ON-POLICY SELF-DISTILLATION WITH OUTCOME-ALIGNED PRIVILEGED INFORMATION FOR AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning (RL) trains large language model agents through long-horizon interactions, but sparse trajectory-level rewards provide limited guidance on individual decisions. Recent on-policy self-distillation (OPSD) methods supplement sparse rewards with dense supervision derived from natural-language skills or reflections used as training-time privileged information (PI). However, discrete textual PI prevents gradients from flowing directly through the generated tokens to optimize PI construction. Learnable latent PI alleviates this bottleneck, but learnability alone does not ensure that its influence on sampled actions is aligned with relative trajectory outcomes. To address this gap, we propose **Pilot** (**P**rivileged **I**nformation **L**earning with **O**utcome-aligned **T**raining), a framework that learns outcome-aligned latent PI to guide on-policy self-distillation for agentic RL. **Pilot** constructs latent PI from source trajectories and measures its influence on distinct same-task query trajectories through sampled-action log-likelihood differences with and without PI. With the actor held fixed, it trains the PI analyzer to align each query's average bounded influence with relative outcome credit derived from the same-task on-policy rollout group. The measured influence weights an OPSD objective optimized jointly with direct RL, yielding an actor that requires neither privileged inputs nor the analyzer at inference. We evaluate **Pilot** with Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct on ALFWorld and WebShop. **Pilot** outperforms the evaluated RL and OPSD baselines on all aggregate task metrics across both benchmarks and model scales.Our code is available at https://anonymous.4open.science/status/PILOT-478B.

1 INTRODUCTION

Large language model (LLM) agents can continually perceive, reason, and act in open-ended environments, enabling long-horizon applications including embodied interaction, web navigation, and tool-augmented reasoning (Shridhar et al., 2021; Yao et al., 2022; Wang et al., 2024). Reinforcement learning (RL) improves such agents using feedback from environments or verifiers (DeepSeek-AI et al., 2025). In particular, critic-free algorithms based on group-relative advantages, such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024), are widely used for post-training LLM agents. When extended to multi-turn interaction, these methods typically estimate an advantage from the final trajectory outcome and apply the resulting signal throughout the sampled trajectory (Wang et al., 2025b; Dong et al., 2026). A single terminal reward, however, does not indicate which intermediate decisions contributed to progress and which led to failure (Feng et al., 2026; Tao et al., 2026; Huo et al., 2026). This ambiguity worsens over long horizons, where an early action's consequences may emerge after many interactions (He et al., 2026; Wang et al., 2026d; Li et al., 2026).

To complement sparse outcome rewards, a growing line of work augments agentic RL with dense supervision computed on actions sampled by the current policy (Wang et al., 2026b; Liu et al., 2026). On-policy distillation (OPD) provides such supervision by using token-level signals from a teacher to guide student-generated actions (Agarwal et al., 2024). Building on this idea, recent on-policy self-distillation (OPSD) methods instantiate the teacher as the policy itself, conditioned on privileged context available only at training time (Zhao et al., 2026; Penaloza et al., 2026). The self-teacher additionally conditions on privileged information (PI), which the student never sees (Lu et al., 2026; Yang et al., 2026b). Existing agentic OPSD methods commonly derive this PI from

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.

completed trajectories as natural-language skills, reflections, or behavioral guidance, allowing the supervision to evolve with the current policy's experience (Wu et al., 2026; Zheng et al., 2026; Wang et al., 2026c). However, verbalizing experience into discrete text introduces a representational bottleneck: fine-grained decision information may be discarded, and the discrete interface prevents gradients from directly shaping PI construction according to its downstream utility. Learnable latent PI alleviates these limitations by compressing experience into continuous representations and optimizing them end to end (Zhang et al., 2026). Yet continuity and learnability alone do not ensure that the policy change induced by PI is aligned with relative outcome credit derived from same-task rollout outcomes. Figure 1 summarizes this progression.

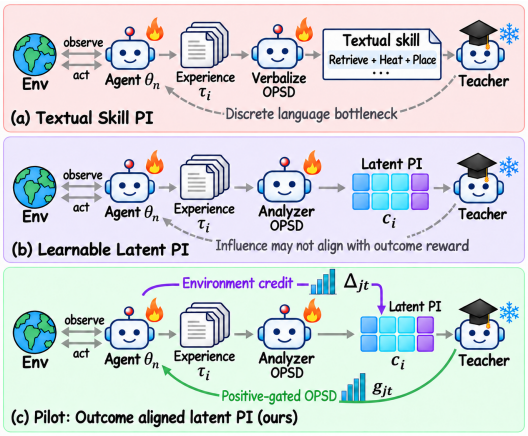

Figure 1: Comparison of textual PI, learnable latent PI, and Pilot. Pilot aligns the cross-trajectory influence of latent PI with relative outcome credit and uses measured influence to weight self-distillation into a PI-free actor at inference.

The distinction between PI influence and task-relevant decision quality is particularly important in cross-trajectory learning. Unlike within-trajectory self-distillation, which can only reuse a rollout's own hindsight, pairing distinct trajectories from the same task allows a source rollout to provide hindsight context for actions sampled in a query rollout. However, the source may contain details specific to its own execution path. For a sampled action, a higher likelihood with PI shows only that the PI supports the action, not that the action advances the task; a lower likelihood, likewise, may reflect mismatched experience rather than reliable negative evidence. Without outcome-based alignment, latent PI may therefore exert a substantial yet directionally unreliable influence on another trajectory's decisions, as illustrated in Figure 1. Unlike process-reward or step-level RL, which assign supervision to individual decisions (Lightman et al., 2024; Xi et al., 2026; Cui et al., 2026), trajectory-level outcomes cannot directly establish whether PI benefits a particular query action. This leads to our central question: *How can an agent learn latent PI from on-policy cross-trajectory experience while aligning its average influence on query trajectories with relative outcome credit?*

To answer this question, we introduce **Pilot** (**P**rivileged **I**nformation **L**earning with **O**utcome Alignment for On-Policy **T**raining), an outcome-aligned latent PI framework for agentic RL. For the same task, Pilot forms source–query pairs from trajectories sampled by the current policy, assigns multiple distinct sources to each query, and uses an Analyzer to compress each source trajectory into latent PI. Under a fixed actor snapshot, Pilot measures the PI's relative influence as the difference between the log-likelihoods assigned to the same sampled query action with and without PI. Pilot then derives relative outcome credit by normalizing each query outcome within its same-task rollout group while excluding its source set from the reference statistics, and trains the Analyzer to align the query's average PI influence with this credit. During actor optimization, Pilot uses a sigmoid gate to assign greater weight to sampled actions with positive PI influence while retaining a smaller, nonzero weight for negative influence; the resulting self-distillation objective is jointly optimized with the direct RL objective. Rollout and deployment use only standard interaction history; PI is restricted to training-time scoring. Pilot thus turns current-policy experience into continuous privileged supervision shaped by relative outcome credit, without discrete skill verbalization.

Our main contributions are as follows: ❶ We identify and formulate a key limitation of learnable latent PI: its average influence on a query trajectory need not agree with relative outcome credit. ❷ We develop outcome-aligned cross-trajectory PI learning that measures PI influence by scoring the same action with and without PI and aligns the query's average influence with relative outcome credit. ❸ We introduce PI-guided on-policy self-distillation that weights the sampled-action objective with a sigmoid gate based on PI influence, jointly optimizes it with direct RL, and yields an actor requiring no PI at inference. ❹ Across ALFWorld and WebShop, Pilot with Qwen2.5-3B/7B-Instruct outperforms GRPO and representative OPSD methods.

2 Related Work

Agentic Reinforcement Learning. Reinforcement learning (RL) has become an important post-training paradigm for large language models because it can optimize behaviors difficult to specify with token-level labels (Jin et al., 2025; Luo et al., 2025). Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is an outcome-based method that learns from environment or verifier feedback without a learned critic. In agentic RL, a policy repeatedly reasons, acts, uses tools, and incorporates observations (Yao et al., 2023) in ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). Yet these environments often provide sparse, delayed trajectory-level rewards: a terminal score reveals the outcome but offers little guidance about intermediate decisions (Wang et al., 2025a; Tan et al., 2026; Wang & Ammanabrolu, 2025). The difficulty of assigning credit to intermediate decisions motivates denser supervision at the decision or token level alongside outcome-based RL (Setlur et al., 2025; Wang et al., 2026a; Zhou et al., 2025).

On-Policy Distillation and On-Policy Self-Distillation. On-policy distillation (OPD) reduces the train–test mismatch of conventional knowledge distillation by having a teacher evaluate prefixes of trajectories sampled from the current student policy, providing token-level guidance on learner-visited states (Agarwal et al., 2024). On-policy self-distillation (OPSD) removes the need for a separate teacher: the same policy acts as a student under ordinary context and as a self-teacher conditioned on training-time privileged information (PI), such as answers, feedback, skills, or reflections (Zhao et al., 2026; Penaloza et al., 2026). Comparing its predictions with and without PI provides fine-grained signals that complement sparse outcome rewards, while PI can be removed at inference time (Yang et al., 2026a). However, these signals depend on the quality and relevance of PI; noisy or mismatched context may yield unreliable guidance across interaction turns.

On-Policy Self-Distillation with Privileged Information. Recent work combines agentic RL with OPSD by constructing PI from skills or reflections (Wang et al., 2026c). SDAR (Lu et al., 2026) uses a gated skill-conditioned objective to attenuate unreliable negative guidance while retaining outcome-based RL. OPID (Yang et al., 2026b) extracts episode-level and step-level hindsight skills and routes fine-grained skills to critical decisions. SEED (Wu et al., 2026) couples the rollout actor with a trajectory analyzer so that hindsight skills evolve with the policy, while GRSD (Zheng et al., 2026) contrasts successful and failed rollouts to construct outcome-discriminative reflections. These methods use discrete natural-language PI, whose generation is outside the differentiable path of the policy objective. LOPD (Zhang et al., 2026) instead learns continuous latent PI from retrieved experience, making PI trainable through its effect on the policy. Pilot further aligns the influence of cross-trajectory latent PI on sampled actions with leave-source-out relative trajectory outcomes, then selectively distills positive PI influence into a PI-free actor.

3 Method

We introduce **Pilot**, an outcome-aligned latent PI framework for agentic RL. Pilot learns privileged context from completed trajectories and uses its influence to train a PI-free actor.

3.1 Problem Setup and Overview

We draw a task x from the task distribution $\mathcal{D}$. At interaction step t, the agent observes history $h_t = (x, o_1, a_1, \ldots, o_t)$ and generates an action $a_t \sim \pi_\theta(\cdot \mid h_t)$ autoregressively, where θ denotes the actor parameters. The environment returns reward r_t and the next observation until termination. A trajectory is $\tau = \{(h_t, a_t, r_t)\}_{t=1}^{T}$, where $T \leq H$ is the realized number of interaction steps and H is the maximum horizon. Given task return $\mathcal{R}(\tau)$, the objective is

$$\max_\theta J(\theta), \qquad J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \tau\sim\pi_\theta(\cdot|x)}[\mathcal{R}(\tau)].$$

Here, $J(\theta)$ is the expected task return under the task and trajectory sampling shown above. Building on learnable latent context (Zhang et al., 2026), Pilot first collects PI-free on-policy trajectories and pairs distinct rollouts of the same task. A trainable analyzer compresses each source trajectory into latent PI and learns to align its influence on the paired query with relative outcome credit. PI influence then weights an auxiliary self-distillation loss for the actor, alongside direct RL (Figure 2). Here, on-policy refers to experience collected by the current actor snapshot; the privileged branch only re-scores sampled actions and collects no additional environment rollouts.

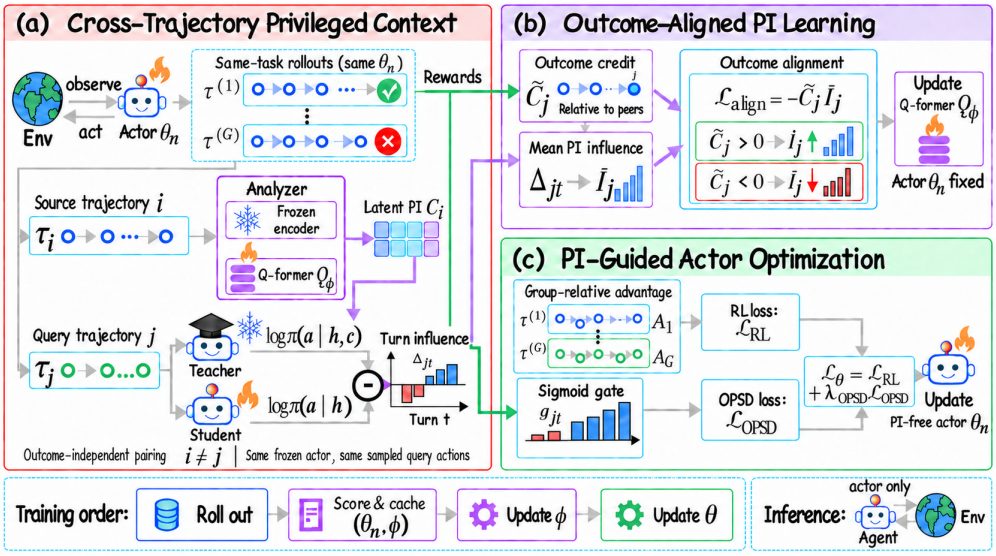

Figure 2: **Overview of Pilot.** (a) A PI-free actor pairs same-task rollouts to measure PI influence on query actions. (b) The analyzer aligns mean PI influence with relative outcome credit. (c) The actor combines RL with sigmoid-gated OPSD; only the PI-free actor is deployed.

3.2 Cross-Trajectory Privileged Context

Collecting and pairing experience. At iteration n, we fix actor snapshot θ_n and collect $G \geq J_{\text{src}} + 2$ PI-free trajectories under the same task initialization:

$$\mathcal{B}_x = \{\tau_{xi}\}_{i=1}^{G}, \qquad \tau_{xi} \sim \pi_{\theta_n}(\cdot \mid x). \tag{1}$$

Here, $\mathcal{B}_x$ is the on-policy rollout group for task x, G is its number of trajectories, and $i \in \{1, \ldots, G\}$ indexes trajectories in the group. We record an environment-specific terminal outcome R_{xi} for PI learning; this signal does not replace the reward stream used by direct RL. For each query j, outcome-independent pairing selects a set $\mathcal{S}_{xj}$ of J_{src} distinct source trajectories from the same task group, with $j \notin \mathcal{S}_{xj}$. The sources and query share the task but need not visit identical states. Below, we omit x when describing a single task group.

Encoding source experience. The analyzer combines a frozen trajectory encoder E_ω with a trainable QFormer-style compressor Q_ϕ (Li et al., 2023):

$$H_i = \text{sg}[E_\omega(\text{Serialize}(\tau_i))], \qquad c_i = Q_\phi(H_i) \in \mathbb{R}^{K \times d}, \tag{2}$$

where ω denotes the frozen encoder parameters, H_i encodes source trajectory i, Serialize converts that trajectory into the ordered source sequence, and sg denotes stop-gradient. The symbol ϕ denotes the trainable compressor parameters, $c_i \in \mathbb{R}^{K \times d}$ is the resulting latent-PI matrix, K is the number of latent tokens, and d is the actor's embedding dimension. For query j, we concatenate the source representations in a fixed order as $c_j^{\text{PI}} = \text{Concat}_{i \in \mathcal{S}_j} c_i \in \mathbb{R}^{J_{\text{src}} K \times d}$. The privileged Analyzer uses this context to score query actions; Appendix A.2 details its construction.

Measuring PI influence. Using the same fixed actor, we score the sampled query tokens with and without source PI:

$$\begin{aligned} \ell_{jt\ell}^{\text{PI}}(\phi) &= \log \pi_{\theta_n}(a_{jt\ell} \mid \bar{h}_{jt}, a_{jt,<\ell}, c_j^{\text{PI}}), \\ \ell_{jt\ell}^{0} &= \text{sg}\left[\log \pi_{\theta_n}(a_{jt\ell} \mid \bar{h}_{jt}, a_{jt,<\ell})\right], \end{aligned} \tag{3}$$

Here, j indexes the query trajectory, t the interaction turn, and the subscript ℓ the target-token position; $a_{jt,<\ell}$ is the prefix preceding that position. The leading symbol ℓ denotes a token log-probability, while superscripts PI and 0 identify the with-PI and PI-free branches. Both branches use the same retained history $\bar{h}_{jt}$, target tokens, and action prefixes. With binary valid-action-token

mask $M^{\mathrm{PI}}_{jt\ell} \in \{0,1\}$, whose value is one exactly for scored action tokens, the turn-level influence is

$$\Delta_{jt}(\phi) = \frac{\sum_\ell M^{\mathrm{PI}}_{jt\ell}\big(\ell^{\mathrm{PI}}_{jt\ell}(\phi) - \ell^{0}_{jt\ell}\big)}{\sum_\ell M^{\mathrm{PI}}_{jt\ell}}. \tag{4}$$

Positive Δ indicates increased conditional support for the sampled action, not a causal return gain. During analyzer learning, the actor parameters remain fixed, while gradients propagate through the actor's PI-conditioned input to the compressor Q_ϕ.

3.3 Outcome-Aligned PI Learning

Relative outcome credit. For query j, let $\mathcal{R}_j = [G] \setminus \mathcal{S}_j$ be the leave-source-out reference set, which excludes all source trajectories and retains j. We define

$$\mu_j = \frac{1}{|\mathcal{R}_j|}\sum_{k\in\mathcal{R}_j} R_k, \qquad \sigma_j^2 = \frac{1}{|\mathcal{R}_j|}\sum_{k\in\mathcal{R}_j}(R_k-\mu_j)^2, \qquad A^{\mathrm{LSO}}_{j|-\mathcal{S}_j} = \frac{R_j-\mu_j}{\sigma_j+\epsilon}. \tag{5}$$

Here, μ_j and σ_j are the reference mean and population standard deviation, and $\epsilon > 0$ stabilizes normalization. The uncentered credit measures query performance relative to this reference. We then center detached credits over the valid queries $\mathcal{J}_x$ of each task:

$$C_{xj} = \mathrm{sg}\Big[A^{\mathrm{LSO}}_{xj|-\mathcal{S}_{xj}}\Big], \qquad b_x = \frac{1}{|\mathcal{J}_x|}\sum_{j\in\mathcal{J}_x} C_{xj}, \qquad \widetilde{C}_{xj} = \mathrm{sg}(C_{xj}-b_x). \tag{6}$$

Here, C_{xj} is the detached uncentered credit, b_x is its mean over valid queries for task x, and $\widetilde{C}_{xj}$ is the detached centered credit.

Aligning average influence. Let $\mathcal{T}_{xj}$ denote a query's valid scored turns. Its average bounded influence is

$$\overline{I}_{xj}(\phi) = \frac{1}{|\mathcal{T}_{xj}|}\sum_{t\in\mathcal{T}_{xj}} \tanh\!\left(\frac{\Delta_{xjt}(\phi)}{T_\Delta}\right), \tag{7}$$

where $T_\Delta > 0$ controls saturation. Letting x range over sampled tasks in the current rollout batch, $N_Q = \sum_x |\mathcal{J}_x|$ is the total number of valid queries. The analyzer minimizes

$$\mathcal{L}_\phi = -\frac{1}{N_Q}\sum_x\sum_{j\in\mathcal{J}_x} \widetilde{C}_{xj}\,\overline{I}_{xj}(\phi) + \lambda_a\mathcal{L}_{\mathrm{anchor}} + \lambda_\Delta\mathbb{E}_w[\Delta(\phi)^2]. \tag{8}$$

The alignment term encourages average influence to agree with centered outcome credit, giving valid queries equal weight. This trajectory-level supervision does not assign credit to individual turns. The nonnegative coefficients λ_a and λ_Δ weight the anchor and squared-influence penalties, respectively. $\mathcal{L}_{\mathrm{anchor}}$ limits latent drift from the initial compressor, while $\mathbb{E}_w$ denotes a query-equal reduction that averages turns within each query and then averages queries. Within $\mathbb{E}_w$, $\Delta(\phi)$ is shorthand for the indexed turn influence $\Delta_{xjt}(\phi)$. Only ϕ is updated. Filtering, reductions, and the anchor are specified in Appendix A.3; Appendix B.1 analyzes the alignment objective.

3.4 PI-Guided Actor Optimization

Pilot updates the PI-free actor by minimizing a joint objective that combines a direct RL loss based on environment-reward advantages with an OPSD loss weighted by PI influence.

Group-relative RL Loss. For the G same-task rollouts, let R^{RL}_{xi} denote the scalar return used by the RL backbone. We first compute the detached group-relative advantage

$$\widehat{A}^{\mathrm{RL}}_{xi} = \frac{R^{\mathrm{RL}}_{xi} - \mathrm{mean}_{k\in[G]}\,R^{\mathrm{RL}}_{xk}}{\mathrm{std}_{k\in[G]}\,R^{\mathrm{RL}}_{xk} + \epsilon_{\mathrm{RL}}}, \tag{9}$$

and broadcast it to the valid action tokens of rollout i. With old-policy snapshot θ_n, the token-level importance ratio is

$$\rho_{xit\ell}(\theta) = \exp[\log\pi_\theta(a_{xit\ell} \mid h_{xit}, a_{xit,<\ell}) - \log\pi_{\theta_n}(a_{xit\ell} \mid h_{xit}, a_{xit,<\ell})]\,. \tag{10}$$

We display the standard clipped GRPO surrogate (Shao et al., 2024) as

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = -\frac{1}{N_{\mathrm{act}}} \sum_{xit\ell} M^{\mathrm{act}}_{xit\ell} \min\Big(\rho_{xit\ell}(\theta) \widehat{A}^{\mathrm{RL}}_{xi}, \mathrm{clip}(\rho_{xit\ell}(\theta), 1-\varepsilon, 1+\varepsilon) \widehat{A}^{\mathrm{RL}}_{xi} \Big). \tag{11}$$

Here, M^{act} is the valid actor action-token mask and $N_{\mathrm{act}} = \sum M^{\mathrm{act}}$. The training implementation retains the configured negative-advantage dual clipping and reduction convention; these choices are shared by Pilot and its GRPO baseline. The RL branch is

$$\mathcal{L}_{\mathrm{RL}}(\theta) = \mathcal{L}_{\mathrm{GRPO}}(\theta) + \lambda_{\mathrm{KL}} \mathcal{L}_{\text{ref-KL}}(\theta), \tag{12}$$

where the reference KL regularizes the PI-free actor against a frozen reference policy.

Sigmoid-gated OPSD Loss. Following SDAR's sigmoid gap gate (Lu et al., 2026), we cache the detached turn-level weight before updating the analyzer:

$$g_{xjt} = \mathrm{sg}[\sigma(s\Delta_{xjt}(\phi_n))], \qquad s > 0. \tag{13}$$

For valid scored turns, positive influence gives $g_{xjt} > 1/2$, zero influence gives $g_{xjt} = 1/2$, and negative influence receives a smaller but nonzero weight. Invalid source sets and unscored turns receive zero weight. For the same sampled token and prefix used in Equation (3), define the teacher–student likelihood gap

$$\delta^{\mathrm{OPSD}}_{xjt\ell}(\theta) = \mathrm{sg}\big[\ell^{\mathrm{PI}}_{xjt\ell}(\phi_n)\big] - \log \pi_\theta(a_{xjt\ell} \mid h_{xjt}, a_{xjt,<\ell}). \tag{14}$$

The OPSD objective is

$$\mathcal{L}_{\mathrm{OPSD}}(\theta) = \frac{1}{N_{\mathrm{act}}} \sum_{xjt\ell} M^{\mathrm{act}}_{xjt\ell} g_{xjt} \delta^{\mathrm{OPSD}}_{xjt\ell}(\theta). \tag{15}$$

The two gaps have different roles: the turn-averaged Δ measures PI influence and weights supervision through g, whereas the token-level δ^{OPSD} expresses the actor update. Because the teacher score and gate are detached, their contribution is constant with respect to θ; Equation (15) therefore has the same actor gradient as the implemented gate-weighted sampled-action NLL. It is a sampled-token log-ratio surrogate rather than full-distribution matching. Related agentic self-distillation methods use privileged-context likelihood gaps either in a separate auxiliary loss (Lu et al., 2026; Wu et al., 2026) or as a token- or turn-level policy advantage (Yang et al., 2026b; Zheng et al., 2026); Pilot keeps environment-reward advantage confined to the RL branch.

Joint update and deployment. The actor objective combines the two branches:

$$\mathcal{L}_\theta = \mathcal{L}_{\mathrm{RL}} + \lambda_{\mathrm{OPSD}} \mathcal{L}_{\mathrm{OPSD}}. \tag{16}$$

Each iteration computes influence and caches gates with (θ_n, ϕ_n), updates ϕ with Equation (8), and then updates θ with Equation (16). No gradient passes through either optimizer update. At deployment, only π_θ is retained.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate Pilot on two interactive agent benchmarks. ALFWorld (Shridhar et al., 2021) consists of text-based household tasks spanning six families: Pick, Look, Clean, Heat, Cool, and Pick2. WebShop (Yao et al., 2022) requires an agent to search for products, inspect their attributes, and make a purchase that satisfies a natural-language request. These environments test multi-turn decision making in embodied and web-based settings, respectively.

Baselines. We compare Pilot with the instruction-tuned backbone without post-training (Vanilla), inference-time Skill-Prompt, and the post-training methods OPSD (Zhao et al., 2026), GRPO, Skill-GRPO, GRPO+OPSD, Skill-SD (Wang et al., 2026c), RLSD (Yang et al., 2026a), SDAR (Lu et al., 2026), OPID (Yang et al., 2026b), and LOPD (Zhang et al., 2026). An asterisk marks methods that receive a skill prompt during evaluation; all unmarked methods, including Pilot, act from the ordinary interaction history. These baselines span prompting, outcome-based RL, and self-distillation.

Evaluation Metrics. For ALFWorld, we report per-family success rates and the episode-weighted overall success rate (Avg.). For WebShop, we report the mean normalized task-completion score and exact success rate. All metrics are expressed as percentages, with higher values indicating better performance. We evaluate the deployed actor without an analyzer, latent PI, or skill prompt.

Table 1: **Performance Comparison on ALFWorld and WebShop.** We report ALFWorld success rates and WebShop score and success rate (all in %). * denotes evaluation with a skill prompt. **Best** and second-best results are highlighted per backbone and metric; ties share the marking.

Method	ALFWorld							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	Avg.	Score	Succ.
Qwen2.5-3B-Instruct									
Vanilla	44.4	11.1	6.2	15.4	28.6	12.5	21.9	6.7	0.8
Skill-Prompt*	51.7	66.7	48.4	0.0	4.3	10.0	28.9	0.2	0.8
OPSD	48.8	41.7	16.7	0.0	15.8	16.7	28.1	11.3	3.1
GRPO	91.2	62.5	96.2	61.9	65.0	47.4	75.0	79.8	63.3
Skill-GRPO	88.9	71.4	58.8	70.6	40.7	29.2	60.2	77.3	60.9
GRPO+OPSD	**100.0**	82.4	85.7	75.0	70.0	60.0	81.2	77.8	66.4
Skill-SD	88.2	50.0	96.2	52.4	65.0	57.9	73.4	75.9	64.0
RLSD	87.9	75.0	90.9	75.0	73.1	68.4	79.7	84.4	66.4
SDAR	97.1	62.5	**100.0**	61.9	75.0	**84.2**	84.4	85.0	68.0
OPID	92.7	**100.0**	88.9	70.0	**84.2**	70.0	84.3	85.0	74.2
LOPD	96.6	**100.0**	**100.0**	88.2	76.0	69.6	87.5	85.2	76.6
Pilot	97.4	80.0	**100.0**	**100.0**	83.3	80.0	**91.4**	**90.6**	**82.0**
Qwen2.5-7B-Instruct									
Vanilla	36.1	22.2	3.1	0.0	0.0	0.0	12.5	5.9	1.6
Skill-Prompt*	51.7	50.0	32.3	5.3	4.3	0.0	23.4	1.7	0.8
OPSD	50.0	60.0	22.7	21.4	17.6	9.5	32.8	4.5	2.3
GRPO	91.2	87.5	96.2	81.0	65.0	57.9	81.2	80.9	72.6
Skill-GRPO	88.5	66.7	65.2	61.1	57.7	73.1	69.5	80.4	71.9
GRPO+OPSD	91.4	61.5	**100.0**	87.5	76.5	52.2	80.4	86.8	76.5
Skill-SD	93.9	**93.8**	90.9	**100.0**	69.2	68.4	85.1	86.1	76.5
RLSD	**100.0**	87.5	92.3	58.8	80.0	65.2	82.0	87.4	77.3
SDAR	94.7	75.0	**100.0**	86.7	68.2	78.9	85.9	89.4	82.8
OPID	**100.0**	81.8	97.1	**100.0**	80.8	80.0	90.0	85.3	79.7
LOPD	97.1	84.6	**100.0**	87.5	64.7	**91.3**	89.8	88.8	79.7
Pilot	92.1	75.0	**100.0**	93.8	**92.6**	87.5	**92.2**	**90.3**	**83.6**

Implementation Details. We use Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2025) as backbones. Pilot constructs distinct source–query pairs from on-policy trajectories and jointly trains a latent-PI analyzer and a PI-free actor using GRPO with influence-weighted self-distillation. Each update samples 16 tasks with eight rollouts per task. Analyzer credit is given by the final task-completion score on WebShop and the episode reward on ALFWorld. Full hyperparameters and preprocessing details are provided in Appendix C.3. A per-update computational-cost breakdown and comparison with baselines are provided in Appendix D.2.

4.2 Main Results

Leading aggregate performance. Pilot achieves the best results on all three aggregate metrics at both model scales. Compared with the strongest baseline for each metric, it improves ALFWorld average success, WebShop score, and WebShop success by 3.9, 5.4, and 5.4 points with Qwen2.5-3B, and by 2.2, 0.9, and 0.8 points with Qwen2.5-7B, respectively. Pilot's gains also extend to individual ALFWorld tasks: it leads the next best method by 11.8 points on Heat with Qwen2.5-3B and on Cool with Qwen2.5-7B.

Outcome-aligned latent PI advances privileged self-distillation. Pilot outperforms the skill-based self-distillation methods Skill-SD, SDAR, and OPID on every aggregate metric at both model scales. It also consistently surpasses LOPD, which learns continuous latent PI without explicitly aligning its policy influence with relative trajectory outcomes. Compared with LOPD, Pilot improves all three aggregate metrics by 3.9–5.4 points with Qwen2.5-3B-Instruct and by 1.5–3.9 points with Qwen2.5-7B-Instruct. These results show that replacing textual skills with a learnable latent representation is not sufficient by itself; aligning the influence of latent PI with outcome-derived credit produces more effective privileged supervision. All Pilot results use the actor alone, without privileged inputs or the analyzer at inference time.

4.3 Training Dynamics

Training dynamics on WebShop. Figure 3(a) compares the validation success rates of Pilot and GRPO. Pilot reaches a success rate of 0.5 at approximately step 55, about 40 training steps earlier than GRPO, and maintains higher success through most of training. As shown in Figure 3(b), Pilot also reduces the mean episode length earlier: its sharp initial decline coincides with the rise in vali-

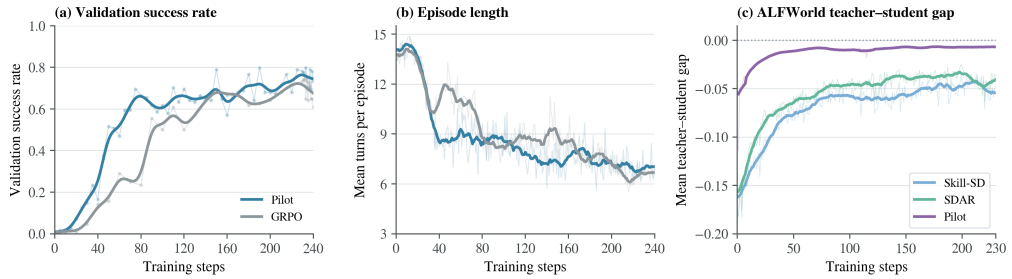

Figure 3: **(a) and (b) Training dynamics with Qwen2.5-3B-Instruct on WebShop. (c) Mean teacher–student gap with Qwen2.5-3B-Instruct on ALFWorld.**

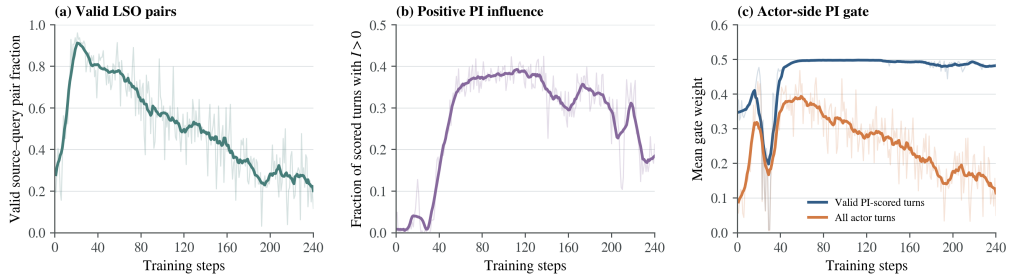

Figure 4: **Outcome credit, PI influence, and actor-side use in Pilot on WebShop.**

dation success. The two methods approach similar episode lengths by the end of training. Together, these trends show that Pilot improves task success and trajectory efficiency earlier in training.

Teacher–student gap dynamics. Figure 3(c) shows that the magnitudes of all three teacher–student gaps decrease rapidly early in ALFWorld training. The Skill-SD and SDAR curves then level off, while Pilot maintains a small turn-averaged gap through step 230. For Pilot, this small signed mean indicates a modest net shift in sampled-action log likelihood under PI conditioning. Figure 4 reports the outcome-credit and PI-influence diagnostics.

Analyzer influence and actor-side use. In the outcome-only, sigmoid-gated Qwen2.5-3B-Instruct run on WebShop, the Analyzer learns from relative trajectory credit and measures PI influence by rescoring the same sampled actions with and without PI. Figure 4(a) shows valid LSO coverage rising to about 0.9 before declining to 0.2. Figure 4(b) shows the fraction of scored turns with positive PI influence increasing from near zero to about 0.4 in mid-training. In Figure 4(c), the mean gate on valid PI-scored turns stays near 0.5, while the all-turn average falls with coverage. Together, these trends show that outcome-aligned PI affects action scores and provides selective supervision to the actor. Additional validation and training-reward curves appear in Appendix D.3.

4.4 Ablation Studies and Analysis

Table 2 analyzes three Pilot designs on WebShop with Qwen2.5-3B-Instruct: constructing latent PI from a distinct same-task source trajectory, continually learning an outcome-aligned analyzer, and selectively internalizing positive PI influence into the actor. Each variant modifies one component while retaining the PI-free deployment interface. We report best validation performance (%).

Table 2: **Ablation study of Pilot on WebShop.**

Method	Score	Succ.
Pilot	90.6	82.0
Cross-trajectory PI construction		
Self-trajectory PI ($i = j$)	84.5	75.8
Outcome-aligned analyzer learning		
Frozen Warm-started analyzer	77.7	69.5
Shuffled Query Credits	88.7	78.1
Positive-influence actor internalization		
Shuffled Turn Gates	84.8	68.8
Mass-matched Uniform Gate	85.1	75.0

Impact of PI construction across trajectories. Pilot constructs latent PI from a source trajectory distinct from its paired query trajectory. Using the query trajectory itself instead reduces the WebShop success rate from 82.0% to 75.8% and the task score from 90.6 to 84.5. This result favors transferring hindsight between dis-

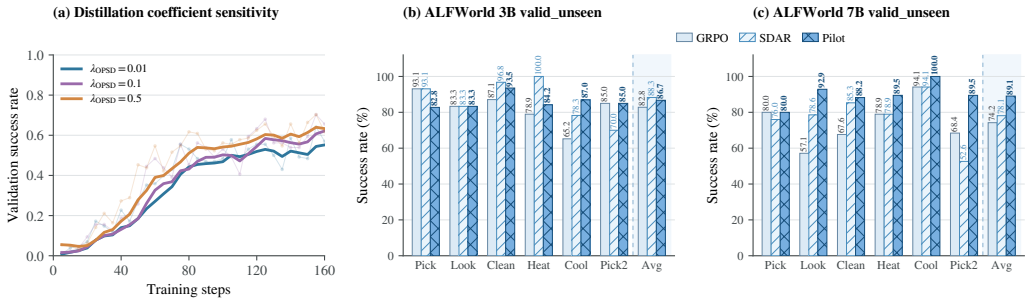

Figure 5: **Hyperparameter sensitivity and generalization.** (a) Effect of the self-distillation weight λ_{OPSD} on WebShop with Qwen2.5-3B-Instruct. (b) and (c) show generalization performance on the ALFWorld unseen split with Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct, respectively.

tinct rollouts for the same task over constructing privileged context from the query's own completed trajectory, supporting Pilot's pairing of source and query trajectories.

Impact of outcome-aligned analyzer learning. Freezing the warm-started analyzer leaves the latent PI pathway intact but reduces the WebShop success rate from 82.0% to 69.5% and the task score from 90.6 to 77.7. This result shows that the analyzer must continue adapting as the actor evolves for its PI guidance to remain effective. Shuffling query-level outcome credits reduces the success rate to 78.1% and the task score to 88.7, disrupting the link between each query's relative outcome credit and the PI influence learned for that query. Together, these results support continual analyzer updates and correctly matching outcome credit to PI influence.

Impact of positive-influence actor internalization. Shuffling Pilot's turn gates yields a peak WebShop success rate of 68.8% and a separate peak task score of 84.8, compared with 82.0% and 90.6 for Pilot. Its success rate is also lower than that of the mass-matched uniform gate (68.8% vs. 75.0%), despite similar peak scores (84.8 vs. 85.1). These results show that turn-specific gating matters: the actor benefits when PI-derived supervision targets the scored query turn.

4.5 Cross-Domain Generalization and Sensitivity Analysis

Effect of the self-distillation weight λ_{OPSD}. Figure 5(a) compares $\lambda_{\text{OPSD}} \in \{0.01, 0.1, 0.5\}$. The largest tested value, $\lambda_{\text{OPSD}} = 0.5$, performs best late in training. A small weight provides too little PI-guided supervision for the actor to internalize the learned information, while increasing the weight to 0.1 improves performance but remains below 0.5. The stronger distillation setting more effectively transfers the learned PI into the PI-free actor.

Generalization to unseen ALFWorld tasks. On ALFWorld Unseen, Pilot achieves an average success rate of 86.7% with Qwen2.5-3B-Instruct, exceeding GRPO by 3.9 points and slightly below SDAR (88.3%). With Qwen2.5-7B-Instruct, Pilot reaches 89.06%, outperforming GRPO and SDAR by 14.84 and 10.93 points, respectively. Its clearest 7B gains over SDAR are on Pick Two & Place (+36.84) and Look at Object in Light (+14.29). Detailed per-family results are provided in Appendix D.1 (Table 6).

5 Conclusion

We introduced Pilot, an outcome-aligned latent PI framework for agentic RL. Pilot encodes distinct same-task source trajectories, aligns their average influence on query actions with leave-source-out relative outcome credit, and uses influence-weighted self-distillation to train an actor requiring no PI at inference. Across ALFWorld and WebShop with 3B and 7B Qwen2.5 backbones, Pilot improves over outcome-only GRPO and achieves the strongest aggregate results among evaluated methods at both scales. Training dynamics show earlier improvements in task success and trajectory efficiency, while ablations support cross-trajectory PI, continual outcome-aligned analyzer learning, and turn-specific gating. These findings highlight that learnable privileged information should be evaluated not merely by its influence on the policy, but by how that influence relates to observed outcomes. Pilot provides a way to learn and internalize such supervision while preserving PI-free deployment.

AI USE STATEMENT

We used generative AI tools to assist with improving the clarity, grammar, and readability of portions of the manuscript. Generative AI tools were also used to support literature retrieval and discovery, including identifying potentially relevant prior work. All cited references were independently located, read, and verified by the authors. Generative AI tools additionally assisted with research execution, primarily by supporting the implementation and debugging of experimental code and suggesting experimental configurations and analyses. All AI-assisted code was manually reviewed and tested, and all experimental results and interpretations were independently verified by the authors. The authors made all final research decisions and take full responsibility for the content, claims, results, code, and artifacts presented in this work.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=9SkkifLopZ.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, January 2025. URL https://doi.org/10.48550/arXiv.2501.12948.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 16981–17017, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Shuo He, Lang Feng, Xin Cheng, Lei Feng, Bo An, et al. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, volume 2026, pp. 27572–27593, 2026.

Yifu Huo, Shunjie Xing, Chenglong Wang, Peinan Feng, Qiaozhi He, Yan Ding, Anxiang Ma, Yuxin Gao, Tongran Liu, Tong Xiao, et al. Learning from environmental feedback: Credit assignment across multiple timescales for agentic reinforcement learning. *arXiv preprint arXiv:2608.08255*, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with

reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Junbo Li, Peng Zhou, Rui Meng, Meet P Vadera, Lihong Li, and Yang Li. Turn-ppo: Turn-level advantage estimation with ppo for improved multi-turn rl in agentic llms. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6227–6243, 2026.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023.

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*, volume 2024, pp. 39578–39601, 2024.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *International Conference on Learning Representations*, volume 2026, pp. 129271–129291, 2026.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. *CoRR*, abs/2508.03680, August 2025. URL https://doi.org/10.48550/arXiv.2508.03680.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ebZcMQImhG.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 60808–60838, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. URL https://doi.org/10.48550/arXiv.2402.03300.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, 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.

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

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. Trace: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026.

Daoyu Wang, Qingchuan Li, Mingyue Cheng, Jie Ouyang, Shuo Yu, Qi Liu, and Enhong Chen. Steppo: Step-aligned policy optimization for agentic reinforcement learning. *arXiv preprint arXiv:2604.18401*, 2026a.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, et al. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *International Conference on Learning Representations*, volume 2026, pp. 104502–104529, 2026b.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *CoRR*, abs/2505.20732, May 2025a. URL https://doi.org/10.48550/arXiv.2505.20732.

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

Ruiyi Wang and Prithviraj Ammanabrolu. A practitioner's guide to multi-turn agentic reinforcement learning. In *First Workshop on Multi-Turn Interactions in Large Language Models*, 2025. URL https://openreview.net/forum?id=yPWJG9wgll.

Yunan Wang, Minghui Song, Zihan Zhang, Shaohan Huang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Group-graph policy optimization for long-horizon agentic reinforcement learning. *arXiv preprint arXiv:2606.22995*, 2026d.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073, April 2025b. URL https://doi.org/10.48550/arXiv.2504.20073.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, et al. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via stepwise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195, 2026.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026a.

Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, et al. Opid: On-policy skill distillation for agentic reinforcement learning. *arXiv preprint arXiv:2606.26790*, 2026b.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

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

Guibin Zhang, Jiayang Lyu, Ran Sun, Xinlei Yu, Haoyu Zhao, Qibing Ren, and Shuicheng Yan. Latent on-policy self-distillation. *arXiv preprint arXiv:2608.13040*, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*, 2026. URL https://openreview.net/forum?id=31wkVZXEaJ.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-reflective self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.28076*, 2026.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025.

A Training Procedure and Implementation Details

A.1 Trajectory Pairing

For each task, Pilot samples J_{src} outcome-independent derangements $\rho_x^{(1)}, \ldots, \rho_x^{(J_{\mathrm{src}})}$ such that $\rho_x^{(r)}(j) \neq j$ and $\rho_x^{(r)}(j) \neq \rho_x^{(r')}(j)$ for every query j and $r \neq r'$. We set $\mathcal{S}_{xj} = \{\rho_x^{(r)}(j)\}_{r=1}^{J_{\mathrm{src}}}$, where $1 \leq J_{\mathrm{src}} \leq G-2$. Every trajectory appears J_{src} times as a source and J_{src} times as a query, producing GJ_{src} source–query assignments. Source representations are concatenated in increasing r order. Sources and queries need not visit identical states. The same rollout group is reused for direct RL and PI learning.

A.2 Context Construction and Token Masks

Each source trajectory is serialized by concatenating its recorded textual contexts and actions in turn order, subject to a fixed per-source token budget. The numerical outcome enters the PI learning objective rather than being appended as an explicit source field. The frozen encoder and trainable compressor produce continuous latent tokens without an intermediate natural-language skill description. For privileged scoring, the query history is retained under a separate scoring-context budget. The PI-conditioned and PI-free scoring branches use the same retained query history, sampled target token IDs, and action prefixes; they differ only in whether the concatenated latent PI tokens are prepended to the input embeddings. The PI-free actor objective uses the ordinary actor context h_{xjt} in Equation (14); the equal-context comparison here defines influence Δ_{xjt}.

The scoring mask $M^{\mathrm{PI}}_{xjt\ell}$ selects valid sampled action tokens and excludes observations, prompts, and padding. The actor mask $M^{\mathrm{act}}_{xjt\ell}$ applies the corresponding validity rules to the PI-free actor input. The separate scoring-context budget can leave fewer scored action tokens than valid actor action tokens in a turn. Turns for which no valid sampled action token remains after scoring-mask construction are excluded from PI scoring. Outcome-invalid queries, invalid source sets, and unscored turns receive gate value $g_{xjt} = 0$. For a scored turn, its gate applies to all valid actor action tokens, including any without an individual PI score. In the displayed OPSD gap, we set the detached teacher-score term to zero at actor-token positions lacking a PI score; this is a bookkeeping convention, not an imputed teacher prediction. The convention changes the objective only by a term constant with respect to θ and leaves the implemented gate-weighted sampled-action NLL gradient unchanged.

A.3 Credit Filtering and Batch Reduction

For each query and its source set, we compute the set-LSO statistics in Equation (5). A query is excluded from PI learning if these statistics are nonfinite or if the reference-set variance is below a threshold ν. Its trajectory remains available for the direct RL objective. Let $\mathcal{J}_x$ contain queries whose set-LSO statistics are finite, whose reference-set variance is at least ν, whose source set is valid, and for which at least one valid scoring turn remains. Let $\mathcal{T}_{xj}$ denote the valid turn set for query j. To implement the query-balanced average used in the main text, we assign

$$w_{xjt} = \frac{1}{|\mathcal{T}_{xj}|} \tag{A.1}$$

and define query-balanced averaging as

$$\mathbb{E}_w[f] = \frac{\sum_x \sum_{j \in \mathcal{J}_x} \sum_{t \in \mathcal{T}_{xj}} w_{xjt} f_{xjt}}{\sum_x \sum_{j \in \mathcal{J}_x} \sum_{t \in \mathcal{T}_{xj}} w_{xjt}}. \tag{A.2}$$

Thus, valid queries receive equal weight irrespective of their numbers of valid turns, while tasks with more valid queries contribute proportionally more weight to the rollout-batch average.

To limit drift from initialization, we retain a frozen copy ϕ_0 of the initial compressor and define

$$\mathcal{L}_{\mathrm{anchor}} = \mathbb{E}_w \left[\frac{\| c^{\mathrm{PI}}_{xj} - \mathrm{sg}[c^{\mathrm{PI}}_{xj,0}] \|_F^2}{J_{\mathrm{src}} K d} \right]. \tag{A.3}$$

Here, $c^{\mathrm{PI}}_{xj,0}$ concatenates $Q_{\phi_0}(H_{xi})$ for $i \in \mathcal{S}_{xj}$ in the same source order. The anchor compares the current and initial representations of the source set and requires no offline skill-annotation stage.

For the actor objective, let

$$N_{\mathrm{act}} = \sum_{xjt\ell} M^{\mathrm{act}}_{xjt\ell} \tag{A.4}$$

be the number of valid sampled actor tokens in the rollout batch. The displayed actor objectives are evaluated only when $N_{\mathrm{act}} > 0$. If no valid actor action token remains, Pilot sets the auxiliary term to zero and skips the actor update for this batch, so $\theta_{n+1} = \theta_n$; the displayed token-normalized GRPO surrogate is not evaluated. The denominator in Equation (15) counts all valid actor action tokens, including tokens whose turn-level gates are zero, rather than normalizing by the total gate weight. This preserves attenuation when gate weights are small or few actor tokens have valid PI scores. Following SDAR's sigmoid gap-gating principle (Lu et al., 2026), Pilot uses a turn-level gate rather than SDAR's token-level gate: $g_{xjt} = \mathrm{sg}[\sigma(s\Delta_{xjt}(\phi_n))]$ for valid scored turns. Here $s > 0$ controls sharpness and Δ_{xjt} averages the PI-induced action log-probability difference over the sampled tokens in turn t. The gate exceeds $1/2$ for positive influence, equals $1/2$ at zero influence, and lies below $1/2$ for negative influence. Thus negative influence receives a smaller but nonzero auxiliary weight; zero weights are reserved for outcome-invalid queries, invalid source sets, and unscored turns.

A.4 Update Order and Degenerate-Batch Handling

Algorithm 1 summarizes the three-stage iteration. All sigmoid influence gates g_{xjt} (Equation (13)) are computed with (θ_n, ϕ_n) and cached *before* any analyzer parameter update. The detached PI-conditioned teacher scores in Equation (14) likewise refer to (θ_n, ϕ_n). Because the teacher-score term is constant with respect to θ, the implemented gate-weighted sampled-action NLL requires only the cached gates for the actor gradient. The updated analyzer ϕ_{n+1} is first used for scoring in the next iteration.

During analyzer optimization, the encoder, actor snapshot, detached outcome credits, and anchor reference remain fixed. During actor optimization, the cached gates remain fixed, and neither the auxiliary loss nor the RL loss updates the analyzer. No gradient is propagated through either optimizer update. Deployment discards the encoder, compressor, and privileged scoring branch.

The GRPO term in Equation (16) is the configured clipped policy surrogate without the separately written reference-policy regularizer. Reward processing, clipping, and reduction are inherited from the agent RL backbone and must be held fixed in matched comparisons. The reference KL is measured against a frozen PI-free policy, not the privileged scoring branch.

If no valid PI samples remain after filtering, Pilot skips the analyzer update and sets the auxiliary self-distillation loss to zero. When valid actor action tokens remain, it continues direct RL with its reference-policy regularizer. If only one valid query remains for a task, or all of its valid queries have identical uncentered credits, centering makes that task's alignment contribution zero; the regularization terms can still contribute to the analyzer objective.

B Theoretical Analysis of Pilot

We analyze two mechanisms specific to Pilot: how relative trajectory-outcome credit supervises cross-trajectory privileged information (PI), and how PI influence reweights the PI-free actor's sampled-action likelihood. The statements concern fixed rollout data and local update objectives, rather than policy improvement or convergence.

B.1 Credit-Aligned Cross-Trajectory PI Learning

Fix an iteration and the actor snapshot used for on-policy rollouts. For task x, let query j receive a distinct, non-self source set $\mathcal{S}_{xj}$, with $c^{\mathrm{PI}}_{xj}(\phi) = \mathrm{Concat}_{i\in\mathcal{S}_{xj}} Q_\phi(H_{xi})$. At a valid query turn t, $\Delta_{xjt}(\phi)$ denotes the mean action-token log-likelihood under the same frozen actor with source-derived PI minus that without PI. Both branches score the same sampled tokens and retained ordinary history. Let $\mathcal{V}_{xjt} = \{\ell : M^{\mathrm{PI}}_{xjt\ell} = 1\}$ and $L_{xjt} = |\mathcal{V}_{xjt}| > 0$. The influence is the length-normalized

Algorithm 1 Pilot Training

Require: Initial actor π_{θ_0} and compressor Q_{ϕ_0}
Require: Frozen encoder E_ω and reference actor $\pi_{\theta_{\text{ref}}}$
Require: Group size G, source count J_{src}, and number of iterations N
for $n = 0, \ldots, N-1$ **do**
  Stage 1: Collect and pair on-policy experience
  for each sampled task x **do**
    $\mathcal{B}_x \leftarrow \textsc{Rollout}(\pi_{\theta_n}, x, G)$
    Record trajectory outcomes $\{R_{xi}\}_{i=1}^{G}$
    $\{\mathcal{S}_{xj}\}_{j=1}^{G} \leftarrow \textsc{DisjointDerangements}(G, J_{\text{src}})$
  end for
  Stage 2: Learn outcome-aligned latent PI
  Initialize $g_{xjt} \leftarrow 0$ for all actor turns in the batch
  for each task group $\mathcal{B}_x$ **do**
    Compute set-LSO credits $\{C_{xj}\}$ and identify outcome-valid queries
    for each outcome-valid query j in $\mathcal{B}_x$ **do**
      for each $i \in \mathcal{S}_{xj}$ **do**
        $c_{xi} \leftarrow \textsc{Compress}(\tau_{xi}; \phi_n)$
      end for
      $c_{xj}^{\text{PI}} \leftarrow \text{Concat}_{i \in \mathcal{S}_{xj}}\, c_{xi}$
      for each valid scored turn t of query j **do**
        $\Delta_{xjt} \leftarrow \textsc{Influence}(\tau_{xj}, t, c_{xj}^{\text{PI}}; \theta_n)$
        $g_{xjt} \leftarrow \textsc{SigmoidGate}(\Delta_{xjt})$; cache g_{xjt}
      end for
    end for
  end for
  For each task x, form $\mathcal{J}_x$ from outcome-valid queries with at least one scored turn
  $N_Q \leftarrow \sum_x |\mathcal{J}_x|$
  if $N_Q > 0$ **then**
    Center $\{C_{xj}\}_{j \in \mathcal{J}_x}$ to obtain $\{\widetilde{C}_{xj}\}_{j \in \mathcal{J}_x}$
    $\phi_{n+1} \leftarrow \textsc{UpdateAnalyzer}(\phi_n, \mathcal{L}_\phi)$
  else
    $\phi_{n+1} \leftarrow \phi_n$; set $\mathcal{L}_{\text{OPSD}} \leftarrow 0$
  end if
  Stage 3: Update the PI-free actor with cached gates
  $N_{\text{act}} \leftarrow \sum_{xjt\ell} M_{xjt\ell}^{\text{act}}$
  if $N_{\text{act}} > 0$ **then**
    $\theta_{n+1} \leftarrow \textsc{UpdateActor}(\theta_n, \mathcal{L}_\theta)$
  else
    $\theta_{n+1} \leftarrow \theta_n$
  end if
end for
return PI-free actor π_{θ_N}

log-likelihood ratio over these scored token positions:

$$\Delta_{xjt}(\phi) = \frac{1}{L_{xjt}} \log \frac{\prod_{\ell \in \mathcal{V}_{xjt}} \pi_{\theta_n}(a_{xjt\ell} \mid \bar{h}_{xjt}, a_{xjt,<\ell}, c_{xj}^{\text{PI}}(\phi))}{\prod_{\ell \in \mathcal{V}_{xjt}} \pi_{\theta_n}(a_{xjt\ell} \mid \bar{h}_{xjt}, a_{xjt,<\ell})}. \tag{B.1}$$

Thus $\Delta > 0$ means that PI increases the frozen policy's conditional support for the scored tokens of the sampled action; it does not by itself imply a causal improvement in environmental return.

Pilot assigns the query-level outcome credit

$$C_{xj} = \text{sg}\left[A_{xj|-\mathcal{S}_{xj}}^{\text{LSO}}\right], \qquad b_x = \frac{1}{|\mathcal{J}_x|} \sum_{j \in \mathcal{J}_x} C_{xj}, \qquad \widetilde{C}_{xj} = \text{sg}(C_{xj} - b_x), \tag{B.2}$$

where $\mathcal{J}_x$ contains valid queries and $\mathcal{T}_{xj}$ contains their valid scoring turns. The credit is detached and shared by all valid turns of a query. Set-LSO excludes the source-set outcomes from a query's uncentered credit; task centering can reintroduce dependence on those outcomes through other queries' credits, so this construction does not imply statistical independence. With $w_{xjt} = 1/|\mathcal{T}_{xj}|$, each

valid query has unit total weight,

$$\sum_{t\in\mathcal{T}_{xj}} w_{xjt} = 1, \qquad \mathbb{E}_w[f] = \frac{1}{\sum_x |\mathcal{J}_x|}\sum_x \sum_{j\in\mathcal{J}_x} \frac{1}{|\mathcal{T}_{xj}|}\sum_{t\in\mathcal{T}_{xj}} f_{xjt}. \tag{B.3}$$

Consequently, trajectory length alone does not increase a query's total weight, while tasks with more valid queries contribute proportionally more weight to the rollout-batch objective. Pilot minimizes

$$\mathcal{L}_\phi = -\mathbb{E}_w\left[\widetilde{C}\tanh(\Delta(\phi)/T_\Delta)\right] + \lambda_\Delta \mathbb{E}_w[\Delta(\phi)^2] + \lambda_a \mathcal{L}_{\text{anchor}}. \tag{B.4}$$

Because $\widetilde{C}_{xj}$ is trajectory-level, the query-balanced turn average gives the alignment statistic $N_Q^{-1}\sum_x \sum_{j\in\mathcal{J}_x} \widetilde{C}_{xj}\,\overline{I}_{xj}(\phi)$, where $\overline{I}_{xj}(\phi)$ is defined in Equation (7). The corresponding term in the minimized analyzer loss is its negative, as in Equation (8). The $\mathbb{E}_w$ form used here is more convenient for differentiation. All expectations and derivatives below condition on fixed rollouts, pairings, validity masks, and credits.

Proposition 1 (Outcome-credit alignment and the zero-influence state). For every task with valid PI samples,

$$\frac{1}{|\mathcal{J}_x|}\sum_{j\in\mathcal{J}_x} \widetilde{C}_{xj}\,\overline{I}_{xj} = \mathrm{Cov}_{j\sim\mathrm{Unif}(\mathcal{J}_x)}(C_{xj}, \overline{I}_{xj}). \tag{B.5}$$

Hence an influence whose query-level average is constant across the task yields zero alignment term. Suppose $\Delta(\phi^\circ) = 0$ for every valid sample and Δ is differentiable at ϕ°. Then

$$\nabla_\phi \mathcal{L}_\phi(\phi^\circ) = -\frac{1}{T_\Delta}\mathbb{E}_w[\widetilde{C}\nabla_\phi\Delta(\phi^\circ)] + \lambda_a \nabla_\phi \mathcal{L}_{\text{anchor}}(\phi^\circ). \tag{B.6}$$

In particular, if the anchor gradient vanishes there and the batch-aggregated credit-weighted influence gradient $\mathbb{E}_w[\widetilde{C}\nabla_\phi\Delta(\phi^\circ)]$ is nonzero, zero influence is not a stationary point.

Proof. Equation (B.3) first averages turns within each query, yielding the query-level average $\overline{I}_{xj}$. Substituting into the task-level alignment term gives $|\mathcal{J}_x|^{-1}\sum_j (C_{xj} - b_x)\overline{I}_{xj}$. Since b_x is the uniform query mean of C_{xj}, expanding the covariance proves Equation (B.5). For Equation (B.6), differentiate Equation (B.4), use $\tanh'(0) = 1$, and note that the squared-influence penalty has zero gradient at $\Delta = 0$. □

Proposition 1 characterizes the learning signal supplied by outcome credit: within each task, the alignment statistic is the covariance between query-level credit and mean PI influence. A constant mean influence across queries therefore provides no alignment gain. Under the stated gradient condition, the objective also supplies a local direction away from zero influence. Because credit is assigned at the trajectory level, this signal aligns each query's average PI influence with its outcome credit without distinguishing the contributions of individual turns.

B.2 Sigmoid-Gated Internalization as Influence-Weighted Imitation

Before the analyzer update, Pilot caches the detached turn gate

$$g_{xjt} = \mathrm{sg}[\sigma(s\Delta_{xjt}(\phi_n))], \qquad s > 0. \tag{B.7}$$

For finite influence in a valid scored turn, the sigmoid gate lies in $(0, 1)$ and increases with PI support for the sampled action. Negative influence is attenuated rather than hard-masked, and zero influence receives weight $1/2$. Outcome-invalid queries, invalid source sets, and unscored turns receive zero gate. Every valid actor action token m in a scored turn receives that turn's gate, even if the scoring-context budget excluded that token from PI scoring. With $N_{\text{act}} > 0$ total valid actor action tokens, including tokens without valid PI scores, the auxiliary loss is

$$\mathcal{L}_+(\theta) = -\frac{1}{N_{\text{act}}}\sum_{m=1}^{N_{\text{act}}} g_m \log \pi_\theta(a_m \mid u_m), \tag{B.8}$$

where u_m is the ordinary context available without PI. Equation (B.8) differs from $\mathcal{L}_{\text{OPSD}}$ in Equation (15) only by the detached teacher-score term, using the zero-extension convention for unscored actor-token positions specified in Appendix A.2. This term is constant with respect to θ; it is an actual PI-conditioned score only at scored positions. Therefore $\nabla_\theta \mathcal{L}_+ = \nabla_\theta \mathcal{L}_{\text{OPSD}}$; below, $\lambda_+ = \lambda_{\text{OPSD}} \geq 0$.

Table 3: **Benchmarks used in the Pilot experiments.** The analyzer outcome is the trajectory-level signal used for relative PI credit; evaluation metrics are computed from the deployed PI-free actor.

Benchmark	Interaction setting	Analyzer outcome	Evaluation metrics
ALFWorld	Text-based household tasks	Episode reward	Success rate by family; Avg.
WebShop	Product search and purchase	Final graded task score	Mean score; exact success

Table 4: **Training and evaluation pools for Pilot.** Training counts denote tasks or goals available for sampling, not cumulative rollout trajectories. The evaluation column distinguishes the available validation pool from the number sampled in each validation pass.

Benchmark	**Training pool**	**Evaluation pool and sampling**
ALFWorld	3,553 training games	140 seen validation games; 128 sampled episodes per pass
WebShop	6,410 training goals	500 held-out goals; 128 distinct goals sampled per pass

Proposition 2 (Exact token-instance reweighting and update scale). Let

$$\bar{g} = \frac{1}{N_{\mathrm{act}}} \sum_{m=1}^{N_{\mathrm{act}}} g_m. \tag{B.9}$$

If $\bar{g} > 0$, define a distribution over observed token instances by

$$q_g(m) = \frac{g_m}{N_{\mathrm{act}} \bar{g}}. \tag{B.10}$$

Then

$$\mathcal{L}_+(\theta) = \bar{g}\, \mathbb{E}_{m \sim q_g} \left[-\log \pi_\theta(a_m \mid u_m) \right]. \tag{B.11}$$

If $\bar{g} = 0$, no actor token has a valid nonzero gate and $\mathcal{L}_+(\theta) = 0$; a valid scored turn with finite influence cannot have a zero sigmoid gate. Moreover, since $0 \le g_m < 1$ and the gates are detached, if $\|\nabla_\theta \log \pi_\theta(a_m \mid u_m)\| \le B$ for valid tokens, the auxiliary contribution to the actor objective satisfies

$$\left\| \nabla_\theta \big(\lambda_+ \mathcal{L}_+(\theta) \big) \right\| \le \lambda_+ B \bar{g}. \tag{B.12}$$

Proof. The gates are nonnegative and $\sum_m q_g(m) = 1$. Substituting Equation (B.10) into Equation (B.11) recovers Equation (B.8). Differentiating the detached-gate loss and applying the triangle inequality gives $\|\nabla_\theta(\lambda_+ \mathcal{L}_+)\| \le \lambda_+ N_{\mathrm{act}}^{-1} \sum_m g_m B = \lambda_+ B \bar{g}$. □

The identity shows that the gate continuously reweights observed context–action token instances, while $\bar{g}$ preserves attenuation when gate weights are small or few tokens have valid PI scores. It does not construct a full-vocabulary teacher distribution or provide targets for unsampled actions. The bound controls the auxiliary term's scale but does not imply alignment with the GRPO gradient; cancellation can also make an ungated aggregate gradient smaller than its gated counterpart. Cached gates block actor-loss gradients into the analyzer during the current update.

Together, the reweighting identity and gradient bound show how PI-derived gates shape the actor's sampled-action update while preserving attenuation through fixed token-count normalization.

C Additional Experimental Detail

C.1 Datasets

We evaluate Pilot on two interactive benchmarks that require an agent to act over multiple environment steps. Table 3 summarizes the interaction setting, the trajectory-level outcome used to train Pilot's analyzer, and the reported evaluation measures. The analyzer outcome is distinct from the reward stream used by the direct RL objective (Section 4.1).

Table 4 reports the environment task pools used for training and evaluation.

ALFWorld. ALFWorld (Shridhar et al., 2021) presents household goals through a text interface. The agent receives observations and issues actions over a sequence of interaction steps. We report

success rates for the six task families used in the main results: Pick, Look, Clean, Heat, Cool, and Pick2. The reported Avg. pools successes over all evaluated episodes, so each episode has equal weight; it is not the unweighted mean of the six family-level rates. For Pilot's PI learning, the trajectory outcome is the episode reward.

WebShop. WebShop (Yao et al., 2022) asks the agent to satisfy a natural-language shopping request by searching the catalog, inspecting products, choosing relevant attributes, and completing a purchase. We report the mean normalized task-completion score, which reflects graded satisfaction of the request, and the exact-success rate. Both measures are expressed as percentages in the main results. Pilot uses the final graded task-completion score as the analyzer's trajectory outcome. These benchmark-specific outcomes enter leave-source-out credit construction; Pilot does not use a local progress signal in the formulation studied here.

C.2 Baselines

The main comparison includes prompting-only, outcome-based RL, and self-distillation methods. An asterisk in Table 1 denotes access to a natural-language skill during evaluation. Unmarked methods, including Pilot, are evaluated from the ordinary task prompt and interaction history, without privileged skill or latent PI input. We group the baselines below by the training signal and the role of privileged context.

Prompting-only methods. **Vanilla** evaluates the instruction-tuned backbone without task-specific post-training. **Skill-Prompt*** leaves the backbone parameters unchanged but supplies a task-relevant natural-language skill at evaluation. Their comparison separates the effect of an inference-time skill from parameter updates.

Outcome-based reinforcement learning. **GRPO** (Shao et al., 2024) optimizes the actor using group-relative advantages derived from environment rewards, without a privileged teacher signal. It is the direct RL reference for Pilot's additional PI learning and self-distillation pathway. **Skill-GRPO** retains outcome-based policy optimization while supplying a natural-language skill during training; the unmarked variant is evaluated without that skill. It tests whether skill-conditioned training can improve the policy that acts from ordinary history alone.

Self-distillation and hybrid methods. **OPSD** (Zhao et al., 2026) uses a privileged self-teacher to provide an auxiliary learning signal on actions generated by the student policy. **GRPO+OPSD** (Shao et al., 2024; Zhao et al., 2026) combines the outcome-based GRPO objective with that self-distillation objective, providing a direct comparison for coupling reward learning and privileged supervision without Pilot's learned, outcome-aligned latent PI. **Skill-SD** (Wang et al., 2026c) conditions a training-time self-teacher on natural-language skills and trains a student that acts without them. **RLSD** (Yang et al., 2026a) uses the privileged teacher–student likelihood gap to modulate the outcome-based policy update. **SDAR** (Lu et al., 2026) adds a gated skill-conditioned self-distillation objective alongside GRPO. **OPID** (Yang et al., 2026b) constructs episode- and step-level hindsight skills from on-policy trajectories and routes the finer-grained guidance to selected decisions. These methods differ in how privileged information is constructed and how its signal enters policy optimization; the main table reports their deployed-policy behavior under the evaluation-time input convention stated above.

C.3 Implementation Details

Training and evaluation setup. We train Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct policies using a Ray-based veRL trainer with FSDP policy sharding and vLLM generation on one node with four NVIDIA GB200 GPUs. Each run is scheduled for 240 policy updates. Table 5 summarizes the training configurations, including settings that differ across environments or model scales. The analyzer learning rate is 1×10^{-5}. The WebShop analyzer uses the final graded task-completion score as its trajectory outcome, whereas the ALFWorld analyzer uses the episode reward; no local progress signal is used. Overlong inputs are truncated from the middle.

Environment and optimization settings. For ALFWorld, each update samples 16 tasks with 2,048/512 prompt/response tokens and at most 50 environment steps. WebShop uses 16 tasks per

Table 5: Training hyperparameters for Pilot. Environment- and model-dependent settings are specified in the value column.

Hyperparameter	Value
Shared (GRPO training)	
Optimizer	AdamW
Policy updates	240
Actor learning rate	1×10^{-6}
GRPO group size G	8
Tasks per batch	16
PPO clip ratio	0.2
Actor KL loss coefficient	0.01
PPO mini-batch size	64
Actor micro-batch size	8 / 4 per GPU (3B / 7B)
ALFWorld prompt / response length	2,048 / 512 tokens
WebShop prompt / response length	4,096 / 512 tokens
Max. environment steps (A/W)	50 / 15
Rollout engine / TP size	vLLM / 1
Rollout / reference log-prob micro-batch	16 / 16 per GPU (3B); 8 / 8 per GPU (7B)
Validation temperature	0.4
Pilot-specific (ours)	
Sources per query J_{src}	4
Latent tokens per source K	8
Frozen trajectory encoder	Initial backbone model
Q-Former depth / attention heads	4 / 8
Maximum source length	2,048 tokens per source
Query / PI-conditioned scoring budget	2,016 / 2,048 tokens
Analyzer learning rate	1×10^{-5}
Anchor loss coefficient	0.05
Influence regularization coefficient	0.01
OPSD loss coefficient	0.5
Teacher micro-batch size	4 (ALFWorld); 2 (WebShop) per GPU
Gate	Detached turn-level $\sigma(5\Delta)$

update, 4,096/512 prompt/response tokens, and at most 15 environment steps. Each task produces $G = 8$ on-policy trajectories. We optimize the actor with AdamW, PPO clipping, and a low-variance reference-policy KL loss. The actor learning rate is 1×10^{-6}, the PPO clip ratio is 0.2, and the KL loss coefficient is 0.01. Validation rollouts use a temperature of 0.4.

Latent PI construction. For each query trajectory, Pilot selects $J_{\mathrm{src}} = 4$ distinct source trajectories from the same task group. A frozen copy of the backbone encodes each source, and a trainable Q-Former produces $K = 8$ latent tokens per source, yielding 32 PI tokens in total. Each serialized source is limited to 2,048 tokens; the query scoring context is limited to 2,016 tokens so that the PI-conditioned input fits within 2,048 tokens. The analyzer uses the trajectory outcome to compute set-LSO credit after excluding all selected sources.

Gating and validation schedule. The current sigmoid-gated Pilot configuration uses the detached turn-level weight $g_t = \mathrm{stopgrad}[\sigma(5\Delta_t)]$, where Δ_t is the mean PI-induced log-probability difference over valid sampled action tokens in turn t. The OPSD loss coefficient is 0.5. We validate every five policy updates and save checkpoints every 50 updates; both operations are performed after each of the final ten updates.

D SUPPLEMENTARY RESULTS

D.1 CROSS-DOMAIN GENERALIZATION

Table 6 reports success rates on the ALFWorld unseen split. With Qwen2.5-3B-Instruct, Pilot achieves 86.7% average success, a gain of 3.9 percentage points over GRPO (82.8%) and slightly below SDAR (88.3%). Its clearest gain over GRPO is on Cool (+21.8). With Qwen2.5-7B-Instruct, Pilot reaches 89.06% average success, gaining 14.84 points over GRPO (74.22%) and 10.93 points over SDAR (78.13%). Its clearest gains over GRPO are on Look (+35.72) and Pick2 (+21.05). The gains across unseen task families suggest that behavioral guidance learned through Pilot remains useful in configurations beyond those encountered during training.

Table 6: **Cross-domain generalization results on ALFWorld Unseen**

	ALFWorld Unseen						
Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.
Qwen2.5-3B-Instruct							
GRPO	**93.1**	**83.3**	87.1	78.9	65.2	**85.0**	82.8
SDAR	**93.1**	**83.3**	**96.8**	**100.0**	78.3	70.0	**88.3**
Pilot	82.8	**83.3**	93.5	84.2	**87.0**	**85.0**	86.7
Qwen2.5-7B-Instruct							
GRPO	**80.00**	57.14	67.65	78.95	94.12	68.42	74.22
SDAR	76.00	78.57	85.29	78.95	94.12	52.63	78.13
Pilot	**80.00**	**92.86**	**88.24**	**89.47**	**100.00**	**89.47**	**89.06**

D.2 COMPUTATIONAL COST

Pilot retains the core training stages of GRPO, including rollout generation, reward computation, old- and reference-policy scoring, advantage estimation, and actor optimization. It additionally performs PI input preparation, PI-conditioned scoring, and Analyzer optimization. We profile these stages on WebShop with Qwen2.5-3B-Instruct, where each update samples 16 tasks with eight rollouts per task. All reported times exclude validation and checkpoint I/O.

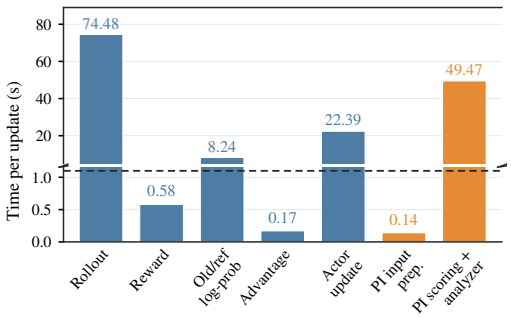

Figure 6: **Pilot per-update time on WebShop (Qwen2.5-3B-Instruct).** Blue denotes steps shared with GRPO; orange denotes Pilot-specific PI steps. The broken axis exposes sub-second phases.

Figure 6 presents the per-update training-time breakdown of Pilot. A complete update takes 156.51 seconds, of which rollout generation accounts for 74.48 seconds. PI input preparation requires only 0.14 seconds, whereas PI-conditioned scoring and Analyzer optimization together require 49.47 seconds. Within this stage, source encoding, PI-conditioned scoring, and backward computation take 5.26, 29.43, and 14.00 seconds, respectively. Analyzer backward computation therefore accounts for 8.9% of the total update time. These additional operations directly reuse the sampled rollouts and require neither additional environment interaction nor autoregressive textual-guidance generation or external model calls.

Table 7 compares the additional operations and observed training costs of GRPO, SDAR, OPID, and Pilot.

GRPO requires 117.27 seconds per update. SDAR requires 136.74 seconds, corresponding to $1.17\times$ the training cost of GRPO, including 12.76 seconds for its skill-conditioned teacher forward pass. OPID requires 151.31 seconds per update, corresponding to $1.29\times$ the training cost of GRPO. Its online teacher-signal generation, waiting, and merging account for 66.42 seconds per update and require 128 external model calls. Pilot requires 156.51 seconds per update, corresponding to $1.33\times$

Table 7: **Additional operations and observed training cost on WebShop with Qwen2.5-3B-Instruct.** Wall-clock times are reported in seconds per update and averaged over all 240 policy updates, excluding validation and checkpoint I/O.

Component	GRPO	SDAR	OPID	Pilot
Additional operations				
Guidance source	–	Offline skills	Online analyses	Learned latent PI
Guidance-conditioned scoring	–	Yes	Yes	Yes
Online textual guidance generation	–	–	Yes	–
Cross-trajectory pairing	–	–	–	Yes
Separate analyzer optimization	–	–	–	Yes
External model calls per update	–	–	128	–
Average wall-clock time per update (s)				
Rollout generation	90.61	92.09	50.87	74.48
Reward computation	–	0.51	1.11	0.58
Skill-conditioned teacher forward	–	12.76	–	–
Teacher-signal wait and merge	–	–	66.42	–
PI input preparation	–	–	–	0.14
PI scoring and analyzer update	–	–	–	49.47
Old/reference log-probability	–	8.35	8.75	8.24
Advantage computation	–	0.16	0.21	0.17
Actor update	26.66	22.60	23.63	22.39
Total training	117.27	136.74	151.31	156.51
Relative cost	1.00×	1.17×	1.29×	1.33×

the training cost of GRPO. Unlike OPID, Pilot learns latent guidance directly from the collected trajectories without online textual-guidance generation or external model calls. Its additional computation is confined to training, and all privileged components are removed after training, leaving the same PI-free inference interface as GRPO.

D.3 Additional Training Dynamics

Validation success rate. Figure 7 presents Pilot's validation success rates on ALFWorld and WebShop with Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct. Validation performance improves in every run, with different learning trajectories across environments and models. On ALFWorld, the 3B model rises from about 0.3 to above 0.8 by mid-training and remains high despite later fluctuations. The 7B model improves steadily and approaches 0.9 late in training. On WebShop, the 3B model follows a steady upward trend and finishes around 0.75–0.8. The 7B model reaches about 0.5 early in training and finishes in a similar range. These trends demonstrate consistent learning progress across both tasks and model scales. The ALFWorld 7B curve comes from a historical outcome-plus-progress configuration and is shown only as a training diagnostic; the other three curves use outcome-only credit with sigmoid gating.

Training reward. Figure 8 presents the corresponding mean rewards on sampled training episodes. Reward increases in every run, although the trajectories vary across tasks and models. On ALFWorld, the 3B model rises from roughly 2 to around 7–8 by mid-training and then fluctuates at a high level. The 7B model progresses to around 9 late in training. On WebShop, both models rise from near zero to approximately 7–8 in the later stages. The consistent increase in training reward, together with the validation gains in Figure 7, shows improvement in both sampled rollouts and validation performance.

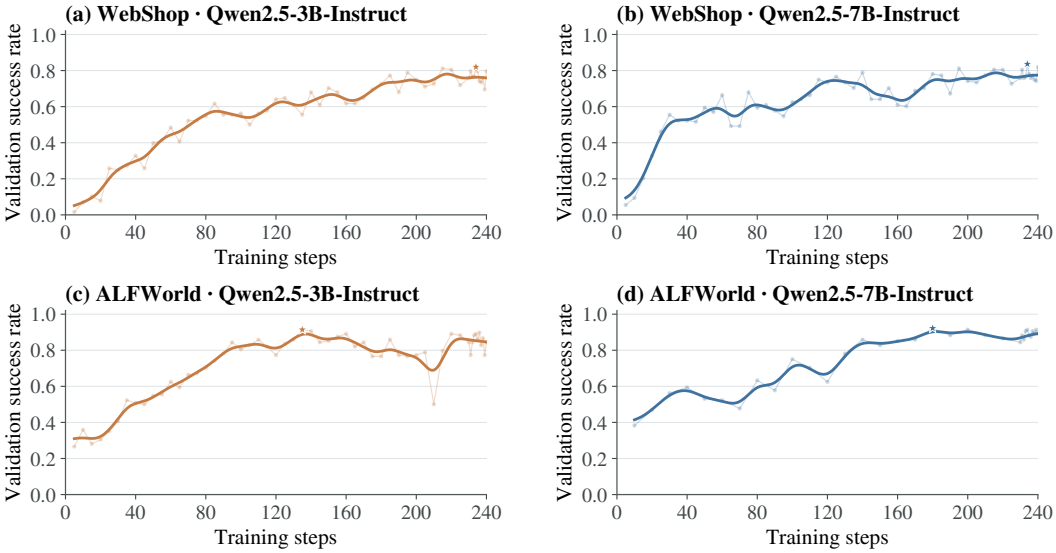

Figure 7: **Validation success during Pilot training.** (a,b) WebShop with Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct; (c,d) ALFWorld with the same respective backbones. Pale curves show logged evaluations, solid curves show smoothed trends, and stars mark the best logged validation points.

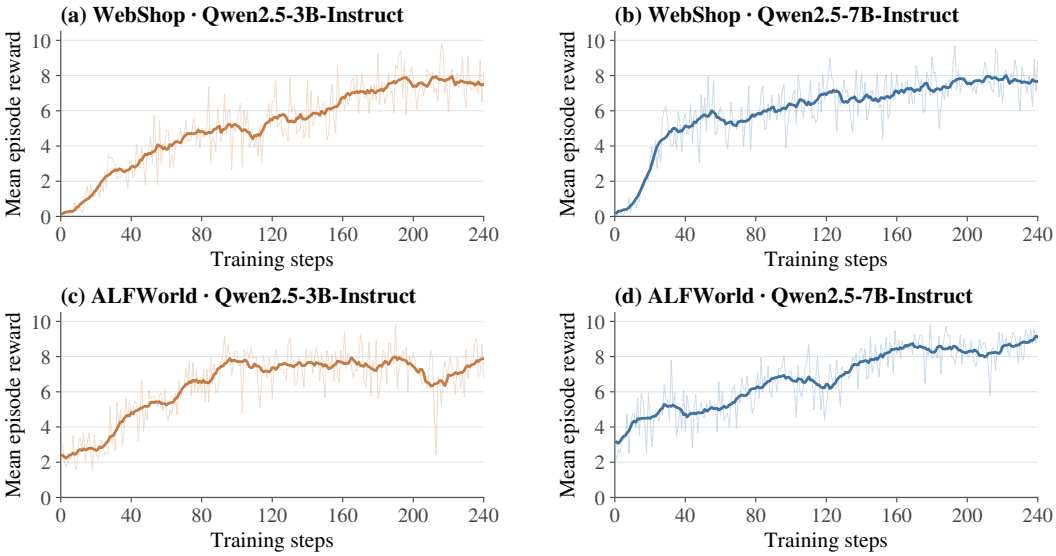

Figure 8: **Mean training-rollout reward during Pilot training.** Panel order matches Figure 7. Pale curves show logged episode rewards, and solid curves show 13-step moving means.

E CASE STUDY

E.1 REPRESENTATIVE SUCCESSFUL TRAJECTORIES

Figures 9–12 show representative successful trajectories generated by PILOT-trained agents. Observations and reasoning are summarized for readability, while task-relevant entities and executed actions are preserved. These examples illustrate agent behavior and are not used as evidence for the aggregate evaluation results reported in the main text.

A Trajectory of an Agent Trained on PILOT - WebShop (Qwen2.5-3B)

Task: Find men's long-sleeve loungewear pajamas with an elastic waistband for daily wear, in Multi 1, size XX-Large, priced below $80.

[Step 1]
Observation: Search page.
Reasoning summary: I should search using the product category, sleeve length, waistband, color, and size constraints.
Action: search[men's sleep lounge long sleeve elastic waistband multi 1 xx-large]

[Step 2]
Observation: Results include B09ND8P2QR, InterestPrint Deer and Stars Men's Loungewear Pajama Sets - Long Sleeve Tee and Jogger Pant, priced $43.59.
Reasoning summary: The result matches the requested category and sleeve length and is within budget, so I will inspect its variants.
Action: click[b09nd8p2qr]

[Step 3]
Observation: The product page lists Multi 1 among the colors and XX-Large among the sizes; the price is $43.59.
Reasoning summary: The required color and size are available. I will first select Multi 1.
Action: click[multi 1]

[Step 4]
Observation: The same product page remains visible after selecting Multi 1, with XX-Large available.
Reasoning summary: The requested color is selected, so I will now select XX-Large.
Action: click[xx-large]

[Step 5]
Observation: InterestPrint men's long-sleeve loungewear pajama set; Multi 1; XX-Large; $43.59; Buy Now is available.
Reasoning summary: The selected variants and listed price satisfy the request, so I can complete the purchase.
Action: click[buy now]

Observations and reasoning are summarized for readability; task-relevant entities and executed actions are preserved.

Figure 9: Successful WebShop trajectory from PILOT (Qwen2.5-3B).

A Trajectory of an Agent Trained on PILOT - WebShop (Qwen2.5-7B)

Task: Find a machine-washable men's long-sleeve T-shirt in army, size 3X-Large Big Tall, priced below $50.

[Step 1]
Observation: Search page.
Reasoning summary: I should include the product type, sleeve length, color, size, and price constraint in the query.
Action: search[men's t-shirts long sleeve color: army size: 3x-large big tall price < 50.00]

[Step 2]
Observation: Results include B00O30JLDK, Legendary Whitetails Men's Non-Typical Long Sleeve T-Shirt, priced $10.52-$40.50.
Reasoning summary: This result is long-sleeved and within budget; I will open it and inspect its options.
Action: click[b00o30jldk]

[Step 3]
Observation: Product page lists army among the colors and 3X-Large Big Tall among the sizes; price is $10.52-$40.50.
Reasoning summary: The required color is available, so I will select army before choosing the size.
Action: click[army]

[Step 4]
Observation: The product page remains visible with army selected and 3X-Large Big Tall available.
Reasoning summary: The requested size is available and the item remains under the price limit.
Action: click[3x-large big tall]

[Step 5]
Observation: Legendary Whitetails Men's Non-Typical Long Sleeve T-Shirt; army; 3X-Large Big Tall; $10.52-$40.50; Buy Now is available.
Reasoning summary: The color, size, sleeve type, and price all satisfy the request, so I can complete the purchase.
Action: click[buy now]

Observations and reasoning are summarized for readability; task-relevant entities and executed actions are preserved.

Figure 10: Successful WebShop trajectory from PILOT (Qwen2.5-7B).

A Trajectory of an Agent Trained on PILOT - ALFWorld (Qwen2.5-3B)

Task: Put a hand towel on a countertop.

[Step 1]
Observation: The room contains a countertop, eight drawers, a hand-towel holder, two sink basins, a toilet, and other bathroom fixtures.
Reasoning summary: The hand-towel holder is the most likely location for a hand towel, so I will inspect it first.
Action: go to handtowelholder 1

[Step 2]
Observation: Hand-towel holder 1 contains hand towel 1.
Reasoning summary: I found the target object and need to pick it up before taking it to the countertop.
Action: take handtowel 1 from handtowelholder 1

[Step 3]
Observation: Hand towel 1 is now in the inventory.
Reasoning summary: The target receptacle is countertop 1, so I will go there next.
Action: go to countertop 1

[Step 4]
Observation: Countertop 1 contains a dish sponge, soap bar, soap bottle, and spray bottle; hand towel 1 remains in the inventory.
Reasoning summary: The target object and receptacle are ready, so I can place the hand towel on the countertop.
Action: move handtowel 1 to countertop 1

Observations and reasoning are summarized for readability; task-relevant entities and executed actions are preserved.

Figure 11: Successful ALFWorld trajectory from PILOT (Qwen2.5-3B).

A Trajectory of an Agent Trained on PILOT - ALFWorld (Qwen2.5-7B)

Task: Cool a tomato and put it in the microwave.

[Step 1]
Observation: The room contains cabinets, two countertops, a fridge, a microwave, and other kitchen fixtures.
Reasoning summary: A tomato is likely in the fridge or a cabinet, so I will inspect the fridge first.
Action: go to fridge 1

[Step 2]
Observation: Fridge 1 is closed.
Reasoning summary: I should open the fridge to check whether it contains a tomato.
Action: open fridge 1

[Step 3]
Observation: Fridge 1 is open and empty.
Reasoning summary: The fridge has no tomato; a countertop is the next plausible location.
Action: go to countertop 1

[Step 4]
Observation: Countertop 1 contains a pepper shaker and a potato, but no tomato.
Reasoning summary: I should inspect the second countertop.
Action: go to countertop 2

[Step 5]
Observation: Countertop 2 contains tomato 1, tomato 2, and other kitchen items.
Reasoning summary: I found suitable tomatoes and will take one for cooling.
Action: take tomato 1 from countertop 2

[Step 6]
Observation: Tomato 1 is now in the inventory.
Reasoning summary: The fridge provides the required cooling operation, so I will return there.
Action: go to fridge 1

[Step 7]
Observation: At the open fridge with tomato 1 in hand.
Reasoning summary: The action to cool tomato 1 is available, so I should apply it now.
Action: cool tomato 1 with fridge 1

[Step 8]
Observation: Tomato 1 has been cooled.
Reasoning summary: The remaining subgoal is to place the cooled tomato in the microwave.
Action: go to microwave 1

[Step 9]
Observation: Microwave 1 is closed.
Reasoning summary: I need to open the microwave before placing the tomato inside.
Action: open microwave 1

[Step 10]
Observation: Microwave 1 is open and empty.
Reasoning summary: The tomato is cooled and the destination is ready, so I can complete the task.
Action: move tomato 1 to microwave 1

Observations and reasoning are summarized for readability; task-relevant entities and executed actions are preserved.

Figure 12: Successful ALFWorld trajectory from PILOT (Qwen2.5-7B).

E.2 Qualitative Comparison with GRPO

Figures 13 and 14 compare representative validation-trajectory excerpts from Qwen2.5-7B agents trained with GRPO and PILOT. Reasoning is summarized for readability, whereas the displayed observations, executed actions, and terminal rewards are preserved. Omitted steps are not reconstructed. These examples illustrate characteristic behavioral differences and are not used as evidence for the aggregate results reported in the main text.

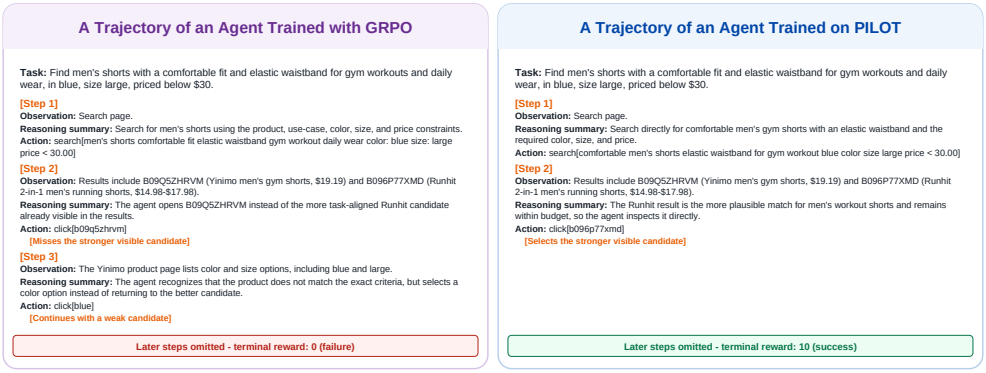

Figure 13: **Qualitative comparison on WebShop with Qwen2.5-7B.** GRPO overlooks a stronger candidate already visible in the search results and continues with a weaker product, whereas PILOT directly selects the more task-aligned candidate. The panels show excerpts from recorded validation trajectories; terminal rewards correspond to the complete episodes.

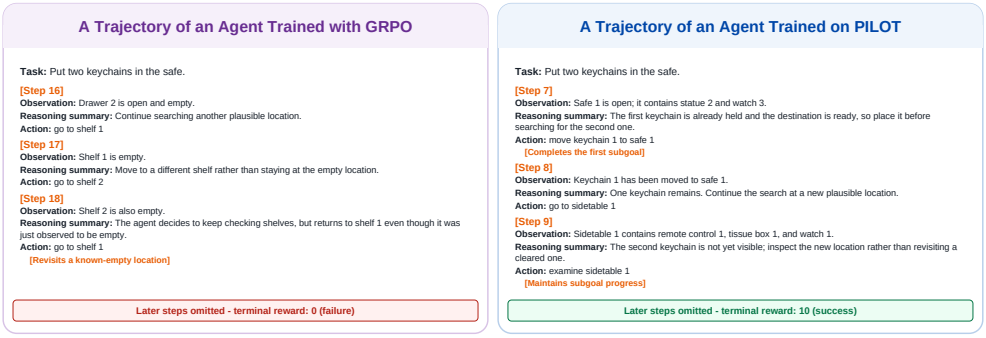

Figure 14: **Qualitative comparison on ALFWorld with Qwen2.5-7B.** GRPO revisits a location that was just observed to be empty, whereas PILOT preserves subgoal progress after placing the first keychain and continues searching for the second. The panels show excerpts from recorded validation trajectories; terminal rewards correspond to the complete episodes.