controls use 183 trajectories, 2,238 turns, and 1,000 permutations; NDCG is defined for 177 trajectories. On this support, small-magnitude gold labels ($|\widehat{A}^{\mathrm{MC}}| < 0.10$) account for 60.95% of

turns but 73.46% of sign disagreements (789/1,074; Appendix C). These controls test prediction, not long-run shuffled-credit training.

6.3 Behavioral Judgments Against Gold Credit

Behavior	Turns	MC-negative	Sign agreement (%)	Mean A^{train}
Avoidable errors	158	78	61.39	-0.96313
Redundant actions	244	78	64.75	-0.24278
Diagnosis	747	287	56.76	+0.47249
Repair / recovery	242	62	67.36	+0.74879
Verification / stopping	864	221	68.29	+0.18650
Mixed / other	158	77	58.23	-0.34205

Table 3: **Behavioral credit on the full analysis collection.** 185 trajectories, 184 tasks, and 2,259 distinct turns; overlapping model-assisted labels produce 2,413 memberships. Sign agreement is turn-weighted three-way agreement, including zero; it is not the weighted metric in Figure 3. Mean A^{train} uses the turn-level mapping specified in Appendix C.

Table 3 summarizes the full model-assisted audit. Avoidable errors and redundancy receive negative mean advantages (-0.96313, -0.24278), while diagnosis and repair/recovery receive positive means ($+0.47249$, $+0.74879$). Verification/stopping is also net positive ($+0.18650$).

These are behavioral preferences, not fixed category rules. Only 78 of 244 redundant turns and 287 of 747 diagnostic turns have negative MC credit: semantic role and source-policy-relative value differ. Unweighted three-way sign agreement ranges from 56.76% to 68.29%; unlike Figure 3, it includes near-zero labels and gives every turn equal weight.

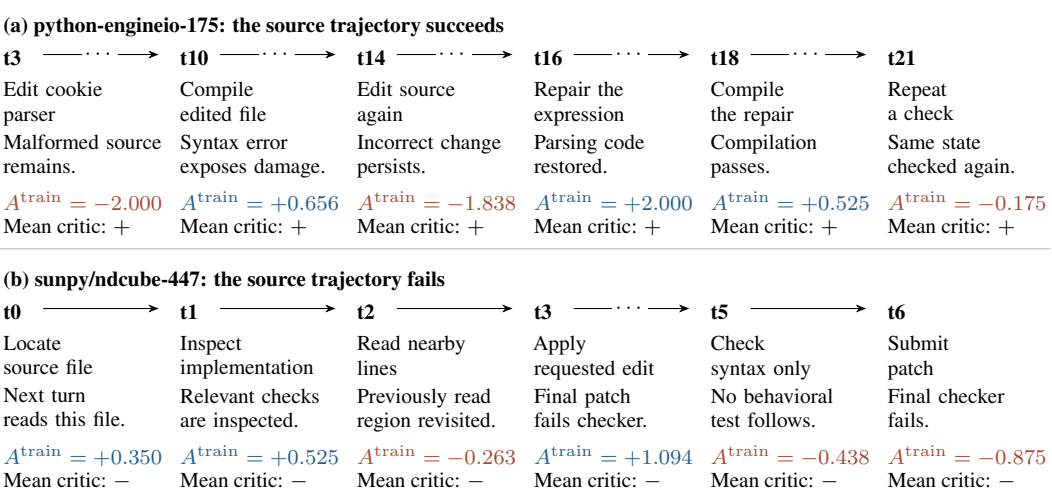


Figure 5: **Causing an error, revealing it, and repairing it are different actions.** Selected turns from two real trajectories, with source-grounded action/evidence summaries. Ellipses mark skipped turns. Selector values use the training-scale mapping. Only critic signs are shown because the two audits use different normalizations. Neither case has an MC label; exact provenance is in Appendix E.3.

Concrete local distinctions. In the successful engineio trajectory, damage receives negative credit, but the compilation exposing it, the repair, and verification receive positive credit. In the failed ndcube trajectory, localization remains positive while repeated reading and unsupported submission are negative. Figure 5 illustrates these within-case judgments, not comparative prediction accuracy: neither case has MC labels, and critic normalizations differ. Only within-case signs are compared.

6.4 From Credit Assignment to Policy Updates

Full, Shuffled, No success-negative, and No negative share each training batch, cached selector scores, and initial actor and optimizer states. Only the supplied credit is changed; evaluator calls are not repeated across branches. Each of four 64-trajectory batches yields one update per branch, with the unchanged actor Pre as reference.

Local probability and action-value probes. One state per held-out task is sampled in proportion to source-turn action-token length. Let $d_{bj}^{(v)}$ be branch v's change from Pre in mean action-token log probability for task j and batch b. Gold from Pre defines

$$S_v = \frac{1}{BJ} \sum_{b=1}^{B} \sum_{j=1}^{J} \widehat{A}_j^{\mathrm{MC}} \, d_{bj}^{(v)}, \qquad B = 4,\ J = 128. \tag{5}$$

Length weighting is induced by state sampling, not applied twice. In the environment probe, each branch chooses the next action and Pre completes all suffixes. Thus ΔU_v measures action-choice utility under one continuation policy, with fixed thresholds and budgets (Appendix E.1).

Branch	S	ΔU vs. Pre	Passed / total	Success (%)	Turns
Pre	0	0	142/256	55.47	31.8
Full	$\mathbf{+0.0014}$	$\mathbf{+0.1035}$	597/1024	**58.30**	28.7
Shuffled	$+0.0002$	$+0.0166$	574/1024	56.05	31.9
No success-negative	$+0.0008$	$+0.0654$	593/1024	57.91	30.2
No negative	$+0.0005$	$+0.0146$	513/1024	50.10	23.7

Table 4: **Matched-policy update probes.** S is gold-weighted log-probability change; ΔU is next-action utility relative to unchanged Pre. Updated branches share Pre continuations. Aggregates cover the same 128 tasks; Pre is reused across four update batches.

Full has the largest measured S and ΔU (Table 4). Its 0.1035 utility gain decomposes into approximately 0.0283 higher success probability and 0.0752 higher efficient-success probability. Shuffling or removing negatives reduces these gains; removing all negatives is shortest but lowers success to 50.10%. These fixed-score comparisons support the structure of the deployed credit at this checkpoint, rather than isolating how it was developed. The off-policy diagnostic in Appendix E.2 measures a different statistic.

7 Discussion and Limitations

Our experiments evaluate the complete execution-supervised scoring configuration and controlled interventions on its outputs. They do not isolate MC-guided development from changes in prompts and inference multiplicity, or every effect of local credit from the joint correctness–efficiency objective. Max-absolute aggregation and clipping can still retain erroneous signs.

Mixed-policy development and reuse across actor scales define the tested setting, not guaranteed accuracy for unseen policies. Behavioral labels are model-assisted; single-run training and local probes leave uncertainty about seed variability, paired effects, and fully updated continuations. Avoiding online counterfactual branching does not itself establish lower end-to-end compute.

8 Conclusion

We connect execution-derived gold supervision to critic-free turn-level learning through a shared selector. Its predictions supply the full actor advantage without online branching. Across both scales, the method exceeds the trained baselines on all four evaluation sets. Interventions on its outputs provide local evidence about credit placement and negative feedback, distinct from the performance of the complete configuration.

AI USE STATEMENT

A Codex root agent proposes selector-prompt revisions using gold-based development feedback. Generative models produce coding trajectories, counterfactual continuations, and selector assessments in the experiments. ChatGPT was also used for literature discovery, research discussion, organization, drafting, editing, LaTeX, and figure preparation. Behavioral labels include model-assisted annotation with evidence checks. The authors are responsible for verifying the final claims, implementations, citations, and numerical records.

ETHICS STATEMENT

The experiments execute model-generated code in sandboxed repositories. Passing task tests or receiving positive selector credit does not certify security or deployment safety. Efficiency-sensitive feedback may undervalue necessary verification. Sharing source code, trajectories, or model outputs requires artifact-specific permissions and removal of credentials, private infrastructure information, and identifying metadata. Public artifacts are released subject to these checks; test success is not a security certification.

REPRODUCIBILITY STATEMENT

Section 3 defines the target, selector, and actor objective. The appendices specify corpus provenance, partitions, evaluation metrics, known run configurations, probe references, and the exact prompt files. The source files reproduce figures and tables from the stored summaries and per-update records. The worked MC example is calculated from candidate outcome counts. Code and data required to rebuild this manuscript are included in its source package. The anonymous code repository is `https://anonymous.4open.science/r/counterfactual-turn-credit-iclr2027-70C2/`. Appendix D specifies the agent adapters and their version records.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024. doi: 10.48550/arXiv.2402.14740. URL https://arxiv.org/abs/2402.14740.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. SWE-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0788. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/21bec6ace947b1b58967b945c8ac0f10-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025. doi: 10.48550/arXiv.2502.10325. URL https://arxiv.org/abs/2502.10325.

DeepSeek. Introducing DeepSeek-V4.1-Flash: Smarter, faster, more efficient. Official model release, September 2026. URL https://www.deepseek.com/en/news/deepseek-v4-1-flash/.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025. doi: 10.48550/arXiv.2509.16941. URL https://arxiv.org/abs/2509.16941.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL https://doi.org/10.1145/582415.582418.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590. PMLR, 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327. Association for Computational Linguistics, November 2016. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

OpenAI. Introducing Codex. Official product announcement, May 2025. URL https://openai.com/index/introducing-codex/.

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

Qwen Team. Qwen3.5: Towards native multimodal agents. Official model release, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347. URL https://arxiv.org/abs/1707.06347.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. *arXiv preprint arXiv:2410.08146*, 2024. doi: 10.48550/arXiv.2410.08146. URL https://arxiv.org/abs/2410.08146.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. ISBN 9780262039246. URL https://mitpress.mit.edu/9780262039246/reinforcement-learning/.

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. Hindsight credit assignment for long-horizon LLM agents. *arXiv preprint arXiv:2603.08754*, 2026. doi: 10.48550/arXiv.2603.08754. URL https://arxiv.org/abs/2603.08754.

The Terminal-Bench Team. Terminal-Bench 2.1. Official release announcement, May 2026. URL https://www.tbench.ai/news/terminal-bench-2-1.

Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. CAPO: Towards enhancing LLM reasoning through generative credit assignment. *arXiv preprint arXiv:2508.02298*, 2025. doi: 10.48550/arXiv.2508.02298. URL https://arxiv.org/abs/2508.02298.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3339f19c5fcee3ad74502947a32be9e6-Abstract-Conference.html.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, Qiaobo Chen, Yinyuting Yin, Hao Zhang, Tengfei Shi, Liang Wang, Qiang Fu, Wei Yang, and Lanxiao Huang. Mastering complex control in MOBA games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6672–6679, 2020. doi: 10.1609/aaai.v34i04.6144. URL https://ojs.aaai.org/index.php/AAAI/article/view/6144.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Zijie Zhao, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. CriticSearch: Fine-grained credit assignment for search agents via a retrospective critic. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12272–12290, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.596. URL https://aclanthology.org/2026.findings-acl.596/.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. SWEET-RL: Training multi-turn LLM agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025. doi: 10.48550/arXiv.2503.15478. URL https://arxiv.org/abs/2503.15478.

A Counterfactual Annotation and Data Coverage

Mixed-policy development and evaluation. The mixed gold corpus contains 300 trajectories and 5,192 assistant turns from 4B and 9B checkpoints before and after RL. We randomly divide these trajectories into 250 for development (4,013 turns) and 50 for evaluation (1,179 turns). All selector before/after metrics in Figure 3 and Appendix B use the same 50-trajectory evaluation partition. The source policy remains fixed within each counterfactual label. The selector receives no source-model name, model scale, checkpoint, or training-step identifier.

The behavior-audit collection contains 185 trajectories, 184 tasks, and 2,259 distinct turns. Locality analysis uses the 183 trajectories with jointly supported predictions and labels, comprising 2,238 turns. These are separate analysis supports, not alternative counts for the 300-trajectory mixed corpus. The initial audit sources included 100 successful trajectories of at most 20 turns; later sources also included failures. This selection is relevant to the scope of that audit, not a restriction imposed on all mixed-policy annotations.

Collection / partition	Trajectories	Turns	Use
Mixed gold corpus	300	5,192	Shared selector
Development	250	4,013	Scoring optimization
Evaluation	50	1,179	Before/after scoring
Behavior-audit collection	185	2,259	Behavioral analysis
Locality support	183	2,238	Permutation controls

Table 5: Data roles. The random development/evaluation split is made within the mixed corpus. Partition rows are nested; the analysis collections have distinct roles and are not added to the mixed corpus.

Replay and branch allocation. For every annotated turn we restore the ordered messages, workspace changes, patch, tool state, and remaining budget. The fixed action is inserted before continuation, including its reasoning and tool-call messages. The horizon is $H = 120$ total assistant turns; the available continuation budget is reduced by the restored prefix and fixed action. Both the mixed corpus and the analysis annotations use $K = 12$ same-policy alternatives and $M = 16$ continuations per fixed action, including the original. These budgets balance annotation cost and the stability of the empirical comparison. Duplicate sampled alternatives are not deduplicated. For the 185-source analysis records, eight hinted candidates and a masked-observation arm are auxiliary interventions; they do not enter the baseline in Equation 2.

The 185-source annotation corpus requests 792,208 continuations across all arms and retains 792,186 valid semantic outcomes. There are 178 complete sources and seven available-case sources. The 22 excluded records are infrastructure failures, not task failures; affected candidates retain at least 14 of 16 planned continuations. Each candidate value uses its valid denominator, with the baseline then averaging equally over the 12 sampled action slots. Available-case averaging does not establish ignorable missingness.

Efficiency reference. The reference length $\bar{T}$ uses the 12×16 root-state base continuations for the same task and annotation policy. It excludes the original action, hints, masked branches, and later-state alternatives. The floating-point reference is shared across turns. Length includes restored prefix and fixed action, while non-voluntary completion is assigned horizon length. Utility zero represents checker success without sufficiently efficient voluntary completion, not an incorrect solution.

Behavior audit. The behavior audit covers all 2,259 assistant turns in 185 trajectories (184 tasks), using model-assisted dual annotation and full-context review. The six overlapping categories contain 2,413 memberships. These are neither additional independent turns nor labels from the new mixed corpus.

Per-category MC-negative counts, three-way sign agreement, and mean training advantages are reported in Table 3. Zero is a separate sign class, and the audit weights turns equally. The metric definitions are given in Appendix C.

A.1 Worked Label: Candidate-Level Outcomes

The source is `showyourwork-456`, turn 4 (zero-indexed). The original action overwrites `clean.py`; Figure 2 shows the relevant loop excerpt. The table below retains all candidate slots, rather than displaying only a winning alternative. All actions have 16 valid continuations in this example. Equal candidate counts make the pooled alternative mean equal to the average of candidate means.

Fixed action	$U=-1$	$U=0$	$U=+1$	$\widehat{Q}$
Original	11	2	3	-0.5000
Alternative 1	1	1	14	$+0.8125$
Alternative 2	0	0	16	$+1.0000$
Alternative 3	3	2	11	$+0.5000$
Alternative 4	1	3	12	$+0.6875$
Alternative 5	0	0	16	$+1.0000$
Alternative 6	1	0	15	$+0.8750$
Alternatives 7–12 (each)	0	0	16	$+1.0000$
Alternative total	6	6	180	$+0.90625$

Table 6: Observed continuation counts behind the worked MC label. The last mean is the policy-sampled baseline, not a best-of-K score.

The source case was selected for large absolute MC credit within a specified agreement pattern. It is an illustration, not a randomly sampled accuracy audit. The selector's reason concerning generator slicing is a model explanation, not independent execution evidence used to calculate the MC label. The quantitative label is obtained from the continuation counts: $-0.5 - 0.90625 = -1.40625$. Candidate counts are provided in `data/gold_example_counts.csv`.

For the separate MC119 probe collection, an audit verifies the label identity for all admitted turns. Its conditional empirical standard error has median 0.0921, conditioning on fixed actions and the length reference. It excludes uncertainty in alternative generation, the estimated reference length, and other dependence. Neither a selected high-magnitude example nor this conditional diagnostic establishes that every annotation is exact.

B Prompt Development and Scorer Comparisons

Inputs and role separation. The 250-trajectory development partition supplies the feedback used to revise the scoring instructions. The other 50 trajectories are reserved for evaluation of the resulting configurations (1,179 turns). Each scoring request contains the task/sample identifier, terminal checker state, legal turn IDs, and complete trajectory with tool observations. Counterfactual branches, MC labels, baseline values, and source-policy identifiers are not included in the scoring request. MC gold is used outside the scorer to evaluate its predictions.

Prompt optimizer. A root agent in Codex (OpenAI, 2025) proposes prompt revisions. Its feedback includes the environment-derived $\widehat{A}^{\mathrm{MC}}$, NDCG@3, absolute-credit mass@3, top-three overlap, sign agreement, MAE, and Pearson correlation. Development involved approximately 13 prompt versions over 15 revision rounds. These metrics track complementary errors in direction, magnitude, and ranking. Evaluator weights remain fixed while the instructions are revised.

Initial and retained prompts. The simplified initial instruction asks for $\widehat{A} = Q(\text{observed action}) - \mathbb{E}[Q(\text{ordinary same-policy alternative})]$ for every assistant turn, with a turn identifier and brief reason, and uses $|\widehat{A}|$ to identify important turns. Its original Chinese text is provided in `prompts/selector_before_original_zh.txt`. The retained configuration uses the unchanged v5, v6, and v8 prompt files in `prompts/`; Appendix F describes the input/output contract.

Scoring and aggregation. Both configurations use DeepSeek v4.1 Flash. Full makes three requests per trajectory, one with each of v5/v6/v8. For each turn, the member with maximum absolute

score supplies the signed output; ties follow the fixed v5, v6, v8 order. We then apply the score multiplier $\eta = 0.35$. This selects a score, not a subset of turns, and is not majority voting. The scorer weights, selected prompts, and aggregation are held fixed throughout actor training. The simplified baseline makes one request per trajectory and uses its single set of turn scores without cross-prompt aggregation. Prompt revision, inference multiplicity, and aggregation are all part of the scoring-configuration optimization. Both configurations receive the same input format; their comparison is not an equal-call prompt-only ablation.

Configuration	Calls	Neg. precision	Neg. recall	High-impact sign	NDCG@3
Before optimization	1	15.52	53.24	71.29	0.5561
After optimization	3	44.70	46.08	83.43	0.6132

Table 7: Scoring configurations on the same 50 evaluation trajectories (1,179 turns), separate from the development partition. Precision, recall, and sign agreement are percentages. The comparison includes prompt content, inference multiplicity, and aggregation. Precision, sign agreement, and ranking improve, while recall decreases.

B.1 Distilling an Alternative Evaluator

We also explore whether teacher-generated assessments can be transferred to the separately deployed Qwen3.5-35B-A3B evaluator. This changes model weights and is distinct from prompt-space development. The recorded experiment uses 443 valid teacher views from 148 training trajectories and 111 requests from 37 development trajectories. Supervision covers the teacher's reasoning and final all-turn JSON, with the two text losses normalized separately and equally weighted. It follows teacher-generated sequence supervision (Kim & Rush, 2016), rather than numerical MSE on advantages. Training uses two epochs, 112 optimizer updates, batch size eight, and a learning rate of 10^{-5}.

Evaluator	Neg. precision	Neg. recall	High-impact sign	NDCG@3
35B before distillation	39.39	8.28	62.59	0.5796
35B after distillation	40.56	36.94	74.39	0.5345

Table 8: Recorded 35B comparison on 36 complete development trajectories and 459 turns. Percentages in the first three columns. This is a separate deployment study, not the 50-trajectory evaluation of prompt optimization.

Distillation improves negative-credit recall and high-impact sign agreement but decreases absolute-credit ranking quality. It therefore provides partial transfer rather than a uniformly stronger evaluator. The main training experiments therefore retain the external selector; distillation is treated as a separate deployment study.

C Credit Metrics and Analysis Protocols

Direction and numerical agreement. Negative credit uses $\widehat{A}_t^{\mathrm{MC}} < 0$ as the reference class. For negative-class precision and recall, TP counts turns for which both gold and prediction are negative, FP counts predicted negatives with nonnegative gold, and FN counts gold negatives with nonnegative predictions. Precision is $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$ and recall is $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$; undefined denominators are reported as missing. Zero is not a negative prediction. Counts are pooled over the same 1,179 evaluation turns. Gold contains 293 negative and 886 nonnegative turns. Table 9 gives the two confusion matrices underlying Figure 3. High-impact sign agreement considers $|\widehat{A}_t^{\mathrm{MC}}| \geq 0.25$, weighting each sign comparison by $|\widehat{A}_t^{\mathrm{MC}}|$. All-turn weighted agreement uses all supported turns instead. MAE compares the stated selector prediction with MC credit before actor-side gain and clipping.

Ranking. NDCG@3 (Järvelin & Kekäläinen, 2002) ranks turns by absolute predicted credit and uses absolute MC credit as relevance. Scores are averaged across trajectories with nonzero ideal

Configuration	TP	FP	FN	TN	Precision (%)	Recall (%)
Before optimization	156	849	137	37	15.52	53.24
After optimization	135	167	158	719	44.70	46.08

Table 9: Pooled negative-credit classification on 50 trajectories and 1,179 turns. Both rows use the same gold labels. Zero gold or predicted credit belongs to the nonnegative class for these two metrics.

gain. The locality analysis has 177 such trajectories and six zero-IDCG trajectories. Positive uniform scaling leaves signs and ranking unchanged, but can change MAE.

Locality controls. Within-trajectory permutation shuffles the score vector among turns of the same trajectory. Within-half permutation does this independently in its first and second halves, retaining a coarse position prior. Both preserve the turn-level score multiset and terminal outcome. We recompute the metrics for 1,000 permutations of each control.

Assignment	Weighted sign (%)	NDCG@3
Original	**67.1479**	**0.534319**
Within-trajectory permutation	58.6562	0.417728
Within-half permutation	59.6206	0.452920

Table 10: Locality controls on 183 trajectories and 2,238 turns. The within-half NDCG conditional permutation range is [0.424399, 0.481542]; it is not a confidence interval over new tasks or training seeds.

Disagreements by gold magnitude. On the 183-trajectory locality support, 1,364 of 2,238 turns have $|\widehat{A}^{\mathrm{MC}}| < 0.10$ and account for 789 of the 1,074 three-way sign disagreements. Conditional disagreement rates are 57.84% below this threshold and 32.61% outside it. Exact zero remains a separate sign; no small score is recoded as zero. This audit uses the same recorded selector scores as the locality analysis, not the 50-trajectory scoring comparison. Gold magnitude and actor-loss weight are different quantities. The accompanying source and bin counts permit direct recomputation.

Behavior-level metrics. The model-assisted audit covers 185 trajectories, 184 tasks, and 2,259 unique turns, with 2,413 overlapping category memberships. For category g, sign agreement is the proportion of its turns satisfying $\mathrm{sign}(\widehat{A}_t^{\mathrm{sel}}) = \mathrm{sign}(\widehat{A}_t^{\mathrm{MC}})$. Negative, zero, and positive are distinct, and turns have equal weight. The category mean training advantage averages $\mathrm{clip}(0.35 \cdot 12.5\, s_t, -2, 2)$ over its turns, where s_t is the signed maximum-absolute prompt score. It is not a category-level gradient contribution or a token-weighted statistic. Categories can overlap and do not determine the MC sign.

Execution proxies. The early–late comparison uses 32 selected task pairs. It records assistant turns, nonzero tool exits, anchored syntax diagnostics, recognized test/compilation calls, and exact command–output repetitions. Mean turns change from 58.97 to 50.25; nonzero exits from 16.69% to 10.05%; syntax diagnostics from 2.23% to 0.82%. Test/compilation calls change from 220 to 245, and exact repetitions from 50 to 75. These are command-level proxies on selected pairs, not semantic labels for every training trajectory.

Training-curve smoothing. Figure 4 shows actor updates 1–128 for Full and PPO and 1–64 for each credit-removal intervention. The horizontal axis counts actual actor updates; update indices are not rescaled. Pale lines show batch means, while heavier lines show trailing eight-update means, using shorter prefixes at the start. These are descriptive smooths, not uncertainty intervals. Table 2 uses the common updates 57–64, weighting its eight batches equally.

D TRAINING AND EVALUATION IMPLEMENTATION

Training system and budgets. The actors use the `mini_swe` scaffold. The full 4B and 9B training schedules contain 128 actor updates, with 64 accepted trajectories per update. PPO performs 128 critic-only warm-up updates before actor learning; these are excluded from actor-update counts. The two negative-credit ablations run for 64 actor updates and are compared over the shared range. Main benchmark results use a training-score-selected checkpoint within the 128-update budget, as specified in Appendix D.2.

Optimization. We use AdamW (Loshchilov & Hutter, 2019). Table 11 reports the shared 4B/9B optimizer and batching configuration. The micro-batch value is a per-GPU setting with dynamic batching enabled, not a claim that every runtime micro-batch has identical token length. The learned-critic PPO baseline uses GAE (Schulman et al., 2016) with $\gamma = \lambda = 1$ and action-token advantage whitening. Selector-only training has no value critic, GAE, additional outcome-advantage term, KL loss, or entropy bonus.

Setting	4B and 9B
Optimizer	AdamW; $\beta_1 = 0.9$, $\beta_2 = 0.999$
Weight decay	0.01
Learning rate / schedule	10^{-6}; two warm-up updates; cosine decay
Gradient-norm clipping	0.5
Trajectories per actor update	64
Mini-batch size	64 trajectories
Configured micro-batch per GPU	1; dynamic batching enabled
PPO epochs per batch	1
Full actor-training schedule	128 updates
Data / actor minibatch / FSDP seeds	42 / 42 / 42
Prompt multiplier / actor gain	$\eta = 0.35$ / $g = 12.5$
Advantage bound	$A_{\max} = 2$
PPO ratio interval / dual clip	$[0.8, 1.2]$ / $d = 2$

Table 11: Shared training settings. The two-update learning-rate warm-up is separate from PPO's 128 critic-only warm-up updates. Seeds refer to the listed training modules, not external API generations.

Rollout and scoring inference. Training rollouts use temperature 1.0, top-$p = 1.0$, top-$k = -1$, and $n = 1$. The total context window is 102,400 tokens, with a prompt budget of 16,384 and response budget of 81,920 tokens. Each callback has an output cap of 32,768 tokens, and trajectories have at most 120 assistant turns. These are configured limits, not measured average lengths. They describe training rollouts, not the separate benchmark budgets.

The scoring configuration uses DeepSeek v4.1 Flash with temperature zero, thinking enabled, and an output cap of 65,536 tokens. Each Full trajectory is scored once with each of v5/v6/v8. Scoring receives the complete recorded history: reasoning, tool calls, and actual tool returns. There is no client-side summarization, compression, truncation, or deduplication of that scoring history. This input contract is shared by the full method and the two negative-credit ablations.

Dual clipping. Following dual-clip PPO (Ye et al., 2020), starting from ℓ_ϵ in Equation 4, the implemented per-token loss is

$$\widetilde{\ell}(\rho, A) = \begin{cases} \min\{\ell_\epsilon(\rho, A), -dA\}, & A < 0, \\ \ell_\epsilon(\rho, A), & A \geq 0. \end{cases} \tag{6}$$

The actor loss averages this quantity over action tokens per trajectory, then across trajectories. Advantage magnitude clipping and probability-ratio clipping are different operations. The fixed gain was selected from a pre-training shadow audit, not recomputed by batch; the prompt multiplier has separate development provenance.

Setting	4B	9B
Actor	Qwen3.5-4B	Qwen3.5-9B
GPU allocation	16 H20	16 H20
Training / rollout GPUs	8 / 8	8 / 8
Training parallelism	FSDP2	FSDP2
Sequence-parallel size	8	8
Rollout tensor-parallel size	1	2
Agent scaffold	`mini_swe`	

Table 12: Hardware and parallelism. Shared hyperparameters are listed in Table 11; only rollout tensor parallelism differs between the two scale-specific configurations.

Runtime and alignment. Action masks and turn IDs are saved during rollout and rechecked before broadcasting scores. Tool returns remain context but do not enter direct policy loss. Every prompt output must cover each legal turn exactly once; an unresolved scorer failure is not converted into zero credit. Asynchronous accepted batches may contain older rollout-policy versions. Token ratios use the recorded generating probability.

Full and the credit-removal interventions use the same scoring configuration and full-history input contract described above. Temperature zero and the training seed do not guarantee identical external API responses.

Negative-credit interventions. The two 64-update runs use the same evolved DeepSeek v4.1 Flash configuration as Full. No-negative sets every negative advantage to zero. No-success-negative does so only for voluntary checker successes, retaining negative credit on failures and cap-pass trajectories. Positive credit, token masks, and loss denominators are unchanged. The shared window is actor updates 57–64, with eight batches and 512 accepted samples per run. Run identifiers are retained in the accompanying data.

D.1 Process-Feedback Baseline Adaptations

CriticSearch (Zhang et al., 2026) and CAPO (Xie et al., 2025) are adapted to the same coding-agent framework. Generative judges use DeepSeek v4.1 Flash and receive the task, trajectory, tool feedback, and terminal checker outcome, without a reference repair patch. Assistant turns are the process units, and credit is broadcast only to their action tokens. The following reward constructions specify our implementations; they are not unchanged executions of the original task pipelines.

CriticSearch. We sample four trajectories per task. A retrospective binary judgment $b_{it} \in \{0, 1\}$ identifies each turn as Good or Bad. Local credit is

$$c_{it} = \frac{b_{it}}{\sum_u b_{iu} + 10^{-6}}, \qquad A_{ik} = 0.25 c_{i,t(i,k)} + 0.75 z_i, \tag{7}$$

where z_i standardizes the terminal environment score within the same-task group using the sample standard deviation. The local component is nonnegative; the outcome component supplies signed trajectory-level feedback. No selector gain, additional advantage clipping, or whitening is applied after this combination. The shared policy-ratio clipping is retained. If the group's terminal scores have zero standard deviation, we set $z_i = 0$ without discarding the group. Local credit c_{it} is retained; the combined advantage is zero only when that local component is also zero.

CAPO. We use the token-wise leave-one-out path, adapting the REINFORCE leave-one-out baseline (Ahmadian et al., 2024) to process-shaped token rewards, using $G = 8$ trajectories per task. A single Flash critique of a checker-failing trajectory identifies process errors; only its earliest erroneous turn receives a penalty. Checker-passing trajectories receive no process penalty. Let R_i^{env} be the terminal environment score. The binary outcome is $y_i = \mathbf{1}\{R_i^{\text{env}} = +1\}$: voluntary success receives one, while failure and reaching the turn cap receive zero, including capped trajectories that pass the checker. With e_{it} indicating the penalized turn, token rewards are

$$r_{ik} = y_i - 0.05 e_{i,t(i,k)}. \tag{8}$$

The advantage subtracts the mean reward of the other same-task trajectories, averaged over the padded response width, before applying the action-token mask. There is no standard-deviation normalization. Exhausted judge-request or parsing failures use outcome-only feedback and are recorded.

D.2 Evaluation Protocol and Checkpoint Selection

The fixed evaluation sets comprise 730 SWE-bench Pro tasks, 89 Terminal-Bench 2.1 tasks from the `mini_swe_v2` subset, 800 cases identified by the evaluation harness as BFCL v4 Agent v3, and 131 LiveCodeBench code-generation tasks. These identifiers refer to the selected task sets rather than the complete BFCL leaderboard or every task in a LiveCodeBench release. They are shared across all methods and both actor scales. For benchmark b,

$$\text{Success}_b(\pi) = \frac{100}{N_b} \sum_{q=1}^{N_b} \mathbf{1}\{\pi \text{ passes task } q\}. \tag{9}$$

Each denominator N_b is the full task count stated above. Benchmark rates are distinct from the accepted training-batch measurements in Table 2.

Task rates and percentage-point differences are computed before rounding to two decimal places, using round-half-up.

Checkpoint selection. Candidates are saved every 16 actor updates through update 128. Each method and scale uses the candidate with the highest unsmoothed batch-mean training environment score at its corresponding update. The 9B selector-trained model uses update 64; all other trained entries use update 128. That single selected checkpoint is used for all four benchmarks, and benchmark scores do not enter selection. Base references are not trained or selected. PPO's 128 critic-only warm-up steps are additional to the actor budget; equal actor updates do not imply equal total compute. Checkpoint directory labels are not substituted for these actor-update counts.

Evaluation decoding. All four suites use temperature 1, top-$p = 1$, configured top-$k = 0$, a 262,144-token serving context, and a 32,768-token generation cap. The evaluation configuration requests one sample per task. These settings differ from the 102,400-token training window and the training decoder's top-$k = -1$ setting. Table 13 gives the suite-specific limits.

Suite	Prompt / generation	Execution limits
SWE-bench Pro	69,504 / 32,768 tokens	120 turns; agent timeout 86,400 s; evaluation timeout 2,700 s
Terminal-Bench 2.1	Prompt unset / 32,768 tokens	4,096 agent rounds (`steps=4096`); task timeout 86,400 s; grading timeout 3,600 s
BFCL subset	64,000 / 32,768 tokens	Tool/turn budget 120; Exercise `steps=4096`; thinking disabled for all evaluated tasks and checkpoints
LiveCodeBench subset	32,768 / 32,768 tokens	Single code generation and grading; no tool-interaction loop; timeout 3,600 s

Table 13: Benchmark inference settings. Prompt limits are evaluator-entry settings, distinct from the 262,144-token service context. Terminal-Bench uses a 4,096-round limit; BFCL retains its 120 tool/turn budget. These suite-specific limits do not change the 120-turn training horizon.

Retry handling and task outcomes. Each SWE-bench Pro task has one solution trial, set by `trial_num=1`. Technical recovery is limited by `eval_max_attempts=3`; it addresses infrastructure or service errors, not unsuccessful solutions. The first valid task verdict is final; a checker failure does not trigger another solution sample. Thus these settings do not implement best-of-three evaluation. Docker commands allow two retries and sandbox startup allows three. Model API calls retry HTTP 429, 502, 503, and 504 responses at most ten times. Such retries do not contribute additional scored task outcomes.

BFCL sets `enable_thinking=false` at the evaluation-task level. This applies to every item in the selected BFCL task set, for both actor scales and every evaluated checkpoint.

Agent software and release artifacts. Terminal-Bench uses the `mini_swe_v2` adapter from branch `fix/tb21-portal-recreate`. SWE-bench Pro uses its `mini_swe_agent` adapter branch. Both installations specify `commit_sha: latest`; immutable agent revisions were not recorded. An evaluation-harness revision does not identify either agent. The anonymous repository is linked in the Reproducibility Statement.

Scope of the interventions. The sustained credit interventions remove all negative values or only success-negative values under Full's scoring configuration. Their 64-update records are separate from offline permutation analysis and one-update probes; they are not long-run shuffled-credit training.

E Update Probes and Additional Cases

E.1 Matched-Policy Update Probes

The two probes separate fixed-action probability changes from the utility of newly selected actions. Both use the same pre-update actor and held-out task states. Table 4 reports their aggregate measurements.

States, updates, and reference. Pre is a fixed Qwen3.5-4B checkpoint. We use 128 distinct held-out tasks, each supplying one intermediate state from a Pre-generated trajectory. The source turn is sampled with probability proportional to its action-token count. Four independent training batches contain 64 trajectories each, from tasks separate from evaluation. Each batch produces four branches: Full, Shuffled, No success-negative, and No negative. Every branch starts from the same saved actor, optimizer, scheduler, and random-number-generator states and applies one update; the four batches are not sequential training steps. Pre is the unchanged actor, not a zero-advantage optimizer step.

Each batch is scored once with DeepSeek v4.1 Flash and v5/v6/v8. All branches reuse these cached scores, with the normal signed max-absolute, scaling, and clipping operations. Branches use the next learning rate from the restored scheduler and the same minibatch, mask, epoch, and loss-reduction settings. Shuffled permutes turn scores within each trajectory. This retains the turn-score multiset, but not necessarily token-weighted credit mass when turns differ in length. The two removal branches zero negative credit either only on voluntary successes or on all trajectories, respectively.

Matched-policy gold and probability changes. For each selected state we sample four alternatives from Pre and perform four continuations after each fixed action, including the original. This probe-specific budget ($K = 4, M = 4$) is separate from the annotation-corpus budget ($K = 12, M = 16$). The efficiency reference is fixed independently before evaluating the branches. Probability changes use identical contexts, action-token IDs, and masks before and after each update, with gold measured under Pre.

Equation 5 averages tasks and update batches equally. Sampling turn t with probability n_{it}/N_i and averaging $\widehat{A}^{\mathrm{MC}}_{it} d_{it}$ accounts for the turn's action-token fraction in expectation. Multiplying by this fraction again would apply the weight twice. S is a gold-weighted probability-change diagnostic, not a covariance or a policy-gradient guarantee.

One-action replacement in the environment. At each held-out state, an updated branch chooses one complete next action; all subsequent turns are generated by Pre. We execute two repeats per task and branch for each update batch. Pre supplies two executions per task and is shared across batches. Consequently, Pre has 256 executions and each updated branch has 1,024, all covering the same 128 tasks. The repeated executions are not additional independent tasks.

Student decoding uses temperature 1, top-$p = 1$, top-$k = -1$, a 102,400-token context window, and at most 120 assistant turns including the restored prefix. Checkers, efficiency thresholds, and budgets are common across branches. Utility is -1 for failure, $+1$ for efficient voluntary success, and zero for other successes. Total turns include successes and failures, not only successful completions. With two repeats per state, the equally weighted effect is

$$\Delta U_v = \frac{1}{J} \sum_{j=1}^{J} \left[\frac{1}{2B} \sum_{b=1}^{B} \sum_{r=1}^{2} U^{(v)}_{bjr} - \frac{1}{2} \sum_{r=1}^{2} U^{(\mathrm{Pre})}_{jr} \right]. \tag{10}$$

The comparison measures current action selection under shared Pre continuations, not the value of replacing the actor at every future turn.

Measured outcomes and scope. Pre passes 142 of 256 executions. Full, Shuffled, No success-negative, and No negative pass 597, 574, 593, and 513 of 1,024, respectively. Their S values are 0.0014, 0.0002, 0.0008, and 0.0005; the corresponding ΔU values are 0.1035, 0.0166, 0.0654, and 0.0146. Full has the largest aggregate improvement in both quantities. Removing all negatives yields the shortest executions (23.7 turns) but a lower success rate than Pre, consistent with the correctness–efficiency trade-off of the joint utility.

Results are aggregate point estimates without paired uncertainty estimates. The shared Pre executions are counted once, rather than as independent observations for each update batch. The probe-specific finite-sample gold and tested state distribution limit extrapolation to other checkpoints. The outcome counts and means are stored in `data/probe_rerun_summary.csv`.

E.2 Auxiliary Off-Policy Diagnostic

A separate completed diagnostic uses four fixed update batches and an off-policy, retokenized MC collection of 119 trajectories, 118 tasks, and 1,450 turns. It measures trajectory-equal covariance between MC credit and changes in mean action-token log probability. This is not the matched-policy experiment above. Zero applies zero advantages but still executes an optimizer step and changes the weights; the unchanged saved actor supplies the absolute reference.

Covariance / difference ($\times 10^{-6}$)	Batch 1	Batch 2	Batch 3	Batch 4
Full minus token shift	-5.244	$+8.124$	$+15.671$	$+1.235$
Full minus no-success-negative	-7.244	$+8.089$	$+8.005$	$+14.973$
Full versus Zero-step	-17.988	-4.551	$+4.238$	$+6.073$
Full versus true pre-update actor	-26.230	-12.793	-4.004	-2.169

Table 14: Complete off-policy diagnostic. Relative differences and absolute alignment answer different questions. The absolute covariances are negative in all four batches; these observations do not establish positive MC alignment or measure environment-return improvement.

Repeated outputs establish repeatability, not the numerical precision of these small covariances. The diagnostic uses BF16 forward computation, float32 returned log probabilities, and float64 turn aggregation. Its conditional MC-label standard-error median is 0.0921 and excludes alternative-generation and reference-length uncertainty. These limitations restrict interpretation of the absolute covariance values.

E.3 Case Provenance

In Figure 5(a), `python-engineio-175` is a 24-turn checker-passing PPO trajectory. The displayed selector is an offline Flash three-prompt replay mapped by $\mathrm{clip}(0.35 \cdot 12.5\,s, \pm 2)$. Critic values are critic268 mean raw $R - V$, without whitening; they are not recovered production-batch GAE. No MC labels exist for this case. Additional selected turns are:

Turn	Observed role	Selector training coefficient	Critic raw $R - V$
3	Damaging edit	-2.000	$+0.068$
10	Compilation exposes damage	$+0.656$	$+0.071$
14	Further damaging edit	-1.838	$+0.096$
16	Correct repair	$+2.000$	$+0.100$
18	Passing compilation	$+0.525$	$+0.075$
21	Repeated check	-0.175	$+0.094$
22	Repeated read	-0.175	$+0.076$

Table 15: Additional evidence from the same engineio trajectory. Turn IDs are zero-based.

Figure 5(b) uses `sunpy/ndcube-447`, a failed seven-turn trajectory. The positive localization label at turn zero has direct evidence: the next turn reads the located file. Displayed selector advantages use the recorded scores and training-scale mapping, whereas critic values are whitened shadow GAE from actor128 trajectories and critic268. The two cases therefore do not use a common critic scale, and neither evaluates an actual update.

A second successful error–recovery case. In `python-redpocket-4`, a sed replacement fails to escape an ampersand, damaging the URL; deterministic replay confirms the string transformation. The damaging turn receives selector advantage -2.00, while the later correctly escaped repair receives $+1.75$. Their whitened shadow-GAE means are $+0.3832$ and $+0.3238$. This supports a concrete sign disagreement, not the claim that critics can never distinguish local actions.

Execution length and verification. For the `python-engineio-175` comparison, a 24-turn PPO path includes malformed edits and repeated recovery, whereas a five-turn selector-trained path locates, edits, reads back, compiles, and submits; both pass. In `textual-4464`, the selector-trained actor completes in 42 turns while preserving related tests and checking that unrelated failures persist without its patch. The PPO path reaches 120 turns. These are selected examples, not unbiased estimates of overall behavior.

Verification and repeated interaction. Shortening does not require eliminating verification. In 32 selected early–late task pairs, turns decrease from 58.97 to 50.25 and nonzero tool exits from 16.69% to 10.05%, while test/compilation calls increase from 220 to 245. Repetitions also increase (50 to 75). These proxies indicate selective change, not uniformly fewer tests or repeated actions; turn counts alone do not measure total cost.

F Selector Prompts and Output Contract

The executable Chinese prompts and their SHA256 manifests are in `prompts/`. The following English summaries explain their roles.

Initial instruction. The original is in `prompts/selector_before_original_zh.txt`. Its English translation is:

> Given the task, the complete student trajectory, tool feedback, and the terminal checker result, estimate for every assistant turn $\widehat{A} = Q(\text{actual action}) - \mathbb{E}[Q(\text{ordinary same-policy alternative})]$. Output the turn identifier, $\widehat{A}$, and a short reason, and use $|\widehat{A}|$ to select important turns.

Every turn receives a score; the importance ranking does not filter turns from the actor loss.

Shared target. Estimate each action's expected terminal utility relative to typical alternatives from the same student at that state. This is not action importance or local correctness alone. The utility distinguishes incorrect completion (-1), correct but inefficient completion (0), and efficient correct completion ($+1$). Counterfactual branches are unavailable to the selector.

Evidence and decision rules. Use tool evidence and subsequent actions, not self-assessment. A failed command may be valuable if it informs later decisions; repeated checks may be unhelpful. Evaluate damage and recovery against alternatives at their respective states. Final success does not make every edit or submission high-credit.

Prompt v5: relative action utility. Reconstruct the known state and ordinary alternatives; assess new evidence, state changes, downstream use, and recovery burden. Determine sign before ranking magnitude. Use conservative values when actions are readily interchangeable, without inventing sign reversals.

Prompt v6: diagnosis and consequential decisions. Include plans or diagnoses that determine later editing, rollback, verification, or stopping. Unsupported fixation is negative only if it worsens execution. Truncated tool output may still contain useful evidence.

Prompt v8: contrasting roles. Contrast localization with repeated confirmation, damage with recovery, and decision-changing tests with redundant verification. The contrasts guide comparison with alternatives; they do not fix category signs.

Magnitude and schema. Scores are finite and lie in $[-2, 2]$. Each legal turn appears exactly once with a score and a brief evidence-based reason. Reasons and internal thinking are not actor-loss targets. This single-entry example illustrates the schema; actual output must cover the full trajectory:

```
{"turns":[{"turn_id":0,"A_hat":0.15,"reason":"brief evidence"}]}
```