TRAINING ADVISORS FOR LLM AGENTS FROM TASK OUTCOMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents tackle multi-step tasks by interleaving reasoning and tool calls with observations from the environment. Prior work has shown that natural-language feedback can help these agents revise their decisions during task execution. We introduce CADDIE, a method for training critics to provide natural-language analysis and advice as agents work through a task. Unlike approaches that rely on step-level labels or reference critiques, CADDIE learns from whether the agent ultimately succeeds after receiving the critic's feedback. We optimize the critic through reinforcement learning while keeping the base model frozen. Trained on multi-hop question answering with a single base model, our Qwen3-4B critic improves success rates across four base models of different scales and architectures, including three not used during critic training. On the MuSiQue benchmark, the trained critic improves Qwen3-4B's success rate by more than 25 percentage points, surpassing the performance of Kimi K3 without a critic. The same critic also yields gains on out-of-domain interactive benchmarks, including τ^3 and DeepDive, with no additional training. Our results show that agents can decide when to seek help from a critic at inference time and that outcome-based critic training can produce guidance that transfers across base models and task domains.

1 INTRODUCTION

Large language model (LLM)-based agents (Yao et al., 2023; 2025) increasingly tackle multi-step tasks by producing actions and reacting to environment observations. During its execution, an agent can get stuck repeating unsuccessful actions or lose sight of the original goal (Shinn et al., 2023; Kim et al., 2025). External feedback offers a way to help models reconsider their approach and steer the agent toward a successful solution by identifying problems and suggesting more useful next actions.

Outcome reward models (ORMs) (Cobbe et al., 2021) and process reward models (PRMs) (Uesato et al., 2022; Lightman et al., 2024) address this need by assessing complete trajectories or intermediate steps, respectively. These models typically predict a score or verdict, sometimes preceded by textual reasoning, that can be used to rank candidates, identify faulty steps, or guide search. However, conventional step-level verification faces two practical limitations:

- **Costly supervision.** Many PRMs require step-level training labels. Human annotation (Lightman et al., 2024) can provide these labels, but collecting them at scale is expensive (Wang et al., 2024). Automatic labeling (Wang et al., 2024; Luo et al., 2024) can also be costly and introduce noisy supervision.
- **Ambiguous step correctness.** Correctness-based assessment assumes that intermediate actions can be meaningfully classified as correct or faulty. In interactive environments, an apparently unproductive action may be needed to explore the environment or gather information. Recent agent PRMs (Xi et al., 2025) address this ambiguity by estimating future success, but using these scores to choose a better action still requires an additional selection or search procedure.

Critique-based methods (Akyurek et al., 2023; Xie et al., 2025; Xi et al., 2026) provide a complementary interface: natural-language feedback can explain a problem in the agent's current context and suggest how to proceed. By guiding the next action directly, such feedback can reduce the need for

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.

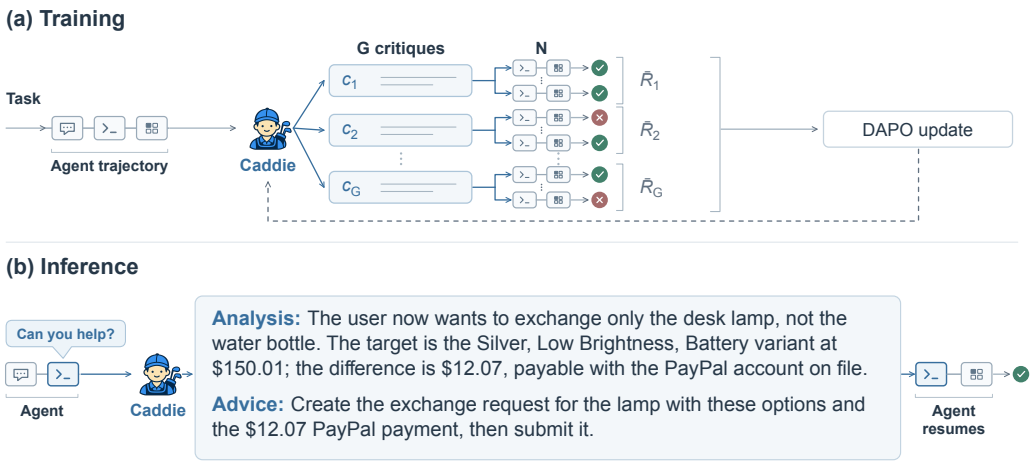


Figure 1: **Training and using CADDIE.** (a) Training: the frozen agent π_b rolls out a task and a step k is chosen randomly; at the prefix s_k the critic π_c^θ samples G critiques; each critique is then appended to the prefix and the frozen agent rolls out N continuations until the end of the episode. The terminal rewards are normalized within the group and update only the critic with DAPO. (b) Inference: the agent decides when to consult the critic through the `call_critic` tool and receives the critique as the tool output.

search, branching, or replay, which may be costly in some interactive environments. Such a method also makes potential errors and proposed next steps easier to interpret.

However, it is not straightforward how to train such critics effectively. Existing approaches train such critics through procedures that include synthesizing reference critiques (CTRL; Xie et al., 2025) or separately optimizing their ability to judge and improve solutions (Critique-RL; Xi et al., 2026). These methods demonstrate the value of learned feedback and motivate simpler ways to train it for an agent that must continue acting and observing.

We introduce CADDIE, a method for training critics to guide agents during task execution (Figure 1). During training, we sample intermediate points in an agent's trajectory and ask the critic to provide natural-language analysis and advice. The agent then continues with this feedback, and we use its final task outcome as the reward for updating the critic through reinforcement learning, while keeping the agent frozen. We directly optimize the final outcome of the policy being steered, and require neither step-level annotations nor reference critiques. At inference time, the agent can decide when to consult the trained critic.

We evaluate our approach across multiple benchmarks and base models to study both in-domain effectiveness and generalizability to other base models and tasks. We also examine how the usefulness of feedback depends on the base model and the critic inference setup.

Our main contributions are:

- CADDIE: a method for training generative advisors to guide ongoing agent interactions using downstream task outcomes, without requiring step-level annotations or reference critiques.
- We show that a Qwen3-4B critic trained with one base model improves three additional models and yields gains on out-of-domain benchmarks without further training.
- We demonstrate useful feedback under various inference strategies, and analyze how its effectiveness differs across setups and base models.

2 METHOD

We introduce CADDIE, a method for training a generative critic that can provide natural-language feedback to an LLM agent at intermediate states within a trajectory. Section 2.1 formalizes the

interaction between the agent, the environment, and the critic, including when the critic is invoked. Section 2.2 describes how we train the critic using the final outcomes of the base model's continuations while keeping the base model frozen.

2.1 CRITIQUE INTERVENTION

Agent–environment interaction. For a task $x \sim \mathcal{D}$, the frozen base policy π_b interacts with an environment $\mathcal{E}$. At step t, its state is defined as:

$$s_t = (p, x, a_0, o_0, \ldots, a_{t-1}, o_{t-1}), \tag{1}$$

where p is the system prompt, a_i an action, and o_i the corresponding observation. Each step samples an action and an environment response, then appends them to the state:

$$a_t \sim \pi_b(\cdot \mid s_t), \qquad o_t \sim \mathcal{E}(\cdot \mid s_t, a_t), \qquad s_{t+1} = s_t \oplus (a_t, o_t). \tag{2}$$

The episode ends at step T with a terminating action or an exhausted interaction budget. The complete trajectory $\tau = s_{T+1}$ receives a final task reward $R(\tau)$.

Critic policy. The critic is a language model with policy π_c^θ and trainable parameters θ. When invoked at step t, it observes state s_t and generates a natural-language critique

$$c_t \sim \pi_c^\theta(\cdot \mid s_t), \tag{3}$$

consisting of an analysis of the state and advice for the next steps (the format is specified in Appendix B). Then, the critique is appended as an additional message,

$$\widetilde{s}_t = s_t \oplus c_t, \tag{4}$$

and the agent samples its next action from the base policy conditioned on the augmented state, $a_t \sim \pi_b(\cdot \mid \widetilde{s}_t)$, following Eq. 2 for subsequent steps.

Intervention protocol. An intervention protocol $\mathcal{P}$ determines whether to invoke the critic at each intermediate step t, i.e. $\mathcal{P}(s_t) \in \{0, 1\}$. A trajectory may contain zero, one, or several critiques. We study three protocols:

1. **Windowed intervention.** An intervention step k is uniformly sampled from a predefined window $\mathcal{K} = \{k_{\min}, \ldots, k_{\max}\}$. The critic is invoked if the episode reaches that step k, $\mathcal{P}_{\text{win}}(s_t) = \mathbf{1}[t = k]$. The window can consist of a single step ($k_{\min} = k_{\max}$).
2. **Probabilistic intervention.** At each step, the critic is invoked independently with a fixed probability α, so $\mathcal{P}_{\text{prob}}(s_t) \sim \text{Bernoulli}(\alpha)$.
3. **Self-call intervention.** The base model's action space includes an additional tool, `call_critic`. The base policy decides when to consult the critic by selecting this tool, $\mathcal{P}_{\text{self}}(s_t) = \mathbf{1}[a_t = \texttt{call_critic}]$, and receives the critique as its output.

Unless stated otherwise, we use $\mathcal{P}_{\text{self}}$ for evaluation. The prompt and tool description are provided in Appendix G, and the protocols are compared in Appendix F. Windowed and probabilistic intervention require no changes to the base model's prompt or tool definitions. However, they do not account for the current trajectory and may intervene too early, too late, or when no help is needed. Self-call intervention instead uses an adapted system prompt $\widetilde{p}$ describing the critic tool, while keeping the base model's parameters fixed. It lets the agent request help without an externally tuned schedule. Windowed or probabilistic intervention can perform better in some settings, but their performance depends on the intervention window or probability.

2.2 OUTCOME-BASED CRITIC TRAINING

We train the critic π_c^θ to generate feedback that improves the final reward of a frozen base model π_b.

Outcome-based rewards. We sample state prefixes s_k using $\mathcal{P}_{\text{win}}$ with $k \sim \text{Uniform}(\mathcal{K})$, where $k_{\min}$ and $k_{\max}$ are fixed before training for each task environment. For each prefix, the critic $\pi_c^{\theta_{\text{old}}}$ generates a group of G critiques $c^{(1)}, \ldots, c^{(G)}$. To evaluate each critique $c^{(i)}$, we append it to the prefix and run N independent continuations, producing complete trajectories $\tau^{(i,n)} \sim P_b(\cdot \mid s_k \oplus c^{(i)})$. Here, P_b denotes the trajectory distribution induced by the frozen base model and environment $\mathcal{E}$. We assign the critique the average final reward across these continuations,

$$R_i = \frac{1}{N} \sum_{n=1}^{N} R\big(\tau^{(i,n)}\big). \tag{5}$$

Thus, feedback is evaluated through its effect on downstream task performance.

Critic optimization. We use DAPO (Yu et al., 2025), which builds on GRPO (Shao et al., 2024) with asymmetric clipping, token-level loss aggregation, and dynamic sampling. Following GRPO, we compute the group-normalized advantage $\hat{A}_i = (R_i - \mu_R)/\sigma_R$, where μ_R and σ_R are the mean and standard deviation of the rewards $R_1, \ldots, R_G$, and use the same advantage for every token in critique $c^{(i)}$. DAPO's dynamic sampling discards groups with zero reward variance and samples additional groups to fill the training batch. For a retained group, the clipped surrogate objective is

$$\mathcal{J}_{\text{DAPO}}(\theta) = \frac{1}{\sum_{i=1}^{G} |c^{(i)}|} \sum_{i=1}^{G} \sum_{t=1}^{|c^{(i)}|} \min\Big[\rho_{i,t}(\theta)\hat{A}_i,\ \text{clip}\big(\rho_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\big)\hat{A}_i\Big], \tag{6}$$

where $\rho_{i,t}(\theta) = \pi_c^{\theta}(c_t^{(i)} \mid s_k, c_{<t}^{(i)}) / \pi_c^{\theta_{\text{old}}}(c_t^{(i)} \mid s_k, c_{<t}^{(i)})$ is the token probability ratio. The objective averages over all generated critique tokens. Only the critic's parameters are updated; task instructions, observations, and base-model continuation tokens are excluded from the loss. Implementation details and hyperparameters are provided in Appendix D.

Other reward definitions are possible, such as improvement over a continuation without a critique; we discuss these alternatives in Appendix A.

Such a formulation also constrains the choice of tasks and environments. For tasks solvable in a single step with a small token budget and no tool calls, an optimal critic may learn to solve the task itself and instruct the base model to copy its solution. We discuss this limitation further in Appendix K.

3 Experiments

We evaluate whether outcome-based critic training improves task performance and whether the learned feedback transfers to base models and tasks unseen during critic training.

3.1 Experimental setup

Tasks and environments. We study in-domain performance on **MuSiQue** (Trivedi et al., 2022) and **ALFWorld** (Shridhar et al., 2021), training a separate critic for each setting. MuSiQue requires combining evidence across two to four reasoning hops. We use two retrieval settings: *attached*, in which the agent retrieves evidence from the 20 supplied passages, and *wiki*, in which it searches Wikipedia. ALFWorld requires agents to navigate text-based household environments and interact with objects to complete tasks. To test transfer beyond the training tasks, we evaluate the MuSiQue-trained critics on **DeepDive** (Lu et al., 2025), which requires web search to answer questions synthesized from knowledge graphs, and on τ^3 Retail and Airline (Yao et al., 2025; Sierra Research, 2026), which require customer dialogue and tool use under domain-specific policies.

Critic training. All critics are initialized from Qwen3-4B-Instruct-2507 (Qwen Team, 2025b) and trained with DAPO as described in Section 2.2. We train separate critics for MuSiQue-attached, MuSiQue-wiki, and ALFWorld, keeping the Qwen3-4B base model frozen. For MuSiQue and ALFWorld, we use the authors' training split. Unless otherwise stated, we use $G = 16$ critiques per state prefix and estimate each critique's reward using one base model continuation ($N = 1$). Training

runs use a single node with eight NVIDIA H100 GPUs, each with 80 GB of memory. Training hyperparameters, intervention windows, and checkpoint selection are described in Appendix D.

Base models and transfer. We evaluate all three critics with Qwen3-4B, the base model used during training. On MuSiQue, we also evaluate the corresponding critics with Qwen3-30B-A3B-Instruct-2507 (Qwen Team, 2025a), Qwen3.8-27B (Qwen Team, 2026), and Kimi K3 (Moonshot AI, 2026) to test how the critic generalizes to different base models. For transfer to DeepDive and τ^3, we use Qwen3-4B and Qwen3.8-27B. The latter tests joint transfer across tasks and base models. For each trained variant, we use a single fixed critic checkpoint throughout evaluation. We abbreviate checkpoint names in tables.

Baselines. We compare CADDIE with no critic, a prompted Qwen3-4B critic, and a prompted Kimi K3 critic. The prompted Qwen3-4B critic uses the same checkpoint as our critic's initialization, providing a fair comparison without outcome-based critic training. For transfer to unseen task families, we also include Agent-RRM (Fan et al., 2026), a generative reward model trained using teacher-generated reasoning, critiques, and trajectory scores. All main critic-assisted evaluations use the self-call protocol $\mathcal{P}_{\text{self}}$, allowing the base model to request at most three critiques per trajectory.

Evaluation metric. For each condition, we generate eight independent rollouts per task using distinct seeds and fixed model checkpoints. We first average the binary success outcomes within each task, then report the mean of these per-task success rates across the evaluation set. The standard error of the mean (SEM) is the sample standard deviation of the per-task rates divided by the square root of the number of tasks. For the smaller DeepDive and τ^3 evaluation sets (Table 3), we instead compute the SEM across the eight seed-level aggregate accuracies, where each seed's success rate over all tasks is one sample.

3.2 IN-DOMAIN PERFORMANCE

We evaluate each critic in its training task setting with Qwen3-4B as the base model. For MuSiQue, we use a random subset of 200 validation questions. We impose no step limit and use a maximum context length of 8,192 tokens. For ALFWorld, we use the official unseen validation split of 134 games, which are held out from the training split.

Table 1: In-domain success rates (%) with Qwen3-4B as the base model. Each CADDIE critic is trained in the corresponding task setting. Entries report mean ± SEM across tasks, with eight rollouts per task. Bold marks the highest mean; italics mark other means whose ± SEM intervals overlap the leader's (not a significance test).

	MuSiQue		
Critic	Attached	Wiki	ALFWorld
No critic	21.56 ± 2.07	10.38 ± 1.59	23.79 ± 3.01
Prompted Qwen3-4B	23.25 ± 2.37	16.06 ± 2.19	20.24 ± 2.89
Prompted Kimi K3	29.13 ± 2.46	*19.19* ± 2.23	30.22 ± 2.80
CADDIE (ours)	$\mathbf{46.88} \pm 2.68$	$\mathbf{21.13} \pm 2.38$	$\mathbf{49.72} \pm 3.42$

Outcome-based training improves feedback quality. CADDIE achieves the highest mean success rate in both MuSiQue settings and on ALFWorld (Table 1). Relative to the no-critic baseline, it improves success on MuSiQue by 25.31 percentage points with attached passages, 10.75 points with Wikipedia retrieval and by 25.93 points on ALFWorld. The trained 4B critic also outperforms both prompted critics in all three settings, indicating that outcome-based training produces more useful feedback than prompting alone in these evaluations.

Critic feedback after generator training. To study whether a critic can further improve a generator that has already been trained on the downstream task, we first trained the Qwen3-4B generator with DAPO-based optimization until it plateaus, then froze it and trained a critic through its continuations. On MuSiQue-attached, this additionally raises success from 49.69% to 63.56% ($+13.88$ percentage

points), showing that critic feedback can provide further gains even after the generator itself has been trained on the task. This suggests that the two forms of training are complementary. Appendix E gives the training setup and results for both retrieval settings.

3.3 TRANSFER TO UNSEEN BASE MODELS

We pair the MuSiQue-trained critics with three base models unseen during critic training, retaining the retrieval settings and evaluation protocol above. This tests whether feedback learned through continuations from Qwen3-4B remains useful to other base models.

Table 2: Transfer across base models on MuSiQue. Success rates (%) are reported as mean ± SEM across tasks, with eight rollouts per task. CADDIE critics are trained with Qwen3-4B and evaluated with each base model without further adaptation. Bold marks the highest mean; italics mark other means whose ± SEM intervals overlap the leader's (not a significance test).

		Prompted critic		
Base model	No critic	Qwen3-4B	Kimi K3	CADDIE (ours)
Attached passages				
Qwen3-30B-A3B	28.44 ± 2.45	34.75 ± 2.82	*36.00* ± 2.86	$\mathbf{41.63} \pm 2.83$
Qwen3.8-27B	46.63 ± 2.93	45.31 ± 2.84	41.13 ± 2.77	$\mathbf{55.94} \pm 2.93$
Kimi K3	38.56 ± 2.87	34.44 ± 2.63	30.19 ± 2.36	$\mathbf{51.00} \pm 2.76$
Wikipedia retrieval				
Qwen3-30B-A3B	*19.94* ± 2.38	*20.50* ± 2.43	*20.06* ± 2.42	$\mathbf{20.75} \pm 2.41$
Qwen3.8-27B	*26.69* ± 2.69	*23.69* ± 2.53	21.44 ± 2.36	$\mathbf{27.50} \pm 2.79$
Kimi K3	20.06 ± 2.29	19.75 ± 2.28	18.75 ± 2.18	$\mathbf{26.25} \pm 2.55$

Feedback transfers across base models. CADDIE improves over the no-critic baseline and achieves a higher mean than both prompted critics in all six conditions (Table 2). With attached passages, gains over the no-critic baseline range from 9.31 to 13.19 percentage points across the three models. Transfer is less consistent with Wikipedia retrieval, ranging from 0.81 to 6.19 points; Qwen3.8-27B increases from 26.69% to 27.50%. Thus, a critic trained with a 4B base model can benefit larger base models without adaptation, although the magnitude of the benefit depends on the base model and retrieval setting.

3.4 TRANSFER TO UNSEEN TASK FAMILIES

We evaluate both MuSiQue-trained critics on 40 τ^3 Retail tasks, 20 τ^3 Airline tasks, and a subset of 64 DeepDive tasks without further critic training. In each critic-assisted condition, the frozen base model uses the self-call protocol to request feedback. For τ^3, we use the updated benchmark environments with the predefined `test` splits inherited from τ^2-bench. Evaluation with Qwen3-4B tests task transfer; evaluation with Qwen3.8-27B tests joint transfer across tasks and base models. For all out-of-domain evaluations, both the critic and the base model use a maximum context length of 32,768 tokens.

Feedback transfers beyond the training tasks. The attached-trained critic exceeds the no-critic mean in five of six base model–task combinations, and the wiki-trained critic in all six (Table 3). With Qwen3.8-27B, the wiki-trained critic improves the mean on all three tasks, including τ^3 Retail (50.31% to 58.12%) and Airline (74.38% to 78.12%). These results extend transfer beyond retrieval to customer-service interactions requiring dialogue and state-changing tool use.

Across the two trained variants, CADDIE achieves the highest mean in five of six conditions; the prompted Kimi K3 critic leads on Qwen3.8-27B Retail. Transfer nevertheless depends on the training setting: on Qwen3.8-27B DeepDive, the attached-trained critic reduces success from 48.83% to 44.53%, whereas the wiki-trained critic yields a small increase to 49.80%.

Table 3: Transfer of MuSiQue-trained critics to unseen tasks. Entries report success rates (%): the mean over all tasks and seeds ± the SEM across the eight seed-level aggregate accuracies. Shading marks our critics. Bold marks the highest mean; italics mark other means whose ± SEM intervals overlap the leader's (not a significance test).

		τ^3	
Critic	DeepDive	Retail	Airline
Base model: Qwen3-4B			
No critic	26.95 ± 2.38	44.69 ± 1.73	31.25 ± 1.57
Prompted Qwen3-4B critic	28.91 ± 1.39	42.50 ± 0.98	31.88 ± 2.49
Prompted Kimi K3 critic	27.34 ± 0.72	40.31 ± 1.53	27.50 ± 2.83
Agent-RRM	29.30 ± 0.75	40.63 ± 2.15	*33.75* ± 2.63
CADDIE (attached)	29.10 ± 2.17	45.00 ± 1.34	$\mathbf{37.50} \pm 2.67$
CADDIE (wiki)	$\mathbf{34.38} \pm 1.72$	$\mathbf{48.75} \pm 1.25$	*36.88* ± 3.26
Base model: Qwen3.8-27B			
No critic	*48.83* ± 1.61	50.31 ± 1.60	*74.38* ± 1.75
Prompted Qwen3-4B critic	*46.09* ± 3.60	53.75 ± 2.83	*74.38* ± 2.58
Prompted Kimi K3 critic	41.80 ± 2.81	$\mathbf{59.69} \pm 2.73$	62.50 ± 2.11
Agent-RRM	*48.44* ± 3.54	44.06 ± 1.70	49.38 ± 5.13
CADDIE (attached)	*44.53* ± 3.21	*59.06* ± 1.70	*75.63* ± 2.74
CADDIE (wiki)	$\mathbf{49.80} \pm 2.58$	*58.12* ± 2.44	$\mathbf{78.12} \pm 2.10$

4 ANALYSIS

We study how the MuSiQue-wiki CADDIE critic helps frozen generators solve problems outside its training domain. The generator decides when to ask for advice, so the analysis below describes the behavior of self-call rather than a randomized intervention. Additional breakdowns and qualitative examples appear in Appendix H.

The generator usually asks once, and feedback helps when requested. Among rollouts that call CADDIE, a single request is by far the most common behavior. A second request does not coincide with lower accuracy on DeepDive. The complete call-count distribution and the corresponding accuracy curves are shown in Appendix H. Figure 2 compares feedback-bearing rollouts with no-critic rollouts on the same tasks. The feedback-bearing group has a higher solve rate in all six settings, but this comparison is conditional on the generator requesting feedback and does not isolate its causal effect.

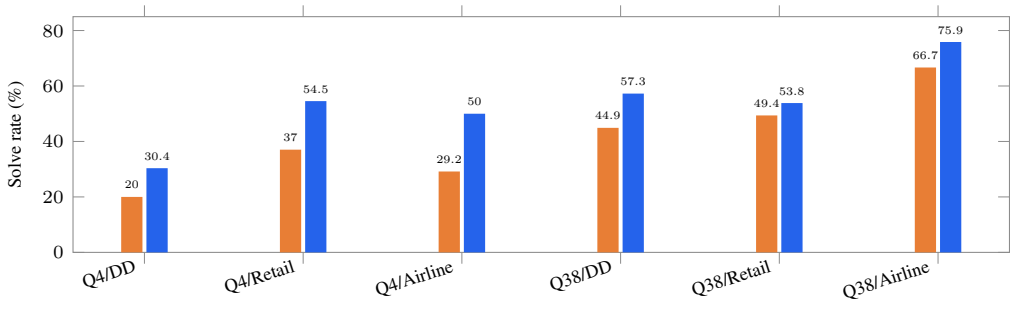


Figure 2: Solve rates on tasks where CADDIE delivered feedback. Blue: feedback-bearing rollouts; orange: no-critic rollouts on the same tasks. This generator-selected subset gives a descriptive comparison, not a causal estimate. Q4/Q38 denote Qwen3-4B/Qwen3.8-27B; DD denotes DeepDive.

The advice adapts to the task. On DeepDive, CADDIE mainly suggests either another evidence search or that the generator should stop and answer. On τ^3, particularly with Qwen3.8-27B, it more

often recommends verifying state, asking the user, or executing an action. Thus the critic does not repeat one generic instruction: its advice changes with the environment. Figure 3 illustrates this behavior after a τ^3 user changes intent. Additional distributions and examples are provided in Appendix H.

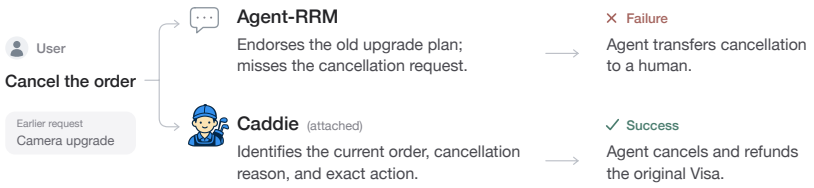


Figure 3: Matched τ^3 Retail trajectories after the same change in user intent. Each path summarizes the critic advice and the frozen-generator continuation.

The most useful advice depends on generator scale. Figure 4 shows a simple contrast. The smaller Qwen3-4B generator benefits most from targeted search advice, matching its tendency to stop before collecting enough evidence. The larger Qwen3.8-27B generator benefits most from being told to answer with evidence it already has, matching its tendency to keep searching after the answer is supported. Additional search can therefore help the smaller generator but distract the larger one.

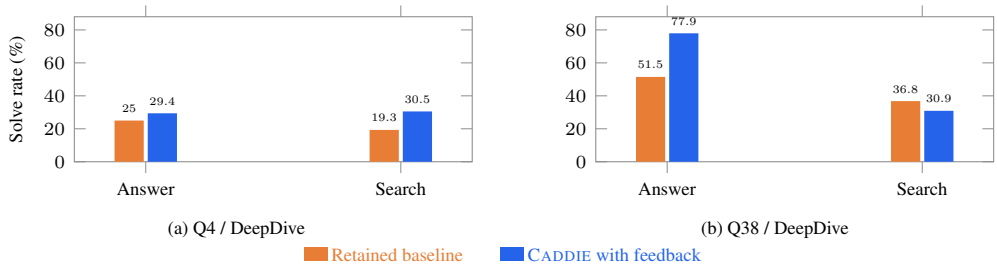


Figure 4: DeepDive solve rate by advice type. Orange (retained baseline): no-critic rollouts on the tasks where that advice type was given; blue: rollouts that received it.

Together, these results suggest a simple OOD mechanism: the generator calls CADDIE selectively, and the critic adapts its correction to both the target domain and the generator's characteristic weakness.

5 Discussion and Limitations

Task structure. Our method is intended for multi-step, tool-using tasks whose solutions require further environment interactions. On self-contained tasks, such as step-by-step mathematical reasoning, the critic can instead learn to generate complete solutions for the base model to copy, which can be optimal under the outcome-based objective. In this case, the critic learns to act as a task-solving policy, limiting its ability to transfer across base models and tasks. We discuss this further in Appendix K.

Practical constraints. Training requires GN base-model continuations per sampled state and the ability to restore the same environment state across branches. This can be expensive for long trajectories, large base models, or costly tool calls. We train only Qwen3-4B critics, leaving the effect of critic scale on performance and transfer unverified.

Scalar verification. Our formulation treats critique as an intervention rather than a correctness judgment about an individual step. Scalar verifiers can rank or select candidate continuations, whereas a generative critic supplies text that conditions the base model's next actions. These approaches can be combined. Our experiments do not establish that natural-language feedback is generally superior to scalar verification; such a comparison would require matched inference procedures and compute budgets.

6 Related Work

Learning critics from task outcomes. RL4F (Akyurek et al., 2023) and CTRL (Xie et al., 2025) train critics to improve a frozen model's revised outputs. Both methods initialize the critic with supervised critique examples: RL4F uses human-written or programmatically generated feedback, while CTRL synthesizes critiques using code-execution feedback. Critique-RL (Xi et al., 2026) first trains correctness judgments, then optimizes feedback for successful revisions while preserving judgment accuracy. We train the critic directly from the final outcomes of agent continuations, providing advice during an ongoing trajectory without supervised critique initialization or a separate correctness-judgment stage.

Learned feedback for agents. Steer, Don't Solve (Gandhi et al., 2026) trains critics using teacher feedback and judge-ranked preferences. Agent-RRM (Fan et al., 2026) trains an evaluator whose critiques guide revisions of completed attempts. Closest to our work, Advisor Models (Asawa et al., 2026) train a small advisor with GRPO to write per-instance advice for a frozen student, rewarded only by the student's task reward, and show that an advisor trained with a low-cost student transfers to frontier models. We share this training signal but differ in where credit is assigned, when advice arrives, and what transfers. They score advice by the student's response to the whole query, delivered before the response, after a completed attempt, or every five steps in their coding-agent experiments; we branch from a sampled intermediate state and compare G critiques through continuations from that same state, so the critic learns to correct an episode in progress rather than to prompt a fresh attempt. Their advice follows a fixed schedule; our agent requests it through `call_critic` when uncertain, avoiding the schedule sensitivity shown in Appendix F. They report transfer across students and no degradation on other benchmarks; we show gains for unseen base models on unseen task families without any adaptation.

Revision and reflection. Welleck et al. (2023) train a separate corrector that directly rewrites a generator's outputs. GLoRe (Havrilla et al., 2024) similarly trains models to revise solutions, using reward models to locate errors and guide global or local revisions. Retroformer (Yao et al., 2024) learns reflections on failed trajectories to guide subsequent attempts. We instead train the critic to advise the agent on its next steps during the same attempt.

Training agents with feedback. ECHO (Li et al., 2026) trains the critic alongside the agent so that feedback adapts as the agent learns. PivoARL (Guo et al., 2026) learns both reflection and action generation from retries after failures. ICRL (Lin et al., 2026) trains a shared model as both solver and critic, while Kim et al. (2026) alternate critic and generator training for open-ended generation. Our training updates only the critic, leaving the acting model's parameters unchanged.

7 Conclusion and Future Work

We introduced CADDIE, a method for training natural-language critics to provide analysis and advice during an agent's interaction. We adapt the GRPO-based optimization generally used for policy training to learn critiques from the final outcomes of a frozen base model's continuations. Using task outcomes as a common training signal avoids the need for task-specific step labels or reference critiques.

On MuSiQue, our trained Qwen3-4B critic improves the Qwen3-4B base model's success rate by 25.31 percentage points in the attached setting. The critic also transfers to base models not used during training, including Qwen3-30B-A3B, Qwen3.8-27B, and Kimi K3, and yields gains on out-of-domain tasks in τ^3 and DeepDive without further training. These results show that the same training procedure can produce feedback that is useful across base models and task settings.

We also identify a limitation concerning task structure. In self-contained tasks that can be solved in one pass, the critic can learn to generate complete solutions for the base model to copy, which can be optimal under the outcome-based objective.

Future work could explore fine-tuning the base policy to use a trained critic more effectively and explore joint training of the base policy and critic.

AI USE STATEMENT

In this work, we used generative AI tools to help formulate mathematical claims and to help interpret results. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis. Additionally, we used generative AI tools to create or modify scientific figures or images, analyze existing literature, for sourcing/searching for information, to edit a research paper to improve readability, to identify relevant literature and format references. We have reviewed all AI-assisted work. We read all the related works and verified the text after AI improvements. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 2 defines the agent–critic interaction, reward computation, and optimization objective. Section 3.1 describes the models, baselines, training hardware, and evaluation protocol, including the aggregation of success rates and standard errors. Appendix C documents dataset splits, task filtering, retrieval settings, and answer evaluation, while Appendices D and E provide optimization and sampling hyperparameters, numerical precision, and additional training configurations. The feedback format, message placement, and interaction budgets are specified in Appendix B, and the prompts are provided in Appendix G. Alternative reward definitions and inference protocols are detailed in Appendices A and F. Section 4 and Appendix H present the self-call and advice analyses. Appendices I and J provide task-family breakdowns and trajectory examples, and Appendix K discusses task structure and environment restoration. All experiments were conducted using a custom internal JAX framework supporting asynchronous reinforcement learning.

REFERENCES

Afra Feyza Akyurek, Ekin Akyurek, Ashwin Kalyan, Peter Clark, Derry Tanti Wijaya, and Niket Tandon. RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7716–7733, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.427. URL https://aclanthology.org/2023.acl-long.427/.

Parth Asawa, Alan Zhu, Abigail O'Neill, Matei Zaharia, Alexandros G. Dimakis, and Joseph E. Gonzalez. How to train your advisor: Steering black-box LLMs with advisor models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.02453.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Kaixuan Fan, Kaituo Feng, Manyuan Zhang, Tianshuo Peng, Zhixun Li, Yilei Jiang, Shawn Chen, Peng Pei, Xunliang Cai, and Xiangyu Yue. Exploring reasoning reward model for agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 1975–1991, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.95. URL https://aclanthology.org/2026.findings-acl.95/.

Shubham Gandhi, Yiqing Xie, Atharva Naik, Ruichen Zhu, and Carolyn Rosé. Steer, don't solve: Training small critic models for large coding agents, 2026. URL https://arxiv.org/abs/2606.21811v2.

Weiyang Guo, Zesheng Shi, Longhui Zhang, Zeen Zhu, Min Zhang, and Jing Li. Agent reinforcement learning via pivotal-aware self-feedback retry, 2026. URL https://arxiv.org/abs/2607.03702.

Alexander Havrilla, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. GLoRe: When, where, and how to improve LLM reasoning via global and local refinements. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17719–17733. PMLR, 2024. URL https://proceedings.mlr.press/v235/havrilla24a.html.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. ReflAct: World-grounded decision making in LLM agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33433–33465, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1697. URL https://aclanthology.org/2025.emnlp-main.1697/.

Jinyoung Kim, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Co-evolving actor-conditioned critics for non-verifiable generation, 2026. URL https://arxiv.org/abs/2608.30397.

Zhicong Li, Lingjie Jiang, Yulan Hu, Xingchen Zeng, Yixia Li, Xiangwen Zhang, Guanhua Chen, Zheng Pan, Xin Li, and Yong Liu. No more stale feedback: Co-evolving critics for open-world agent learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12643–12660. Association for Computational Linguistics, July 2026. doi: 10.18653/v1/2026.acl-long.576. URL https://aclanthology.org/2026.acl-long.576/.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Jianbo Lin, Xiaomin Yu, Yi Xin, Yifu Guo, Zhuosong Jiang, Zhongqi Yue, Weishi Wang, Heqing Zou, Chengwei Qin, and Hui Xiong. ICRL: Learning to internalize self-critique with reinforcement learning, 2026. URL https://arxiv.org/abs/2605.15224.

Rui Lu, Zhenyu Hou, Zihan Wang, Hanchen Zhang, Xiao Liu, Yujiang Li, Shi Feng, Jie Tang, and Yuxiao Dong. DeepDive: Advancing deep search agents with knowledge graphs and multi-turn RL, 2025. URL https://arxiv.org/abs/2509.10446.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024. URL https://arxiv.org/abs/2406.06592.

Moonshot AI. Kimi K3 model card, 2026. URL https://huggingface.co/moonshotai/Kimi-K3.

Michael Oberst and David Sontag. Counterfactual off-policy evaluation with Gumbel-Max structural causal models. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4881–4890. PMLR, 2019. URL https://proceedings.mlr.press/v97/oberst19a.html.

Qwen Team. Qwen3-30B-A3B-Instruct-2507 model card. https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507, 2025a. URL https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507. Accessed 2026-09-04.

Qwen Team. Qwen3-4B-Instruct-2507 model card. https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507, 2025b. URL https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507. Accessed 2026-09-04.

Qwen Team. Qwen3.8-27B model card. https://huggingface.co/Qwen/Qwen3.8-27B, 2026. URL https://huggingface.co/Qwen/Qwen3.8-27B. Accessed 2026-09-04.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Sierra Research. τ^3-bench: Benchmark for tool-agent-user interaction, 2026. URL https://github.com/sierra-research/tau2-bench. Repository for the τ^3-bench release; accessed 2026-09-23.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://doi.org/10.1162/tacl_a_00475.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2211.00053.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress, 2025. URL https://arxiv.org/abs/2511.08325.

Zhiheng Xi, Jixuan Huang, Xin Guo, Boyang Hong, Dingwen Yang, Xiaoran Fan, Shuo Li, Zehui Chen, Junjie Ye, Siyu Yuan, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. Critique-rl: Training language models for critiquing through two-stage reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d0e08f7a2d1b2d5c2da159395e8d61aa-Abstract-Conference.html.

Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68559–68577. PMLR, 2025. URL https://proceedings.mlr.press/v267/xie25a.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Ramapura Narasimha Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/29f421fbdcc82aeb349d784d3aaccdb3-Abstract-Conference.html.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

A ALTERNATIVE OBJECTIVES

Alongside the terminal-outcome reward used in the main method, we explore alternative reward definitions and continuation counts. Figure 5 compares MuSiQue validation curves on a subset of 20 tasks with $N = 1$ and $N = 5$, alongside baseline subtraction and a Gumbel counterfactual reward.

Terminal outcome. For the outcome-trained critics, $R_i = \frac{1}{N}\sum_{n=1}^{N} R(\tau^{(i,n)})$, where $R(\tau^{(i,n)}) \in \{0, 1\}$ indicates final task success. The main experiments use $N = 1$; we also evaluate $N = 5$. Figure 5 shows broadly similar validation curves, with smaller fluctuations for $N = 5$ after roughly 1,000 training steps. Final performance on the full validation set produces similar results for both variants. Since $N = 1$ requires fewer base-model continuations and therefore less compute, we use it in the main experiments.

Gumbel-coupled counterfactual reward. We explore a paired reward comparison inspired by Gumbel-Max structural causal models (Oberst & Sontag, 2019). For each critique $c^{(i)}$ at state s_k, we run two continuations with the frozen base model, one with the critique and one without it:

$$\tau_i^{\text{crit}} = F_b(s_k \oplus c^{(i)}; \xi_i), \qquad \tau_i^0 = F_b(s_k; \xi_i).$$

Here, F_b denotes a rollout of the base model and environment, and ξ_i supplies shared sampling noise. Both continuations start from the same environment state. Under Gumbel-max sampling, each generated token is selected as

$$z_j = \arg\max_v \left(\log q_j(v) + g_{j,v}\right), \qquad g_{j,v} \overset{\text{iid}}{\sim} \text{Gumbel}(0, 1),$$

where $q_j(v)$ is the base model's probability of token v given the current context in that continuation. The two continuations reuse the same noise $g_{j,v}$ at corresponding generation positions j, while their contexts may differ. The critic receives the difference in final task rewards,

$$R_i^{\text{CF}} = R(\tau_i^{\text{crit}}) - R(\tau_i^0).$$

The coupling is positional, not semantic. The contexts differ in length as soon as the critique is inserted, and matching generation positions need not correspond to the same action or reasoning step once the continuations diverge. Shared noise therefore does not guarantee matching tokens, tool calls, or environment observations. It reduces the variance of the reward difference relative to independent sampling only when the paired rewards have positive covariance. We do not measure how much correlation survives trajectory divergence, so we cannot quantify the resulting variance reduction.

The plotted counterfactual run rises earlier, but the small validation subset and absence of confidence bands do not establish faster convergence. Since this reward also requires no-critique continuations, we retain the simpler terminal-outcome reward for the main experiments.

No-critique baseline subtraction. For each state s_k, we sample M independent continuations from the frozen base model without a critique and estimate its expected final reward as

$$\widehat{b}_0(s_k) = \frac{1}{M}\sum_{m=1}^{M} R(\tau^{0,m}), \qquad \tau^{0,m} \sim P_b(\cdot \mid s_k).$$

Instead of centering critique rewards by their group mean, we use the advantage

$$\widehat{A}_i^0 = R_i - \widehat{b}_0(s_k).$$

Critiques whose estimated reward exceeds the no-critique baseline receive positive advantage; those below it receive negative advantage.

Because the baseline is sampled independently of the critique and treated as constant during optimization, its contribution to the expected, unclipped score-function gradient is zero:

$$\mathbb{E}_{c \sim \pi_c^\theta(\cdot \mid s_k)}\left[\widehat{b}_0(s_k) \nabla_\theta \log \pi_c^\theta(c \mid s_k)\right] = 0.$$

This identity does not imply that changing the baseline leaves the clipped training objective unchanged. The baseline may reduce gradient variance by accounting for differences in state difficulty, but its estimation introduces additional noise. For independent continuations, this noise decreases with M:

$$\mathrm{Var}\left[\widehat{b}_0(s_k) \mid s_k\right] = \frac{\mathrm{Var}_{\tau\sim P_b(\cdot|s_k)}[R(\tau)]}{M}.$$

In the plotted run, this variant improves early and later falls below the terminal-outcome curves; without uncertainty estimates, however, this does not establish a reliable ranking. One possible explanation concerns the signs of the advantages. Early in training, the no-critique baseline assigns negative advantages to feedback that worsens task outcomes. Later, if most critiques outperform this baseline, even critiques that are weaker than others in the group can receive positive advantages. This may affect optimization under clipping. A possible extension is to gradually shift from the no-critique baseline to the group-mean baseline during training; we leave this for future work.

Self-call during training. The main experiments use windowed intervention $\mathcal{P}_{\text{win}}$ during training. We also explore training with the self-call protocol $\mathcal{P}_{\text{self}}$. This variant achieves similar in-domain performance but transfers less consistently to other base models and tasks. One possible explanation is that different base models request feedback at different states. Training only on states where one base model chooses to call the critic may therefore limit its exposure to states encountered with other base models.

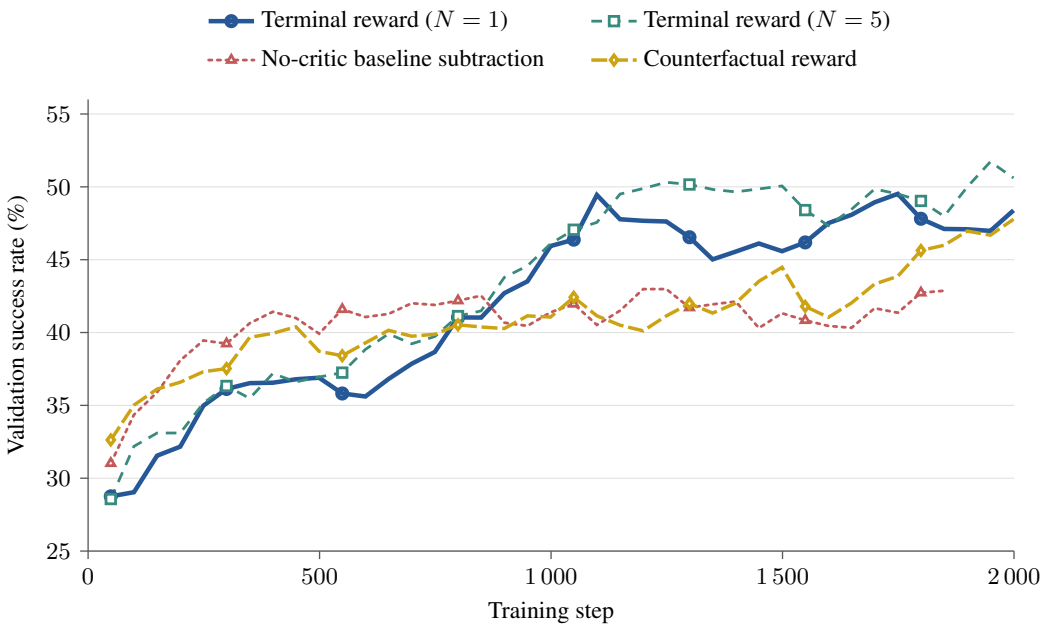


Figure 5: MuSiQue validation success on a 20-task subset for four reward variants, shown up to 2,000 training steps. All curves use an exponential moving average with weight 0.5 on the current evaluation. Confidence bands are unavailable; these smoothed traces illustrate training dynamics but do not establish differences in performance, convergence speed, or reward variance.

B FORMAT

Feedback format. The critic is instructed to analyze the trajectory and advise the base model on its next steps without calling tools, completing the task, or providing a complete action sequence. The two required sections are:

```
Analysis: <brief diagnosis>
Advices for next steps: <concise guidance>
```

Both sections must be nonempty. During training, malformed feedback receives a reward of -1 without running a base-model continuation.

Tool semantics and message placement. In self-call evaluation, the base model emits brief reasoning followed by the empty tag `<critic/>`, which implements the `call_critic` action of Section 2.1. Feedback is delivered in a `user`-role observation message with the prefix `Critic feedback:`; the full preceding base-model conversation is retained. A critic call does not advance the environment or consume an environment action. The main evaluations allow at most three critic calls per trajectory; after the third, the observation explicitly permits only an environment action.

Budgets and termination. ALFWorld evaluation allows 50 environment actions, an 8,192-token context, 128 output tokens per base-model decision, and 512 tokens per critique. Training uses a 12-action budget, shared by the prefix and continuation (Appendix D). An episode ends on success, exhaustion of the action budget, malformed output, an invalid tool call, or context overflow.

C Tasks, Splits, and Evaluators

MuSiQue. In the attached setting, the base model searches the 20 passages supplied with each MuSiQue question (Trivedi et al., 2022). In the wiki setting, it queries a frozen Wikipedia retrieval service. The main comparisons use the same 200 validation questions and eight trials per question (1,600 trajectories per condition). This subset contains 104 two-hop, 63 three-hop, and 33 four-hop questions.

The attached-setting results with trained critics as well as all the no-critic or prompted critic baselines in the main tables require at least three searches. The evaluator checks for a nonempty normalized exact match against any reference answer. Normalization consists of case-folding, replacing punctuation with spaces, removing English articles, and collapsing whitespace.

ALFWorld. We use the ALFWorld text environment (Shridhar et al., 2021) and the standard `json_2.1.1` data. The official splits contain 3,553 training games, 140 seen-validation games, and 134 unseen-validation games. We restrict critic training to place, heat, and examine-under-light tasks, yielding 1,557 eligible games (790 place, 459 heat, and 308 examine-under-light). Evaluation covers all six task families on all 134 unseen-validation games, with eight trials per game.

DeepDive and τ^3. The transfer evaluations use 64 DeepDive questions (Lu et al., 2025), 40 τ^3 Retail tasks, and 20 τ^3 Airline tasks (Sierra Research, 2026), with eight trials per task and a 32,768-token context. The τ^3 evaluations use the predefined `test` splits inherited from τ^2-bench. For DeepDive, the base model searches through Tavily, and a Qwen3-32B judge evaluates final answers in non-thinking mode.

D Models and Training Details

Model roles. In the main experiments, the critic is initialized from Qwen3-4B-Instruct-2507, and all its parameters are trainable. The Qwen3-4B-Instruct-2507 base model remains frozen.

Intervention windows. For both MuSiQue settings, we sample one intervention step uniformly from $\mathcal{K} = \{1, \ldots, 10\}$ during training. No critique is requested if the episode terminates before the sampled step.

For ALFWorld training, the intervention step is sampled uniformly from $\mathcal{K} = \{2, \ldots, 10\}$, using a task-dependent seed based on 42 and the task ID. No critique is requested if the episode has already terminated. The base-model output cap is 128 tokens per action, and the prefix and continuation share a 12-action budget. The critic is trained with one scheduled intervention and evaluated using self-call with up to three interventions and a 50-action budget.

Training duration and checkpoint selection. MuSiQue critic training stops at step 2,000, and we use this final checkpoint throughout evaluation rather than selecting the best checkpoint separately

Table 4: Optimization settings for MuSiQue-attached, MuSiQue-wiki, and ALFWorld.

Parameter	Value
Objective	Token-level DAPO; lower/upper clip $0.20/0.28$
Optimizer	AdamW; learning rate 10^{-6}; $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$
Schedule	8-update linear warmup from zero, then constant learning rate
Regularization	Weight decay 0.1, excluding normalization parameters; global gradient-norm clip 0.2
Auxiliary losses	No KL or entropy penalty
Batch size	128
Advantage	Group centering and normalization by population standard deviation; $\epsilon = 10^{-6}$
Sampling	Temperature 1, top-$p = 1$; 8,192-token context; 512-token critic output cap
Precision	FP32 parameters, BF16 transformer computation; FP32 unembedding
Format reward	-1 for malformed/truncated feedback; no continuation for invalid feedback

for each base model or task. This training duration was chosen because validation performance on a 20-task monitoring subset had plateaued by approximately 2,000 steps. For ALFWorld, we similarly stop at step 800 after observing a validation plateau and use that checkpoint in all evaluations.

E Critic Feedback on a GRPO-Trained Generator

Two-stage training. We first train Qwen3-4B-Instruct-2507 directly on MuSiQue in attached mode using GRPO-based optimization with the token-level DAPO objective and optimizer settings in Table 4, group size $G = 8$, batch size 128, and maximum policy staleness four. Training uses an 8,192-token context, at least three searches, and a 128-token cap per generator response. We evaluate the retained generator checkpoint at step 1,120.

We then freeze this generator and train the critic in the attached retrieval setting using the terminal success of one generator continuation per critique ($G = 8$, $N = 1$). We evaluate that critic run at step 2,000. For comparison, we also pair the trained generator with the critics trained using the original instruction-tuned generator.

Evaluation. The GRPO-generator conditions use the same 200 questions and eight trials per question, temperature 1, top-$p = 1$, an 8,192-token context, at least three searches, and no total action limit. All critics use the same self-call prompt, with at most three empty `<critic/>` requests and 512 tokens per critique. The generator selects when to call; the request carries no question payload. The critic reads the visible history and supplies general feedback, which remains in the history seen by both models on subsequent turns.

Table 5: Critic feedback on the GRPO-trained Qwen3-4B generator (step 1,120), with the original instruction-tuned generator as a reference. All evaluations with the GRPO-trained generator require at least three searches. Success rates (%) are mean ± SEM across 200 tasks, with eight rollouts per task. Both critic rows use self-call with at most three critiques per trajectory. Bold marks the highest mean; italics mark other means whose ± SEM intervals overlap the leader's (not a significance test).

Critic	Attached	Wiki
Qwen3-4B before GRPO		
No critic	21.56 ± 2.07	10.38 ± 1.59
Qwen3-4B after GRPO		
No critic	49.69 ± 2.98	*19.69* ± 2.36
CADDIE trained with original generator	51.69 ± 2.87	$\mathbf{24.44} \pm 2.64$
CADDIE trained with GRPO generator	$\mathbf{63.56} \pm 2.98$	*23.38* ± 2.63

On attached retrieval (Table 5), the critic trained with the GRPO generator improves success by 13.88 percentage points over the no-critic condition (paired task-level SEM 2.35 points), compared with 2.00 points for the critic trained with the original generator. On wiki retrieval, the corresponding gains are 3.69 points (paired task-level SEM 1.67) and 4.75 points (paired task-level SEM 1.83). The

GRPO-generator critic was trained only on attached retrieval; its wiki result measures transfer to a different retrieval setting. The results therefore support an additional benefit after generator training, with the largest gain in the retrieval setting used to train both models.

Our main goal is to train critics that transfer across base models and tasks, including settings where the base policy cannot be optimized directly. We do not aim to show that critic training outperforms task-specific optimization of the base policy. In this experiment, direct GRPO training yielded slightly better results than pairing the original instruction-tuned generator with a critic. However, training a critic with the GRPO-trained generator provided further gains after the generator's performance had plateaued, suggesting that the two approaches can be combined to improve task performance and achieve the better score.

F Results with Other Inference Protocols

The main results use self-call intervention, but a fixed intervention step can yield higher mean success in some settings. Fixed-step intervention is a special case of $\mathcal{P}_{\text{win}}$ with $\mathcal{K} = \{k\}$. Table 6 gives examples from additional MuSiQue runs, comparing the same base model and critic checkpoint within each row. These diagnostic runs allow at most one self-call per trajectory, unlike the three-call limit used in the main evaluations.

Table 6: Selected MuSiQue comparisons where fixed-step intervention has a higher mean than self-call. Both critics are Qwen3-4B models; their training retrieval settings and base models are listed below. Success rates (%) are mean ± SEM across 200 tasks, with eight rollouts per task. Self-call permits at most one critique. Bold marks the highest mean; italics mark other means whose ± SEM intervals overlap the leader's.

Critic training			Inference protocol	
Retrieval	Base model	k	Fixed step	Self-call
Evaluation: Qwen3-4B, attached				
Wiki	Qwen3-4B	4	**28.00** ± 2.47	*26.88* ± 2.46
Attached	Qwen3.8-27B	5	**30.50** ± 2.32	*28.88* ± 2.60
Evaluation: Qwen3.8-27B, wiki				
Wiki	Qwen3-4B	4	**25.94** ± 2.63	19.06 ± 2.13
Attached	Qwen3.8-27B	3	**25.69** ± 2.58	*20.94* ± 2.33

Performance can also differ substantially between neighboring intervention steps (Table 7). For example, with the $N = 1$ critic and Qwen3-4B, moving the intervention from $k = 3$ to $k = 4$ raises the mean success rate from 17.75% to 30.88%. The best reported step also depends on the base model. These are descriptive comparisons; the reported SEMs quantify uncertainty in each mean, not in the paired difference.

Table 7: Sensitivity to neighboring intervention steps on MuSiQue-attached. Each row keeps the base model and critic checkpoint fixed. Entries are success rates (%; mean ± SEM across 200 tasks, with eight rollouts per task). Bold marks the highest mean; italics mark other means whose ± SEM intervals overlap the leader's.

Base model	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Terminal-outcome critic, $N = 1$				
Qwen3-4B	22.12 ± 1.92	17.75 ± 1.66	*30.88* ± 2.18	**33.69** ± 2.35
Qwen3-30B-A3B	36.44 ± 2.42	35.44 ± 2.19	**41.50** ± 2.46	36.38 ± 2.29
Terminal-outcome critic, $N = 5$				
Qwen3-4B	23.06 ± 1.96	13.81 ± 1.37	27.88 ± 2.14	**32.56** ± 2.30
Qwen3-30B-A3B	*34.38* ± 2.36	25.19 ± 1.88	**35.94** ± 2.17	*34.75* ± 2.32

Some timing effects have a task-specific explanation. In τ^3 Airline, for example, a critique at step 2 can prevent the base model from pursuing an incorrect database entry or identifier, whereas feedback at step 3 may arrive after it has already committed to that direction. Other differences are harder to explain from the trajectories. Self-call avoids choosing a fixed step for every task and base model, which motivates its use in our main evaluations.

G Prompts and Baseline Implementations

G.1 ALFWorld: self-call evaluation

All three critic-assisted conditions use the same base-model system prompt, observation instruction, critic system prompt, and critic user-message template below.

Base model system prompt

```
You are an agent acting in a household text environment.
You can choose an environment action OR call your critic tool for
advice.
The critic tool is a real available action, separate from the
admissible environment commands.  You have
at most 50 environment actions to finish the task.  At every turn,
identify the
earliest unfinished subgoal and take the shortest action that
advances it.
Never repeat an action that did not change the state, revisit a
location without
a task reason, or explore after the needed object or receptacle is
known.

Use these compact task recipes:
- place:  find and take the object, go to the destination, open it
if needed,
then put it.
- clean:  take the object, use the sink and faucet, retrieve it,
then place it.
- heat:  take the object, use the microwave correctly, retrieve it,
then place it.
- cool:  take the object, use the fridge correctly, retrieve it,
then place it.
- examine under light:  take the object, go to a lamp, and switch
the lamp on.
- place two:  complete both placements without undoing the first.

Before every action, reason briefly in one to three short sentences
about the
current unfinished subgoal and whether to act in the world or
request a review.  Keep the reasoning under
60 words.

You have up to THREE critic reviews.  Calling the critic does not
consume an environment action.  The critic sees your task,
environment actions, current observation, admissible actions and
earlier reviews.  You cannot ask it questions:  use the EMPTY
<critic/> tool action and receive general feedback.  Keep useful
advice in mind, check it against observations, and execute the
required environment actions yourself.  Every world action must
match an admissible command exactly.

Critic reviews are part of the normal workflow, not a last resort.
Before committing to an extended search or a multi-action plan,
request a review of your planned subgoals.  Prefer a review over
speculative exploration when the object location, required
transformation, or correct completion action is unclear.  Use at
least one review while there is still enough action budget to apply
```

```
it.  After finding or taking the target object, review any uncertain
remaining prerequisites before attempting completion.  Reconsult
when the plan stops making progress.  Avoid consecutive reviews
without acting on useful advice.

Reply with exactly one of these two action forms, with no text
outside the tags:
<reason>why an independent review is useful, at most 60
words</reason>
<critic/>
OR
<reason>brief reasoning, at most 60 words</reason>
<action>one currently admissible environment action</action>
```

Instruction appended to observations

```
Available tools:  (1) critic review, by emitting <critic/>; (2) one
admissible environment action.
Reply with exactly one of these two action forms, with no text
outside the tags:
<reason>why an independent review is useful, at most 60
words</reason>
<critic/>
OR
<reason>brief reasoning, at most 60 words</reason>
<action>one currently admissible environment action</action>
Use a review before speculative exploration or an uncertain
completion.  Reviews do not consume the 50 environment actions.
```

Critic system prompt

```
You are a critic advising an agent acting in a household text
environment.
Review the task, action history, current observation, and admissible
actions.
Diagnose the trajectory so far and give concise, useful advice for
what the
agent should consider next.  Do not solve or complete the trajectory
yourself.
Do not output environment commands, actions, or a complete action
sequence.
Reply with exactly these two sections and no other text:
Analysis:  <brief analysis of the trajectory so far>
Advices for next steps:  <concise guidance without taking actions
yourself>
```

Critic user-message template

```
Task and initial situation:
{task}

Actions taken so far:
{action_history}

Current observation:
{current_observation}

Currently admissible actions:
{admissible_actions}

Critique the trajectory so far and advise the agent how to continue.
```

Each observation also states the number of critic calls remaining. After a review, the runtime supplies the current observation, admissible commands, and `Critic feedback:` fol-

lowed by the parsed critique. Earlier critiques are included in subsequent critic requests under `Previous critic feedback in this trajectory (verify against the current state):`, numbered in chronological order. Once the call limit is reached, only the environment-action form is permitted.

G.2 MuSiQue: training and self-call templates

Base model prompt

```
You are an information-retrieval agent.  Answer the multi-hop
question by
searching hidden evidence passages.  Use one action per turn.
Search for
specific entities or relations and combine evidence across results.
Do not
answer until you have enough evidence.  You have unlimited actions.
A critic may be consulted after an action.  Treat its feedback as
advice,
verify it against retrieved evidence, and continue the same
trajectory.
Reply in exactly one of these forms and nothing else:
<reason>brief reasoning, at most 60 words</reason>
<search>concise search query</search>
OR
<reason>brief reasoning, at most 60 words</reason>
<answer>short final answer only</answer>
```

Self-call instruction for base model

```
You control whether to consult an advisor.  After each search, check
whether the retrieved passages support every link needed to answer
the question.  Consider asking for help when a bridge entity or
relation is missing, passages conflict, repeated searches do not
advance the evidence chain, or a proposed answer relies on memory
instead of the retrieved evidence.  Do not ask merely because a
particular step number was reached.  Verify any advice against
retrieved passages before answering.  Consider critic/advisor as a
very important tool and consider calling it at least once:  either
when you're stuck and not sure what to do or verifying before
answering.  Critic sees the same history as you but you can't ask
him any questions, he just gives general feedback.  Use exactly one
of these three action forms, with no text outside the tags:
<reason>brief reasoning, at most 60 words</reason>
<search>concise search query</search>
OR
<reason>brief reasoning, at most 60 words</reason>
<answer>short final answer only</answer>
OR
<reason>why advice is needed, at most 60 words</reason>
<critic/>
```

Critic system prompt

```
You are a concise critic advising an information-retrieval agent.
Diagnose
whether its searches and evidence support progress on the multi-hop
question,
and advise what information or relation to seek next.  Do not answer
the
question, do not perform a search, and do not emit <search> or
<answer> tags.
Reply with exactly two sections and no other text:
```

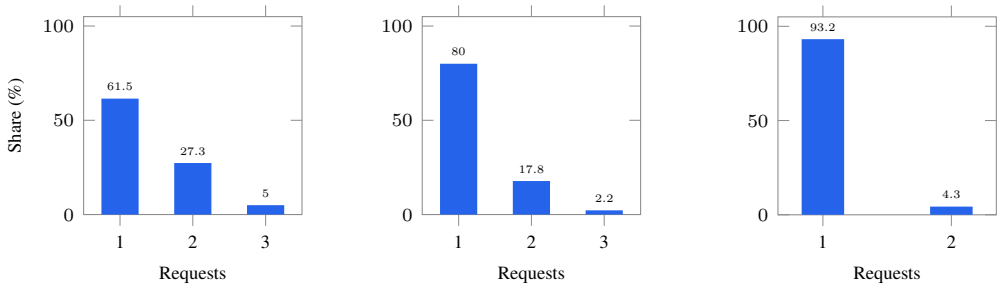
```
Analysis: <brief diagnosis of the trajectory so far>
Advices for next steps: <concise advice about what to investigate
next>
```

H OOD Self-Call Analysis

This appendix supports the analysis in Section 4. A raw attempt is any generated `call_critic` action, while delivered feedback requires a valid critic response in the trajectory. Because the generator chooses when to call, all comparisons are descriptive.

H.1 How often is Caddie called?

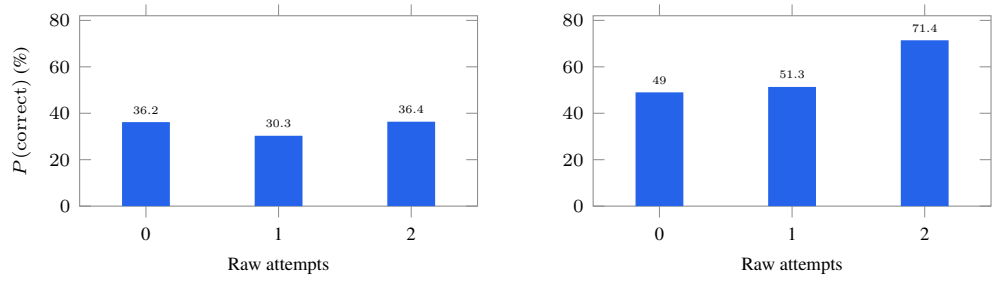
Figure 6 shows the number of requests among rollouts that call the critic at least once. One request dominates every setting. The smaller generator repeats requests more often on DeepDive; the larger generator almost always asks once.



(a) Q4 / DeepDive (b) Q4 / τ^3 Retail (c) Q38 / DeepDive

Figure 6: Critic requests per called rollout.

Figure 7 relates the number of DeepDive requests to the final answer. A second request does not coincide with lower accuracy for either generator.



(a) Q4 / DeepDive (b) Q38 / DeepDive

Figure 7: DeepDive accuracy by the number of critic requests.

H.2 What advice is given, and is it used?

Figure 8 shows two complementary views. The generator follows most advice exactly or in part. The content changes with the task: DeepDive emphasizes search and answer completion, whereas τ^3 more often requires verification, clarification, or an action in the environment.

Representative examples of followed, partially followed, ignored, and unobservable advice appear in Appendix J. They illustrate targeted evidence gathering on DeepDive and state-aware action planning on τ^3.

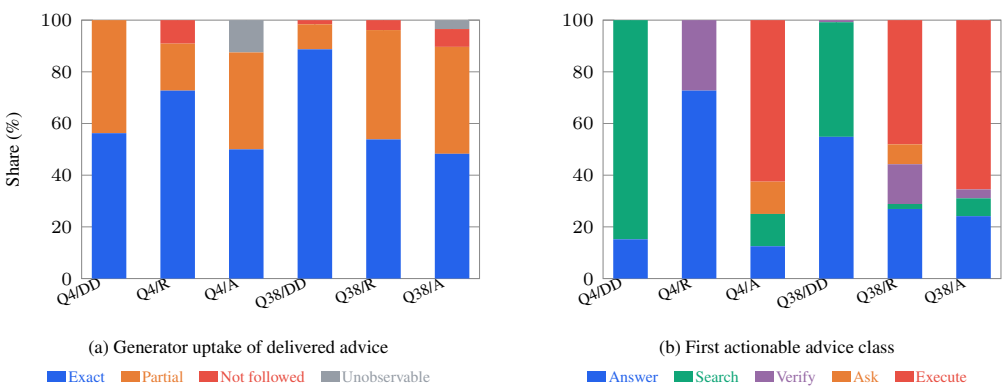


Figure 8: How generators use critic feedback and how the advice classes vary across domains.

Figure 4 in the main paper contrasts which advice is most useful for the two frozen generators. Targeted search helps Q4, while Q38 benefits more from advice to answer with the evidence already available.

Figure 9 gives the τ^3 counterpart to the DeepDive comparison in Figure 4.

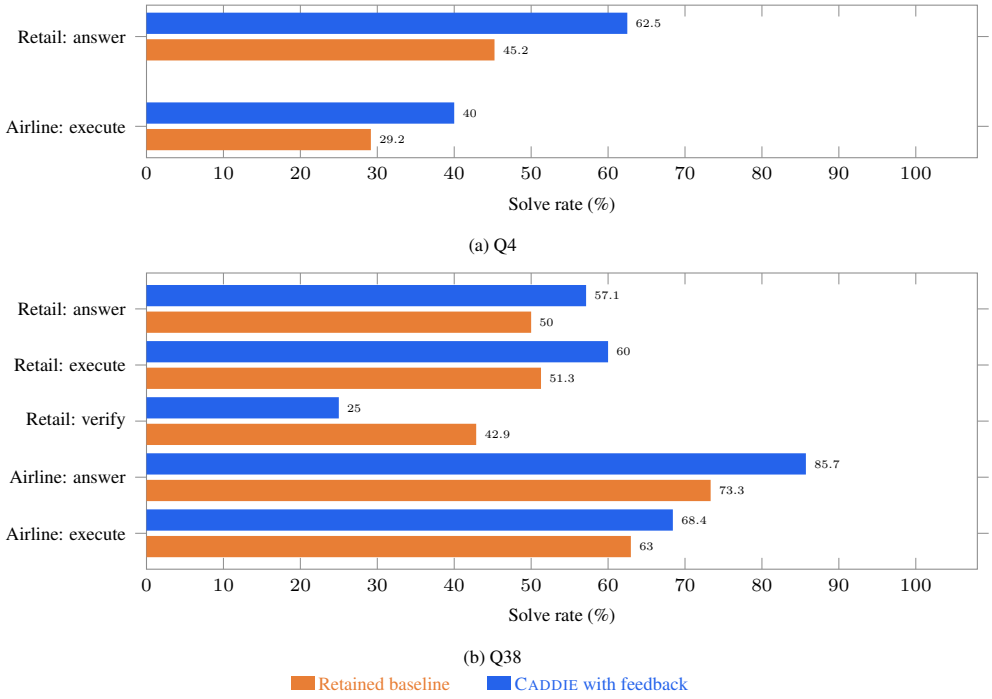


Figure 9: τ^3 solve rate by advice type.

H.3 Complementary views

Figure 10 groups the same DeepDive behavior by task family. The view is consistent with the advice analysis: useful interventions either direct the search or help the generator commit to an answer.

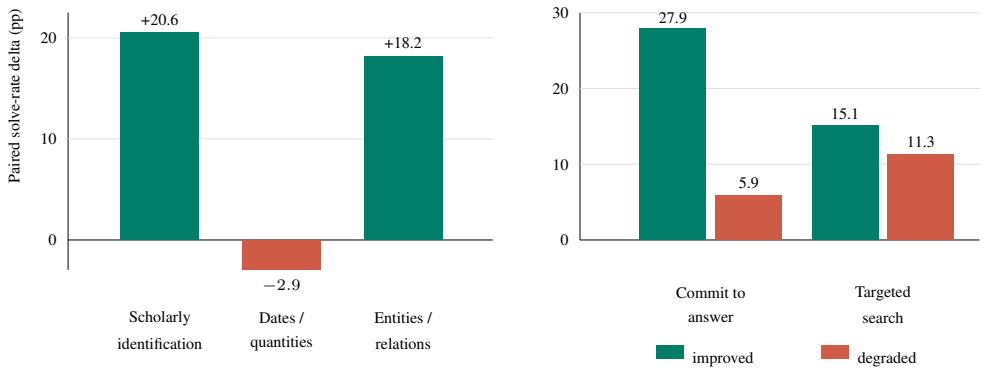


Figure 10: Qwen3.8-27B on DeepDive with the MuSiQue-wiki CADDIE critic. Left: paired solve-rate change (percentage points) on the tasks where the critic was called, by task family. Right: share of called rollouts that improved or degraded after the generator either committed to an answer or performed another search.

I PERFORMANCE ACROSS τ^3 TASK FAMILIES

We cluster τ^3 tasks by the user's primary intent and aggregate matched replicates within each family (Figure 11). This analysis compares CADDIE with the prompted critics and Agent-RRM on identical task subsets, revealing which interaction types account for differences in aggregate performance.



Figure 11: Qwen3.8-27B solve rate by τ^3 task family. Each group uses the same tasks and replicates for every critic; A/W denote the MuSiQue-attached/wiki CADDIE checkpoints.

J QUALITATIVE OOD TRAJECTORIES

We show representative task-matched trajectories from the analysis in Section 4. We compare advice from CADDIE with the critic baselines, examining how each intervention changes the generator's behavior and subsequent trajectory. The "Advice" rows reproduce the stored advice verbatim; the surrounding "Before" and "After" rows summarize the trajectory for readability.

J.1 PRESERVING TRANSACTIONAL STATE ON τ^3 RETAIL

Original payment method. The user asks to return a damaged bicycle to the card used for its order.

	Prompted Q4	**CADDIE (attached)**
Before	Finds the bicycle order, whose payment history names Mastercard ending in 8484.	Finds the same order, item, and original Mastercard.
Advice	*Proceed with issuing the refund to the Visa card ending in 6791 using the confirmed order and item details. Confirm the refund amount, tracking ID, and estimated processing time to the user before concluding the conversation.*	*Call the request_refund function with the item_id "7758198585" and payment_method "credit_card_8105988".*
After	Uses the wrong card; the return is rejected and the trajectory transfers to a human.	Returns the bicycle to the original payment method; the task succeeds.

Ordering irreversible actions. The user asks to exchange hiking boots and return two other items from the same delivered order, prioritizing the exchange.

	Kimi K3	CADDIE (wiki)
Before	Has the correct order, replacement boots, return items, and payment method.	Has the same state and the user's explicit request to complete the exchange first.
Advice	*Process the return for the hose and backpack and the exchange for the hiking boots to the confirmed waterproof variant, applying all refunds (item refund and $23.99 difference) to the original PayPal account. Then clearly confirm completion to the user.*	*Proceed with the exchange using the original order's payment method (PayPal) as it was already linked to the user's account. Provide a clear, concise confirmation message like: "Your hiking boots will be exchanged to the waterproof version. A $23.99 refund will be issued to your original PayPal account. Is this correct?" Then wait for the user to reply with "yes" to complete the exchange request.*
After	Submits the return first, changing the order state and making the boot exchange unavailable.	Completes the boot exchange first, satisfying the user's prioritized action.

Tracking a changed request. During a two-item exchange, the user changes the request and ultimately asks only to return a water bottle.

	Agent-RRM	CADDIE (attached)
Before	Reviews the earlier exchange plan before the user's final correction.	Requests feedback after the user has withdrawn both exchanges and confirmed the return.
Advice	*• Correctly used find_user_id_by_name_zip to verify identity and obtain a user ID.* *• Properly called get_user_details to retrieve full account information.* *• Accurately identified the order containing both items with get_order_details.* *• Appropriately called get_product_details for each product to find matching variants.* *• Filtered variants based on user preferences and presented a clear list.* *• Handled the unavailable original lamp variant correctly.* *• Minor: did not explicitly ask the user to confirm return or keep items, but this does not impact tool usage.* *• No irrelevant, repeated, or malformed tool calls; no fabricated arguments or filenames.*	*Call the request_return function with the parameter item_id set to "8538875209".*
After	Executes a lamp exchange; the later return cannot be completed and the trajectory transfers.	Returns only the water bottle to the original PayPal method; the task succeeds.

Ordering dependent updates. The user asks to change both the shipping address and an item in the same pending order; either item modification locks the address.

	Prompted Q4	CADDIE (wiki)
Before	Has the prior shipping address, replacement watch, and confirmation for both changes.	Has the same address, item identifiers, payment method, and confirmation.
Advice	*Verify the shipping address by retrieving it from the order #W3445693 (sneakers and air purifier) since the user specified it is the same as that order. Then confirm whether the user wants to change the silicone-to-metal strap on the white-dial variant, noting the price difference and ensuring the change is eligible—especially since the blue-dial metal version is unavailable. Only after confirming these details should the next update be initiated.*	*Call the update_order function with the following parameters:* *• order_id: "#W4219264"* *• new_shipping_address: "144 Lakeview Drive, Suite 925, New York, NY 10228, USA"* *• product_variant: {"item_id": "2407258246", "strap material": "metal", "dial color": "white"}* *After successful update, respond to the user with: "Your order has been successfully updated with the new shipping address and the metal watch with a white dial. Thank you!"*
After	Modifies the item first; the address is then locked and requires a human transfer.	Updates the address first and then the watch; both requested changes succeed.

Following the latest intent. After exploring a camera upgrade, the user abandons the modification and asks to cancel the pending order.

	Agent-RRM	**CADDIE (attached)**
Before	Reviews the earlier camera-variant plan after the user has switched to cancellation.	Sees the same revision and the still-pending order.
Advice	*• Tool calls are well sequenced: email → user → orders → order details → product variants → price difference. No unnecessary or irrelevant calls.* *• The agent correctly extracted the zoom level from the order and informed the user.* *• When the user asked for a 10x zoom, the agent used get_product_details to find the exact variant, which is appropriate.* *• The agent calculated the price difference correctly.* *• Minor: the agent did not explicitly confirm that the user still wants the same resolution and storage before proceeding, but this is implied by the order data and does not affect tool usage.* *• No hallucination of tool arguments, filenames, or outputs; all tool calls are grounded in the provided data.*	*Call the cancel_order function with order_id "#W9284598" and cancel_reason "ordered by mistake".*
After	Transfers the still-actionable cancellation to a human.	Confirms and cancels the order, refunding the original Visa; the task succeeds.

J.2 PRESERVING ITINERARY CONSTRAINTS ON τ^3 AIRLINE

Global rather than local cheapest route. The user requests the cheapest economy itinerary from Atlanta to Philadelphia on the following day.

	Kimi K3	**CADDIE (wiki)**
Before	Retains the original connection through Chicago and prices that route at $216.	Compares the available itineraries and finds a $207 connection through New York.
Advice	*Execute the reservation change immediately using update_reservation_flights with reservation_id M05KNL, cabin "economy", the two confirmed flights, and payment_id "gift_card_8887175" since the user explicitly requested the original payment method. Do not ask for further confirmation—all details have been validated and the user has given explicit consent. After the change completes, calculate the refund amount ($2,787 − $216 = $2,571) and confirm the refund will be processed to the original gift card.*	*The final action should be a simple, clear confirmation of the flight change. Since the user has confirmed the change and the refund to the original payment method, the next decision should be to generate a final response confirming the updated travel details and the refund amount. For example: "Your flight has been successfully changed to May 24. Your new route is ATL → LGA → PHL with a total cost of $207. A refund of $2,580 will be issued to your original payment method (gift card). Thank you for choosing our service!" Then end the conversation.*
After	Changes the reservation to the more expensive route.	Books the cheapest itinerary and refunds the difference to the original gift card; the task succeeds.

Completing a two-stage fare change. A Basic Economy reservation must first be upgraded before it can be moved to a cheaper nonstop flight.

	Prompted Q4	**CADDIE (wiki)**
Before	Finds the required upgrade, its temporary charge, and the cheaper nonstop itinerary.	Establishes the same policy constraint, payment method, and nonstop option.
Advice	*Verify with the user whether they are willing to proceed with the economy upgrade and confirm that the requested nonstop change is possible under current policy. Then, validate the total cost of the upgrade and flight change using accurate pricing data before finalizing the action. Avoid speculative calculations without user confirmation or system validation.*	*Call the correct function: upgrade_reservation with the reservation ID "OWZ4XL" and new cabin class "economy". Set the payment method to "credit_card_9659780" (Visa ending 1777). After successfully processing this action, confirm the upgrade and return a clear message that it has been completed. Then, follow up with a separate step to change the entire flight to the nonstop option (HAT041 from EWR to LAX).*
After	Treats the intermediate upgrade charge as exceeding the budget and transfers the user.	Executes both steps; the second change more than offsets the upgrade and the requested itinerary succeeds.

J.3 COMMITTING TO RETRIEVED EVIDENCE ON DEEPDIVE

IEEE service-chain paper. The task asks for the exact title of a 2019 paper on reliability-guaranteed deployment of service-function chains.

	Prompted Q4	**CADDIE (wiki)**
Before	Retrieves the correct author, venue, DOI, and candidate title.	Retrieves the same decisive bibliographic evidence.
Advice	*Verify the Stanford affiliation of the textbook co-author and cross-check if any of Young-Han Kim's 2019 papers on service chains or redundancy include a co-author from a Seoul-based university. Specifically, search for "Young-Han Kim 2019 IEEE paper service chain redundancy Seoul co-author" to confirm the required collaboration and match the full profile. Also, investigate if any other prominent information theorist with Stanford ties (e.g., in network information theory) meets the criteria to rule out the primary candidate.*	*Return the final answer: "An Efficient Reliability Guaranteed Deployment Scheme for Service Function Chains". This is the exact title of the intended paper and meets all the criteria: it is published in an IEEE-accessible venue (IEEE Access), fits the described research topic, and includes the correct author profile.*
After	Spends the remaining budget investigating the false mismatch and submits no valid title.	Returns *An Efficient Reliability Guaranteed Deployment Scheme for Service Function Chains*; the task succeeds.

Semiconductor growth paper. The task asks for the exact title of a paper on ex-situ calibration for commercial semiconductor heterostructures.

	Prompted Q4	**CADDIE (wiki)**
Before	Follows a different X-ray characterization paper whose authors and technical description do not match the clues.	Retrieves the exact title, abstract language, journal, year, and author affiliations.
Advice	*Investigate the full authorship and affiliations of the 2003 "Characterization of MBE grown II-VI semiconductor thin layers by X-ray interference" paper to confirm links to Surrey or Essex institutions with fellowships. Cross-reference Brian Cavenett's biography and publications to establish his PhD timeline, THz research, and industrial collaborations (especially in automotive). Then, search for publications from that period jointly citing "XRD," "calibration," and "MBE" with a co-author from Surrey or Essex who is a Fellow of the IET or FRSE.*	*Provide the final answer: "Precision growth for the manufacture of semiconductor heterostructure devices". This is the correct and most relevant paper title based on all the described conditions.*
After	Continues around the false lead and does not return the required title.	Returns *Precision growth for the manufacture of semiconductor heterostructure devices*; the task succeeds.

Across these cases, CADDIE preserves the decisive state needed for the next action. The comparator advice instead substitutes an unseen value, drops operation order, narrows the candidate set too early, or summarizes a stale request.

K ADDITIONAL DISCUSSION AND LIMITATIONS

Task structure. In our MuSiQue experiments, the base model gathers missing evidence through retrieval calls. However, in self-contained tasks, such as mathematical reasoning, the critic may instead have enough information to produce a complete solution in a single step. In our experiments on step-by-step mathematical reasoning, the critic increasingly supplied solutions that the base model could follow directly, with this behavior emerging after a few hundred training steps.

The training signal can favor this behavior. Consider a group of G critiques, each evaluated using $N > 1$ base-model continuations. General advice from the critic can steer the base model toward a particular solution, but the base model must still work out the remaining steps, so the continuations may have different outcomes and yield an intermediate average reward. If the base model reliably follows a complete solution given by the critic, the continuations tend to agree on the outcome. Correct solutions then receive rewards near one, and incorrect solutions receive rewards near zero, concentrating their rewards near the two endpoints.

Within a mixed group, complete solutions that succeed in every continuation receive the largest positive advantages. For such a solution and another critique with $R_i < 1$, the group-normalized advantage in Section 2.2 gives $\hat{A}_{\text{sol}} - \hat{A}_i = (1 - R_i)/\sigma_R > 0$, where $\sigma_R > 0$ is the group's reward

standard deviation. A consistently failing solution instead receives a negative advantage whenever the group contains successful continuations. Repeated updates can therefore reinforce correct solution generation and suppress incorrect solutions, effectively teaching the critic to solve the task itself. This incentive depends on higher mean rewards, not on reduced variance alone.

Direct solution generation can also be optimal under the critic's objective. With binary task rewards, a correct solution that the base model always copies achieves expected reward one, the maximum possible, and cannot be outperformed by advice that leaves a chance of failure. If the critic always provides a complete solution and the base model reproduces it, the critic's expected task reward reduces to

$$J(\theta) = \mathbb{E}_{s_k \sim d_{\text{win}},\, c \sim \pi_c^\theta(\cdot|s_k)} \big[r_{\text{sol}}(s_k, c) \big], \tag{7}$$

where d_{win} is the distribution of sampled state prefixes and $r_{\text{sol}}(s_k, c)$ is the task reward for the solution contained in c. In this limiting case, the critic is optimized as a solver, while the base model's parameters remain fixed. This can improve task success without establishing that the critic has learned guidance that transfers across base models or tasks. Tasks that require further environment observations make this strategy less applicable, but tool use alone does not rule it out.

Advisor Models (Asawa et al., 2026) also study a related self-contained mathematical reasoning setting. Their released implementation generates advice for a completed initial attempt and rewards the correctness of a single revised base-model response, without further tool interaction. They report 20 training epochs, but do not analyze whether the advisor increasingly supplies complete solutions.[1] Training duration may influence when this behavior appears.

Continuation sampling. A critique's reward depends on how the base model responds to it. For binary task rewards, let q_i denote the success probability of a continuation from $s_k \oplus c^{(i)}$. If the N continuations are conditionally independent, the reward estimate satisfies

$$\text{Var}\Big(R_i \mid s_k, c^{(i)} \Big) = \frac{q_i(1 - q_i)}{N}. \tag{8}$$

Thus, critiques that lead to nearly certain success or failure have lower conditional reward variance. Increasing N reduces estimation noise, but does not necessarily accelerate convergence. In our experiments, $N = 5$ yielded validation curves similar to $N = 1$, with smaller fluctuations later in training and similar final performance (Appendix A), at the cost of additional base-model continuations.

Environment restoration. Training requires continuing multiple branches from the same intermediate state. For retrieval over a fixed corpus, this can often be done by replaying the trajectory prefix. In stateful environments, the external environment must also be restored. For example, software-engineering tasks may require checkpointing files and running processes. Restoring equivalent states across branches can therefore be difficult or costly, particularly when tools interact with external services.

Critic scale. We train only Qwen3-4B critics. Although we evaluate their transfer to larger base models, we do not establish how increasing the critic's capacity affects training, performance, or transfer.

Training cost. Reaching a performance plateau in our experiments required approximately 2,000 training steps. Each sampled state requires GN base-model continuations, so training cost depends on the base model and environment as well as the critic. Expensive tool execution, long trajectories, and larger base models can substantially increase CPU and GPU requirements.

[1] The released math training script specifies 10 epochs over 1,000 examples with batches of 16, implying up to $10\lfloor 1000/16 \rfloor = 620$ outer training iterations after dropping incomplete batches, before prompt-length filtering. This differs from the paper's 20 epochs, so it does not establish the exact duration of the reported run.