PIVOTOPD: LEARNING TO RECOVER FROM PIVOTAL MISTAKES IN MULTI-TURN AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) is a promising approach for training language agents, providing dense teacher supervision on student-generated trajectories. However, in multi-turn interaction, an incorrect action changes the states the student encounters later, so errors compound across turns. In our preliminary experiments across three Qwen3 models (8B–235B), we find that more than half of the failed rollouts contain a *pivotal mistake*, an action that moves the agent farther from completing the task, and this mistake typically occurs early. These pivotal mistakes often remain recoverable: guiding the model for only a few turns after the pivotal turn can restore task success. We therefore propose **PIVOTOPD**, an on-policy distillation framework that jointly trains the student to prevent pivotal mistakes and to recover from the states they create. At each pivotal mistake, a teacher model provides a gold action and then names a recovery action at each of the next few turns. ***Preventive distillation*** uses the gold action with reverse KL to steer the student away from the pivotal mistake, while ***recovery distillation*** uses the recovery actions with forward KL to transfer recovery behaviors that the student rarely samples. Against 13 baselines on ALFWorld, WebShop, and Search-based QA, PIVOTOPD achieves the strongest average performance for both Qwen3-1.7B and Qwen3-8B students, improving over the strongest baseline on ALFWorld by $+5.5\%$ with the 1.7B student. The gains also transfer to another model family on the software engineering domain, where PIVOTOPD raises the resolve rate of a Nemotron-3.5 student on SWE-Bench Verified by $+3.2\%$.

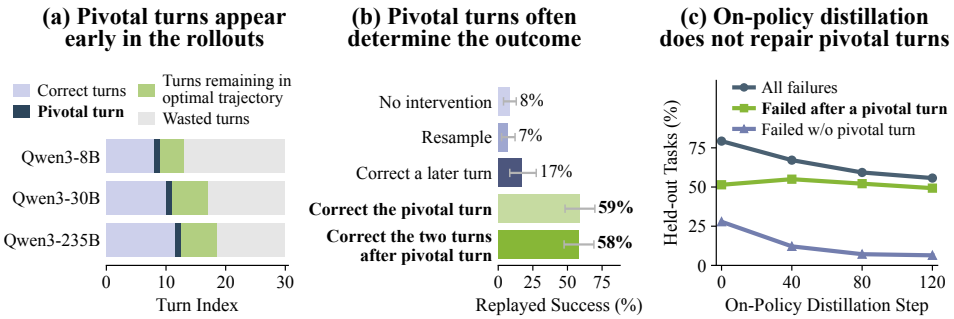


Figure 1: **Failed rollouts trace back to an early pivotal mistake, and standard on-policy distillation does not repair it.** **(a)** Pivotal turns arrive early; the remaining optimal trajectory is short, yet students waste the turns until the end. **(b)** Correcting the pivotal turn or guiding recovery turns both restore success effectively. **(c)** OPD eliminates most incomplete-trajectory failures, but pivotal-turn failures persist across training.

1 INTRODUCTION

Language agents are increasingly capable of solving complex tasks through multi-turn interactions [45, 15, 52, 41, 46, 29, 44, 7]. To improve their performance, recent work has adopted on-policy distillation (OPD) as an effective training paradigm [50, 53], since it provides dense, token-level teacher supervision on trajectories that the student generates itself [1, 4, 16]. However, OPD is more challenging in multi-turn settings because each action changes the external environment, which then returns new observations and determines which actions are available in later turns [25, 34]. A single

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

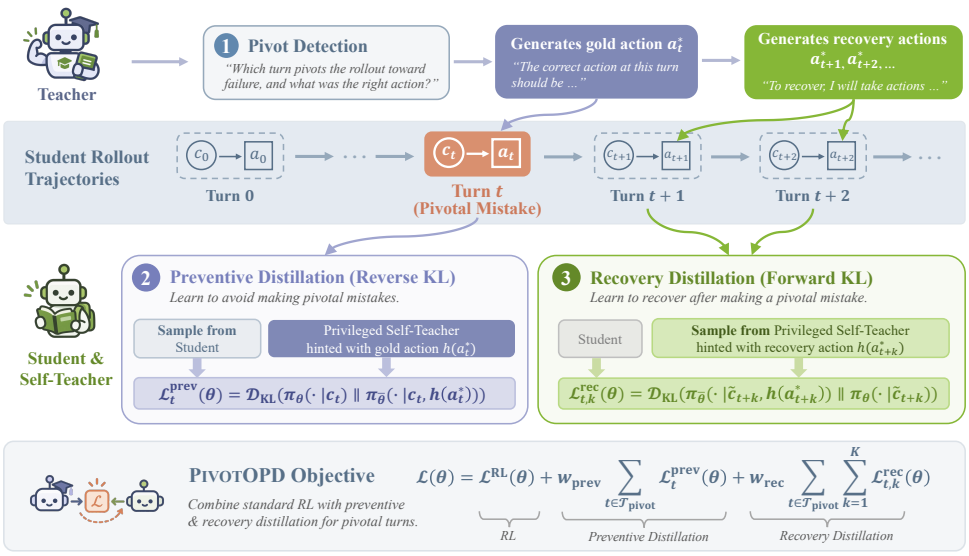


Figure 2: **Overview of PivotOPD. (1) Pivot detection.** A teacher model reads each rollout with its outcome, selects candidate turns, and names a gold action at each. A candidate turn is pivotal when the student's committed action differs from the gold action. After each pivotal turn, the teacher names a recovery action at each of the next few turns. **(2) Preventive distillation** trains the student to avoid the pivotal mistake. The frozen student hinted with the gold action serves as a privileged self-teacher that re-scores the student's recorded response, and a reverse KL loss moves the student toward the gold action. **(3) Recovery distillation** trains the student to recover after the mistake. The self-teacher is instead hinted with the recovery action and writes a recovery response, on which the student is trained without the hint through a forward KL loss. Later recovery turns start from the state reached by executing its action in a copy of the environment that replays all preceding actions. Both terms are combined with group-based RL in a single PPO update.

incorrect action can lead to *error accumulation*, in which the student ends up in states created by its own mistake and diverges further from the teacher over the turns that follow [24, 25, 34, 51]. Existing multi-turn OPD methods address error accumulation by reweighting turns or restricting which turns receive the distillation signal [53, 51, 34, 49], but leave open whether such failures hinge on a single decisive mistake and whether the student can still recover from it.

To answer these questions, we analyze failed rollouts in ALFWorld [29], where the agent completes household goals, such as heating an egg, through textual actions (Appendix D.5), and summarize the results in Figure 1. Its symbolic oracle reads the full environment state and computes the remaining optimal trajectory, i.e., the shortest action sequence that completes the task, at every turn. Across three Qwen3 models (8B–235B) [39], more than half of the failed rollouts contain a *pivotal mistake*: an action that lengthens the remaining optimal trajectory or makes the task unsolvable. We call the turn where it is committed a *pivotal turn*. The first one typically arrives early, and all three models waste most of the remaining turns without recovering (Figure 1a). In counterfactual replays of Qwen3-8B's failures, correcting the pivotal turn with the oracle action raises the replayed success from 8% to 59% (Figure 1b). To test whether a failure can be repaired after the mistake, we keep the mistake and apply the oracle action at the next two turns, which still reaches 58%. Therefore, pivotal mistakes are largely recoverable.

Existing training methods rarely teach this recovery, because they learn mostly from the student's own trajectories, and these rarely contain a recovery. Outcome-based RL rewards a recovery only when a rollout happens to find one, and its group-relative advantage is zero whenever every rollout of a task fails [27]. Methods that assign credit to individual turns or focus on the turns that decide the outcome share this limitation [3, 47, 42], and so does OPD. In our analysis, standard OPD lowers the failure rate from 79% to 56% of held-out tasks, while failures after a pivotal turn only fall from 51% to 49% (Figure 1c). Although OPD reduces the probability of the committed mistake, the oracle action still has less than 1% probability at every evaluated pivotal turn (Appendix A). The student therefore lacks a direct learning signal at the states its own mistakes create.

To provide this signal, we introduce **PivotOPD**, which augments group-based RL with dense token-level supervision concentrated at pivotal turns and the turns that follow them (Figure 2). Since

most environments offer no oracle, PIVOTOPD estimates pivotal turns through *pivot detection*: a teacher model, a larger LLM in our main setting, reads each rollout in hindsight, selects a few candidate turns where the student may have gone wrong, and names a gold action at each. We treat a candidate turn as pivotal when the student's committed action differs from the gold action. On ALFWorld, at least one detected pivotal turn falls within one turn of the oracle-labeled one in 77.8% of failed rollouts on average, at least twice the rate of randomly chosen turns (Table 3).

PIVOTOPD obtains its token-level targets from a *privileged self-teacher*, the frozen student conditioned on a hint that names the teacher model's action, so the teacher only names actions [50, 20]. At each pivotal turn, ***preventive distillation*** re-scores the student's recorded response with the self-teacher hinted with the gold action and minimizes a reverse KL loss, which shifts the student toward the gold action and away from the committed mistake. After each pivotal turn, ***recovery distillation*** has the teacher model name a recovery action at each of the next few turns and trains the student, without the hint, on the responses that the hinted self-teacher writes toward them. Its mass-covering forward KL loss raises the probability of recovery actions that the student rarely samples [11].

Our contributions are as follows:

- **Diagnosis.** Using ALFWorld's oracle, we show that more than half of the failed rollouts contain a recoverable pivotal mistake, which standard OPD does not repair (Section 2).
- **Method.** We propose PIVOTOPD, which uses a privileged self-teacher to train the student both to avoid pivotal mistakes and to recover from them (Section 3).
- **Results.** Against 13 baselines, PIVOTOPD achieves the best average performance on ALFWorld, WebShop, and Search-based QA for both Qwen3 students (Table 1), and at 8B it recovers from pivotal mistakes more than three times as often as standard OPD (Figure 5b). With a Nemotron-3.5 student on SWE-Bench Verified, it also raises the resolve rate by $+3.2\%$, versus $+0.2\%$ for standard OPD (Figure 4).

2 MOTIVATING ANALYSIS: AGENTS OFTEN FAIL AT A PIVOTAL TURN AND RARELY RECOVER FROM IT

We ask whether failures hinge on a single decisive mistake, whether that mistake remains recoverable, and whether standard OPD repairs it (Figure 1). On ALFWorld [29], a symbolic oracle computes the remaining optimal trajectory from the full environment state, and we call its next action the *oracle action*. We roll out Qwen3-8B, Qwen3-30B-A3B, and Qwen3-235B-A22B [39] on 140 held-out tasks, replay each with this oracle, and analyze Qwen3-8B below (Appendix A).

A single pivotal turn often determines the outcome, yet it is recoverable. Across the three models, 59% of the failed trajectories contain an action that lengthens the remaining optimal trajectory or makes the task unsolvable (Appendix A). We call such an action a ***pivotal mistake***, and the turn where it is committed a ***pivotal turn*** (formalized in Section 3.1). In failed trajectories with a pivotal turn, the first one typically arrives early, at a median of turn 8–12 out of 30 (Figure 1a). All three models then waste an average of 18–21 more turns without recovering from it.

Among the 72 failed Qwen3-8B trajectories with a pivotal turn, correcting the first pivotal turn with the oracle action raises the replayed success from 8% to 59%, whereas correcting a later turn helps far less (Figure 1b). To test whether the post-mistake state is still recoverable by some admissible action sequence, we leave the mistake in place and force the oracle action at the next two turns, which still reaches 58%. Learning to recover after a pivotal mistake can therefore be nearly as effective as preventing it.

Standard on-policy distillation does not repair pivotal turns. We train Qwen3-8B for 120 steps with standard OPD, in which a privileged copy of the model provides token-level supervision on the model's own responses [50]. At each checkpoint, we use the same oracle to categorize the remaining failures (Figure 1c). On held-out tasks, OPD lowers the overall failure rate by 23.6% but the rate of failures after a pivotal turn by only 2.1%. Most of the gain comes from failures without a pivotal turn, which fall by 21.4%. OPD does suppress the committed mistake, moving its median probability from 0.999 to below 10^{-5}, yet the oracle action stays below 10^{-2} at every pivotal turn (Appendix A). A group of eight rollouts is therefore not expected to sample it even once, so it receives little direct reinforcement.

3 PIVOTOPD: PIVOT-AWARE ON-POLICY DISTILLATION

Section 2 shows that more than half of the failed rollouts contain a recoverable pivotal mistake that standard OPD does not repair. PIVOTOPD therefore concentrates the dense token-level signal of on-policy distillation at pivotal turns and the turns that follow them. It adds three components to group-based RL (Figure 2): *pivot detection* (Section 3.2), where a teacher model names the actions that the student should take at and after pivotal turns, and *preventive* and *recovery distillation* (Section 3.3), where a privileged self-teacher provides the dense token-level targets. Section 3.4 analyzes the learning signal of recovery distillation, and Algorithm 1 in Appendix B summarizes one training step.

3.1 PRELIMINARIES

We model an agentic task as a partially observable Markov decision process [9]. At turn t, the environment is in a latent state s_t and emits an observation o_t, where o_0 states the task instruction. The agent holds a context $c_t = (o_0, y_0, o_1, y_1, \ldots, o_t)$, which is the history of observations and responses up to the current observation. The current observation specifies the admissible actions $\mathcal{A}(c_t)$, which ALFWorld and WebShop list explicitly, Search-based QA defines as any search query or answer, and SWE-Bench defines as any call to the agent's tools or the final patch submission. The student policy generates the next response $y_t \sim \pi_\theta(\cdot \mid c_t)$, which includes free-form reasoning followed by a committed action $a_t := \mathrm{act}(y_t) \in \mathcal{A}(c_t)$. A trajectory $\tau = \{(o_t, y_t)\}_{t=0}^{T-1}$ records one attempt of T turns and receives an outcome score $R(\tau)$ when it ends.

Following group-based RL for agents [27, 3], we roll out each task G times from the same initial state. The outcome scores of these rollouts yield a group-relative advantage A^{RL}, which we optimize with the clipped PPO loss $\mathcal{L}^{\mathrm{RL}}$ [26]. We write $\pi_{\bar{\theta}}$ for the student policy frozen at the start of each training step. A teacher model M assists training, and in the self-distillation setting, M is the student itself.

Pivotal mistake & pivotal turn. We now formalize the pivotal mistakes of Section 2. We write $L(s_t)$ for the length of the remaining optimal trajectory, which is the minimum number of turns needed to complete the task from the latent state s_t, with $L(s_t) = \infty$ once the task is unsolvable. A turn t is a *pivotal turn* if its committed action a_t increases this length, i.e., $L(s_{t+1}) > L(s_t)$, and a_t is then a *pivotal mistake*. The pivotal turns of a trajectory τ form the set $\{t : L(s_{t+1}) > L(s_t)\}$, which can contain several turns even when τ succeeds. For each failed trajectory, Section 2 uses the smallest element of this set. Pivot detection estimates these turns without access to L.

3.2 PIVOT DETECTION WITH A TEACHER MODEL

Computing L requires an oracle that knows the optimal trajectory from every state (Section 3.1), such as the symbolic oracle of ALFWorld that Section 2 uses. Since most environments offer no such oracle, PIVOTOPD instead detects pivotal turns during training with the teacher model (Figure 2).

Pivotal turns and gold actions. After collecting the rollouts of a training step, we prompt the teacher model to read each trajectory τ together with its outcome $R(\tau)$ and to select up to m *candidate turns* where the student may have gone wrong. At each candidate turn t, the teacher model also names a ***gold action*** $a_t^* \in \mathcal{A}(c_t)$, which serves as its estimate of the oracle action. A candidate turn does not necessarily contain a mistake, since the student's committed action a_t may already agree with the gold action. We therefore treat a candidate turn as pivotal only when the two disagree, i.e., $a_t \neq a_t^*$. The teacher-detected pivotal turns of a training step form the set $\mathcal{T}_{\text{pivot}}$. At least one teacher-detected pivotal turn falls within one turn of the oracle-labeled pivotal turn in 77.8% of failed ALFWorld trajectories on average over our two teachers, at least twice the rate of randomly chosen turns (Table 3). Appendix B compares pivot detection with this oracle labeling.

Recovery actions. A gold action shows how to avoid a pivotal mistake, but the student also needs to learn how to continue from the state that the mistake creates. After each pivotal turn $t \in \mathcal{T}_{\text{pivot}}$, the teacher model names ***recovery actions*** for up to K *recovery turns*, where K is the recovery budget. We write $\tilde{c}_{t+k}$ for the context of the k-th recovery turn, $k = 1, \ldots, K$. The first recovery turn starts from the post-mistake state, whose context is the recorded $\tilde{c}_{t+1} = c_{t+1}$, and Section 3.3 describes how later recovery turns are reached. At each recovery turn, we query the teacher model again with $\tilde{c}_{t+k}$, and it names a recovery action $a_{t+k}^* \in \mathcal{A}(\tilde{c}_{t+k})$ as the next action toward completing the task.

3.3 PREVENTIVE AND RECOVERY DISTILLATION

Privileged self-teacher. To turn a gold or recovery action into a token-level target, we provide it to the frozen student $\pi_{\bar{\theta}}$ as a *hint*. For an action a, the hint $h(a)$ is a short instruction that presents a as a reasonable action at the current turn and asks the model to reason toward it in its own words. Conditioning $\pi_{\bar{\theta}}$ on this hint gives a *privileged self-teacher* $\pi_{\bar{\theta}}(\cdot \mid c, h(a))$, which differs from the student only through the information that the action provides [50, 20]. This keeps the distillation target in the student's own reasoning style.

Preventive distillation. At each pivotal turn $t \in \mathcal{T}_{\text{pivot}}$, preventive distillation trains the student toward the self-teacher hinted with the gold action through the reverse KL loss

$$\mathcal{L}_t^{\text{prev}}(\theta) = D_{\text{KL}}(\pi_\theta(\cdot \mid c_t) \,\|\, \pi_{\bar{\theta}}(\cdot \mid c_t,\, h(a_t^*)))\,. \tag{1}$$

This loss takes its expectation under the student, so we evaluate it on the student's recorded response y_t. Minimizing it shifts the student toward the gold action and away from the committed mistake.

Recovery distillation. A reverse KL loss like Eq. 1 can only reweight responses that the student has already produced, while recovery distillation trains the student on responses that the self-teacher writes. At the k-th recovery turn after a pivotal turn t, we sample a recovery response $y_{\text{rec}} \sim \pi_{\bar{\theta}}(\cdot \mid \tilde{c}_{t+k}, h(a^*_{t+k}))$ from the self-teacher hinted with the recovery action. We keep the response only if it commits to an action and does not refer to the hint (Appendix B). For $k < K$, we then execute its action $\text{act}(y_{\text{rec}})$ in a copy of the environment that replays all preceding actions, which reaches the next recovery context $\tilde{c}_{t+k+1}$.

At each recovery turn, recovery distillation then trains the student toward this self-teacher through the forward KL loss

$$\mathcal{L}_{t,k}^{\text{rec}}(\theta) = D_{\text{KL}}\big(\pi_{\bar{\theta}}\big(\cdot \mid \tilde{c}_{t+k},\, h(a^*_{t+k})\big) \,\big\|\, \pi_\theta(\cdot \mid \tilde{c}_{t+k})\big)\,, \tag{2}$$

in which the student sees $\tilde{c}_{t+k}$ without the hint. This loss takes its expectation under the self-teacher, so we evaluate it on the accepted responses y_{rec}. Because the forward KL is mass-covering, minimizing it raises the student's probability of the recovery action at states that arise from its own mistakes (Section 3.4). We select K on validation and study its effect in Section 5.

PIVOTOPD training objective. We combine group-based RL with preventive and recovery distillation in a single training objective,

$$\mathcal{L}(\theta) \;=\; \mathcal{L}^{\text{RL}}(\theta) \;+\; w_{\text{prev}} \sum_{t \in \mathcal{T}_{\text{pivot}}} \mathcal{L}_t^{\text{prev}}(\theta) \;+\; w_{\text{rec}} \sum_{t \in \mathcal{T}_{\text{pivot}}} \sum_{k=1}^{K} \mathcal{L}_{t,k}^{\text{rec}}(\theta), \tag{3}$$

where the sums cover the pivotal turns of the current training step and the recovery turns after each. We implement all three terms in a single PPO update over the rollout and recovery responses. The teacher model also writes brief feedback on each trajectory as a whole, which we distill at every turn with a small weight (Appendix B).

To implement both distillation losses within the PPO update, we assign the ℓ-th token of a response y at a context c with a named action a the *distillation advantage*

$$A_\ell^{\text{distill}} = \log \pi_{\bar{\theta}}(y_\ell \mid c,\, h(a),\, y_{<\ell}) - \log \pi_{\bar{\theta}}(y_\ell \mid c,\, y_{<\ell})\,, \tag{4}$$

which measures how much the hint $h(a)$ changes the frozen student's log-probability of this token. Preventive distillation adds $w_{\text{prev}}\, A_\ell^{\text{distill}}$ to A^{RL} on each token of y_t, with $c = c_t$ and $a = a_t^*$, which implements a per-token form of Eq. 1. Recovery distillation uses a clipped $w_{\text{rec}}\, A_\ell^{\text{distill}}$ as the only advantage on each token of y_{rec}, with $c = \tilde{c}_{t+k}$ and $a = a^*_{t+k}$ (Appendix B). This update is not the gradient of Eq. 2, but without clipping it vanishes only at its minimizer (Lemma 2).

3.4 THEORETICAL ANALYSIS: LEARNING SIGNAL AFTER A PIVOTAL MISTAKE

Group-based RL and reverse KL losses like Eq. 1 train on responses that the student produces, whereas recovery distillation trains on responses that the self-teacher writes. We analyze how this difference affects the learning signal on the recovery action, treating the committed action at a recovery turn as one categorical decision (proofs in Appendix C).

Method	ALFWorld							Search-based QA								WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	**Avg.**	NQ	Triv	Pop	Hotp	2Wk	MuS	Bam	**Avg.**	Score	**Succ.**
Qwen3-1.7B																	
Base Model	6.8	60.2	0.0	0.0	3.6	4.1	12.4	16.8	50.8	37.9	26.9	21.0	10.0	15.6	25.6	45.0	4.6
OPSD	23.7	31.2	10.9	0.0	2.2	6.5	12.4	43.4	57.9	48.2	33.3	34.0	10.0	30.5	36.8	49.2	10.1
GRPO	74.0	44.1	33.3	41.9	30.4	32.5	42.7	42.4	58.3	50.2	38.2	38.5	9.1	27.4	37.7	74.5	57.0
Skill-GRPO	**89.8**	62.4	73.0	50.4	68.8	43.9	64.7	43.0	57.3	44.3	26.2	34.3	10.0	20.2	33.6	69.0	53.9
AgentOPSD	72.3	35.5	59.8	56.4	58.7	30.1	52.1	45.6	57.0	41.7	33.7	32.7	8.7	24.0	34.8	69.0	46.4
Skill-SD	79.1	58.1	54.0	57.3	50.0	56.1	59.1	39.2	53.1	47.2	32.4	34.0	12.0	28.3	35.2	77.2	58.6
PivotRL	45.8	75.3	26.4	15.4	18.8	19.5	33.5	35.6	45.3	**63.8**	31.4	29.8	**21.4**	21.5	35.5	66.6	28.8
TurnOPD	63.3	75.3	57.5	50.4	30.4	41.5	53.1	22.0	43.0	43.0	**40.5**	47.2	16.5	34.9	35.3	70.6	55.5
StepOPSD	72.3	60.2	73.0	21.4	42.0	48.0	52.8	42.1	59.5	48.5	**40.5**	35.0	9.1	28.0	37.5	72.8	60.9
TCOD	66.7	75.3	60.9	36.8	22.5	49.6	51.9	24.9	44.7	47.2	35.0	46.0	17.8	29.0	34.9	77.2	46.1
SOD	70.1	**83.9**	55.2	42.7	39.9	**65.0**	59.5	24.9	52.4	51.5	39.8	47.2	15.5	36.1	38.2	66.6	54.7
RLSD	72.3	67.7	73.0	50.4	61.6	39.0	60.7	42.1	57.3	51.5	38.2	40.1	12.0	29.0	38.6	83.2	62.5
SDAR	83.1	55.9	92.5	71.8	58.0	48.0	68.2	42.4	57.3	46.3	36.9	42.4	8.1	26.5	37.1	64.6	57.0
OPID	83.1	57.0	65.5	71.8	46.4	43.9	61.3	44.0	57.9	49.2	35.9	37.2	11.0	27.1	37.5	76.6	68.0
PivotOPD	87.6	55.9	**93.7**	**72.6**	**73.9**	58.5	**73.7**	**48.5**	**64.1**	57.0	39.2	**49.2**	15.9	**37.4**	**44.5**	**84.4**	**76.6**
Qwen3-8B																	
Base Model	31.1	39.8	23.0	0.0	8.0	26.0	21.3	13.6	37.5	13.6	17.8	25.2	3.9	17.4	18.4	27.5	7.0
OPSD	40.1	49.5	63.2	28.2	46.4	52.8	46.7	47.2	67.6	**57.3**	36.2	42.1	11.0	41.7	43.3	47.2	18.2
GRPO	44.6	59.1	11.5	21.4	31.2	43.9	35.3	35.0	63.1	44.3	32.0	41.1	9.1	37.7	37.5	83.5	75.0
Skill-GRPO	86.4	49.5	73.6	77.8	13.0	89.4	65.0	36.2	54.0	36.2	23.3	38.2	8.1	38.6	33.5	78.0	74.2
AgentOPSD	94.9	76.3	85.1	52.1	67.4	67.5	73.9	48.9	62.5	40.1	29.4	44.0	12.9	35.2	39.0	81.4	65.6
Skill-SD	89.8	91.4	**100.0**	88.9	76.8	84.6	88.6	43.0	65.7	43.0	32.0	46.0	11.0	40.2	40.1	81.7	72.7
PivotRL	85.3	74.2	42.0	29.9	30.4	61.8	53.9	28.5	52.4	48.2	36.9	39.5	17.2	43.9	38.1	65.9	53.0
TurnOPD	85.3	82.8	50.0	51.3	42.0	57.7	61.5	25.6	56.6	45.3	38.2	45.3	15.9	39.6	38.1	84.2	58.6
StepOPSD	93.2	82.8	94.8	55.6	60.1	89.4	79.3	42.1	67.6	55.3	33.3	46.0	11.0	44.2	42.8	80.7	72.7
TCOD	85.3	82.8	54.0	37.6	42.0	53.7	59.2	18.4	63.8	45.3	**41.1**	48.2	18.4	46.7	40.3	79.0	55.5
SOD	78.0	93.5	57.5	44.4	61.6	61.8	66.1	22.7	63.8	42.4	36.9	46.6	18.4	38.3	38.4	83.9	43.0
RLSD	**100.0**	**100.0**	84.5	**94.0**	76.8	89.4	90.8	36.2	58.3	42.1	23.0	34.0	7.1	38.3	34.1	84.6	78.9
SDAR	96.6	66.7	84.5	71.8	**83.3**	**100.0**	83.8	45.0	68.0	44.3	34.0	50.5	11.0	39.3	41.7	85.5	79.7
OPID	**100.0**	66.7	94.8	71.8	79.7	89.4	83.7	**52.4**	65.4	50.2	40.1	**51.1**	15.9	44.2	45.6	81.1	73.4
PivotOPD	**100.0**	**100.0**	**100.0**	85.5	76.8	95.9	**93.0**	50.2	**72.5**	52.1	39.2	48.9	**19.1**	**49.8**	**47.4**	**88.2**	**81.9**

Table 1: **Results on ALFWorld, Search-based QA, and WebShop with Qwen3-1.7B and Qwen3-8B students.** Results are averaged over three seeds, Avg. columns are unweighted means over task types or datasets, and every method with a teacher uses Qwen3-30B-A3B (1.7B) or Qwen3.5-122B-A10B (8B). QA columns abbreviate Natural Questions, TriviaQA, PopQA, HotpotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle. We highlight the **best** and second-best results.

Proposition 1 (Learning signal after a pivotal mistake)**.** *At a recovery turn, let p and q be the action distributions of the frozen student and its privileged self-teacher, and let $g(a^*)$ be the expected update on the student's logit of the recovery action a^*. Then:* ***(i)*** *at the frozen student, the recovery loss satisfies $\mathcal{L}^{\mathrm{rec}}_{t,k} \geq q(a^*)\log\big(1/p(a^*)\big) - \log 2$;* ***(ii)*** *if actions are sampled from p and weighted by any per-action signal w, as in group-based RL or a reverse KL loss like Eq. 1, then $g(a^*) = p(a^*)\big(w(a^*) - \mathbb{E}_p[w]\big)$;* ***(iii)*** *if actions are sampled from q and weighted by the distillation advantage clipped at a bound δ, as in recovery distillation, and the hint raises the probability of a^* by a factor of at least e^{δ}, then $g(a^*) \geq \delta\big(q(a^*) - p(a^*)\big) > 0$.*

When the student assigns little probability to the recovery action, its divergence from the self-teacher in (i) is large, yet the student-sampled updates in (ii) become too small to reduce it. Recovery distillation instead provides the update in (iii), whose size depends on the self-teacher rather than the student, as we confirm empirically in Section 5 (Figure 6).

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate PivotOPD on four multi-turn agentic benchmarks. **(1) ALFWorld** [29] is an embodied household environment with six task types, where the agent completes language-

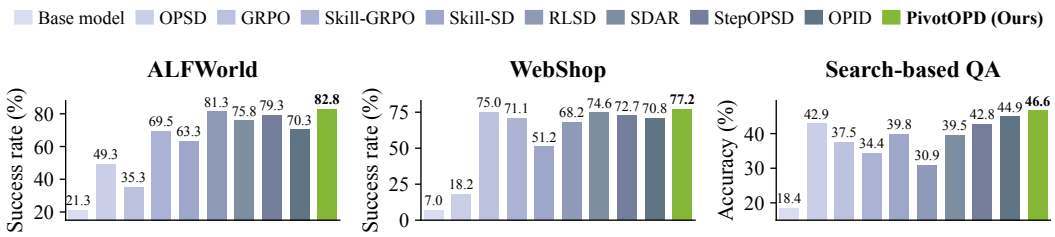


Figure 3: **Self-distillation results on Qwen3-8B**, where the student serves as its own teacher. PIVOTOPD is best on all three benchmarks, outperforming the strongest baseline by $+3.9\%$ on average.

specified goals through textual actions. **(2) WebShop** [44] is a simulated e-commerce website where the agent searches for and purchases a product that satisfies an instruction. **(3) Search-based QA** [7] requires the agent to answer questions from seven open-domain QA datasets [13, 8, 18, 43, 5, 31, 22] by calling a search engine. **(4) SWE-Bench Verified** [6] requires the agent to resolve real GitHub issues by exploring and editing a Python repository, and we use it to test transfer to software engineering and to a different model family (Section 4.2). Appendix D.5 shows an example episode from each of the first three benchmarks.

Models & training configurations. We use Qwen3-1.7B and Qwen3-8B [39] as students, paired with a Qwen3-30B-A3B teacher and a Qwen3.5-122B-A10B teacher, respectively. Every method that uses a teacher, including the baselines, uses the same teacher for a given student. In the self-distillation setting, the student serves as its own teacher. On the first three benchmarks, all methods share the same training data, 160 training steps, and 8 rollouts per task (Appendices D.1 and D.4).

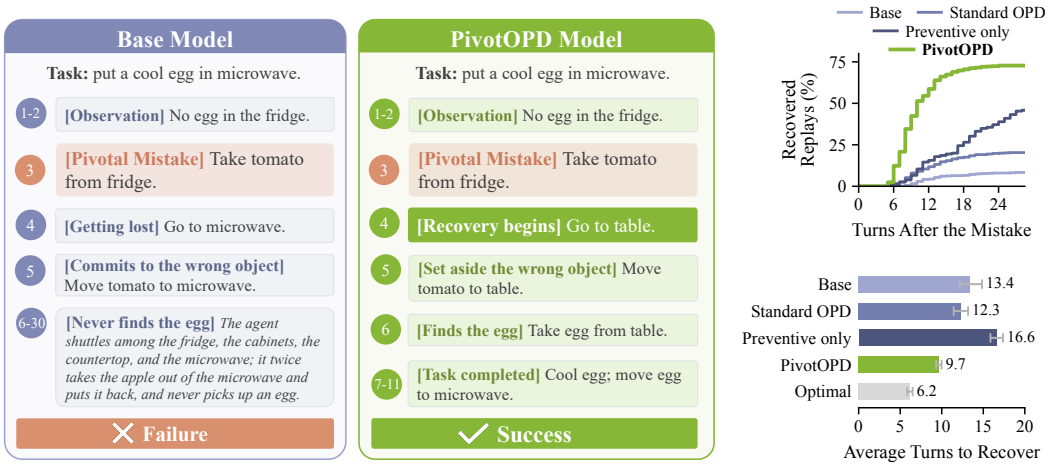
Baselines. Besides the base model, we compare PIVOTOPD against three groups of methods, namely **(1)** RL and self-distillation, with GRPO [27], OPSD [50], RLSD [40], and SDAR [17]; **(2)** turn-level distillation for multi-turn agents, with TurnOPD [53], TCOD [34], SOD [51], StepOPSD [49], and AgentOPSD [38]; and **(3)** high-level guidance through skills or pivotal turns, with Skill-GRPO and OPID [42], Skill-SD [33], and PivotRL [47] (Appendix D.3).

Evaluation. We report the task success rate on each ALFWorld task type, the exact-match accuracy on each QA dataset, and both the normalized score and the success rate on WebShop. The ALFWorld and Search-based QA averages are unweighted means over task types and datasets, respectively. Each experiment is run with three random seeds, and we report the average. We select each checkpoint on a separate validation set and evaluate it on held-out test sets of 274 ALFWorld tasks, 500 WebShop instructions, and 725 QA questions, sampling at temperature 0.4 (Appendix D.2).

4.2 MAIN RESULTS

PIVOTOPD attains the best average performance on all three main benchmarks, especially where successful rollouts are scarce. As Table 1 shows, PIVOTOPD ranks first on all eight per-benchmark averages. With the 1.7B student, it improves over the strongest baseline by $+5.5\%$ on ALFWorld and $+5.9\%$ on Search-based QA. With the 8B student, the margins are smaller but remain at least $+1.8\%$. The comparison with GRPO, which learns from outcome rewards alone, shows that PIVOTOPD helps most where successful rollouts are scarce. With the 1.7B student, it improves over GRPO the most on Clean, Cool, and Heat, which are the three ALFWorld task types that the base model solves least often. When every rollout of a task fails, group-relative advantages provide no learning signal (Proposition 2 in Appendix C), whereas recovery distillation still trains the recovery action at the states where the student gets stuck (Section 3.4).

PIVOTOPD turns partial progress into successful task completion. On WebShop, the normalized score gives partial credit for partially satisfying the instruction, whereas the success rate counts only fully completed tasks. With the 1.7B student, RLSD attains the highest score among the baselines. PIVOTOPD improves over RLSD by only $+1.2\%$ in score but by $+14.1\%$ in success rate. With both students, its gains over GRPO are likewise larger in success rate than in score. The larger gains in success rate are consistent with the effect of recovery distillation, which teaches the student to recover from pivotal mistakes that would otherwise leave a partially completed task unfinished. Indeed, removing recovery distillation ($K = 0$) substantially lowers the best WebShop validation score of the 1.7B student (Figure 7).



(a) Case study: continuations after the same pivotal mistake.

(b) Recovery rate and efficiency.

Figure 5: **PivotOPD learns to recover from pivotal mistakes.** Each policy continues from a replayed prefix that ends with the pivotal mistake (Appendix E.2). **(a)** After the same pivotal mistake, the base model fails, whereas PivotOPD recovers and completes the task. **(b) Top:** the percentage of replays that recover within a given number of turns. **Bottom:** the average number of turns to recover among recovered replays, with the optimal number of turns in gray. PivotOPD recovers the most often and in the fewest turns.

PivotOPD remains effective without a stronger external teacher. In its main setting, PivotOPD uses a stronger teacher to name the gold and recovery actions. To test whether it still works without such a teacher, we use the student as its own teacher for PivotOPD and for every baseline that uses a teacher. As shown in Figure 3, PivotOPD achieves the best performance on all three benchmarks, outperforming the strongest baseline on each by at least $+1.5\%$. Relative to training under the stronger teacher (Table 1), it stays within 5% on Search-based QA and in WebShop success rate but falls 10.2% short on ALFWorld. These results suggest that much of the gain comes from where and how the teacher intervenes rather than from teacher capacity alone.

PivotOPD also improves a different model family on software engineering. The experiments so far all use Qwen3 students, so we next ask whether the gains of PivotOPD carry over to another model family and to software engineering. To this end, we train Nemotron-3.5-SFT with Nemotron-3-Super as the teacher, both from the Nemotron model family [19], and evaluate its resolve rate on SWE-Bench Verified (Appendix D.6). Since SWE-Bench episodes are long and containerized, we adapt pivot detection and the recovery budget to them (Appendix D.6). As shown in Figure 4, PivotOPD raises the resolve rate by $+3.2\%$ and closes roughly a third of the gap to the teacher, whereas standard OPD improves it by only $+0.2\%$.

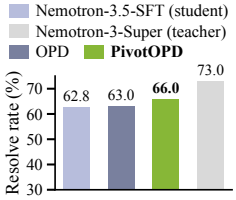


Figure 4: **SWE-Bench Verified results.** PivotOPD improves more than OPD.

5 Understanding Recovery from Pivotal Mistakes

We now study the recovery behavior behind these gains, asking **(1)** whether PivotOPD learns to recover from pivotal mistakes, **(2)** what makes this recovery learnable, **(3)** why recovery distillation provides a learning signal, and **(4)** how many recovery turns to train.

PivotOPD learns to effectively recover from pivotal mistakes. We replay the 72 ALFWorld pivotal mistakes of the motivating analysis (Section 2) and let each trained policy continue after the pivotal mistake (Appendix E.2). In the example of Figure 5a, after the student takes a tomato instead of the requested egg, the base model commits the tomato to the microwave and never finds the egg, whereas PivotOPD sets the tomato aside, takes the egg, and completes the task. Across all 72 pivotal mistakes, PivotOPD recovers roughly nine times as often as the base model and in the fewest turns (Figure 5b). Since the trained policy never sees the oracle, these recoveries also show that the mistakes are recoverable without its privileged information. It also raises the recovery

rate by $+26.9\%$ over the preventive-only variant, i.e., PIVOTOPD with recovery budget $K = 0$, so explicit recovery distillation improves recovery beyond what preventive training alone achieves.

Recovery requires supervision at the right turns and on the right actions. To see what makes this recovery learnable, we train variants of PIVOTOPD on ALFWorld that each remove or replace a single component (Table 2 and Appendix E). Neither the preventive-only nor the recovery-only variant matches the average success rate of PIVOTOPD, so the two distillation terms are complementary. This holds even though the preventive-only variant compensates with a larger preventive weight (Appendix E). The budget ablation shows a larger effect of recovery: the selected budget raises the best validation score over $K = 0$ by 5.4 points on ALFWorld, 21.5 on WebShop, and 2.4 on Search-based QA (Figure 7). When we inject the same hints at random turns, the average success rate is the lowest among all variants, indicating that supervision must land at pivotal turns. Similarly, when a generic reflection prompt replaces the gold action in each hint, the average success rate stays close to that of preventive only but drops sharply on Pick2 (Table 6), suggesting that the hint must name the correct action on task types whose decisions cannot be inferred from the observations alone.

Variant	Clean	Heat	Cool	**Avg.**
Random pivotal turns	76.2	61.1	50.0	62.6
Hints w/o gold actions	90.5	63.7	72.9	71.9
Reverse-KL recovery	85.7	50.0	61.5	64.5
Recovery only	66.7	40.9	53.9	64.7
Preventive only	93.2	70.2	72.9	72.5
PIVOTOPD	**93.7**	**72.6**	**73.9**	**73.7**

Table 2: **Component ablations on ALFWorld with the Qwen3-1.7B student.** Each variant changes one component of PIVOTOPD (preventive only also raises w_{prev}). The average is over all six task types (Table 6).

Recovery distillation restores the learning signal that on-policy updates lose after a pivotal mistake. Beyond where and what to supervise, we now ask why recovery distillation provides a learning signal when the student rarely samples the recovery action. Proposition 1 predicts that without recovery distillation, the student stays far from its privileged self-teacher after a pivotal mistake. In Figure 6, we track the forward KL divergence from the self-teacher to the student, which measures how far the student remains from its self-teacher (Appendix E.3). After the mistake, the preventive-only student indeed remains at least twice as far from its self-teacher as PIVOTOPD at every displayed turn. This gap persists well beyond the single recovery turn that PIVOTOPD trains in this run ($K = 1$), suggesting that the learned recovery carries over to later turns rather than fitting the trained turn alone.

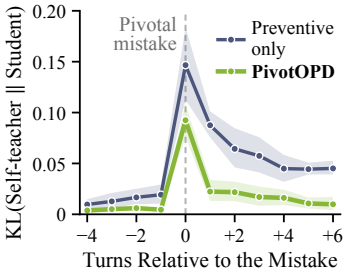


Figure 6: Recovery distillation quickly brings the KL divergence from the self-teacher back down after the pivotal mistake.

The best recovery budget varies across benchmarks. It is $K = 1$ on WebShop and Search-based QA but $K = 2$ on ALFWorld (Figure 7). One plausible explanation, consistent with Proposition 3 in Appendix C, is that ALFWorld tasks require longer sequences of fine-grained actions after a mistake. Appendix E.1 reports the training compute overhead of each budget.

6 Related Work

Our baselines span RL and on-policy self-distillation, alone or combined [27, 50, 40, 17], turn-level distillation for multi-turn agents [53, 34, 51, 49, 38], and guidance from skills or pivotal turns [33, 42, 47], all of which learn from responses that the student samples. Several locate outcome-deciding turns, and OPID's turn-level skills are closest to our preventive distillation. PIVOTOPD additionally trains the student on self-teacher responses at the states that its own mistakes create, so it teaches recovery actions that the student rarely samples (Appendix F).

7 Conclusion

We showed on ALFWorld that failed rollouts often contain a recoverable pivotal mistake that standard OPD does not repair, and proposed PIVOTOPD, which distills a privileged self-teacher at and after pivotal turns. PIVOTOPD attains the best average performance against 13 baselines on three agentic benchmarks and recovers from pivotal mistakes far more often than standard OPD. It also improves a different model family on software engineering.

REPRODUCIBILITY STATEMENT

We describe PIVOTOPD in Section 3 and give further method details, including the teacher report, the hints, and leakage control, in Appendix B. Appendix C states the assumptions and proofs of our analysis of recovery distillation. Appendix D lists the training configurations and hyperparameters, the evaluation protocol, the baseline configurations, and the full prompts. All four benchmarks that we use are publicly available.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work trains and evaluates language agents on public benchmarks, and it involves no human subjects, private data, or personally identifiable information.

AI USE STATEMENT

Beyond the language models that PIVOTOPD trains and uses as teachers, we used generative AI tools for writing and debugging the training and evaluation code, for assisting with the formulation and proofs of the propositions in Appendix C, and for analyzing rollout logs and interpreting results. We have not used generative AI tools for proposing hypotheses or designing the method and experiments, and generating synthetic datasets, translation, and qualitative or thematic analysis are not applicable to this work. Additionally, we used generative AI tools for literature search and reference formatting, for other code generation and editing, for writing the plotting code of the figures, and for drafting and editing the manuscript. We have reviewed all AI-assisted work, including the code, the proofs, and the analyses. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

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

[2] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes LLM better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.

[3] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025.

[4] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024.

[5] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[6] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? *arXiv preprint arXiv:2310.06770*, 2023.

[7] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[8] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

[9] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

[10] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *IEEE International Conference on Robotics and Automation*, 2019.

[11] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, 2016.

[12] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

[13] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural Questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[14] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[15] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[16] Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

[17] 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.

[18] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 9802–9822, 2023.

[19] NVIDIA. NVIDIA Nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

[20] Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models, 2026. URL https://arxiv.org/abs/2602.04942.

[21] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems*, 2018.

[22] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.

[23] Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *Advances in Neural Information Processing Systems*, 2024.

[24] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 661–668, 2010.

[25] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[27] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[28] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[29] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[30] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[31] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[32] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009.

[33] Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents, 2026. URL https://arxiv.org/abs/2604.10674.

[34] Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. TCOD: Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents, 2026. URL https://arxiv.org/abs/2604.24005.

[35] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[36] Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2024.

[37] Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *arXiv preprint arXiv:2402.11651*, 2024.

[38] Zi-Han Wang, Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Jie Wu, Zhengzhou Cai, Yueqing Sun, Ziang Ye, Linji Hao, Qi Gu, Xunliang Cai, Yongliang Shen, and Yujiu Yang. AgentOPSD: Recursive self-distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2608.05987.

[39] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[40] Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled RLVR, 2026. URL https://arxiv.org/abs/2604.03128.

[41] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

[42] 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*, 2026.

[43] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

[44] 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.

[45] 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.

[46] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

[47] Junkeun Yi, Damon Mosk-Aoyama, Baihe Huang, Ritu Gala, Charles Wang, Sugam Dipak Devare, Khushi Bhardwaj, Abhibha Gupta, Oleksii Kuchaiev, Jiantao Jiao, Jian Zhang, and Venkat Srinivasan. PivotRL: High accuracy agentic post-training at low compute cost, 2026. URL https://arxiv.org/abs/2603.21383.

[48] Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

[49] Yanfei Zhang, Xu Lin, and Chenglin Wu. StepOPSD: Step-aware online preference self-distillation for agent reinforcement learning, 2026. URL https://arxiv.org/abs/2605.27140.

[50] 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, 2026. URL https://arxiv.org/abs/2601.18734.

[51] Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. SOD: Step-wise on-policy distillation for small language model agents, 2026. URL https://arxiv.org/abs/2605.07725.

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

[53] Yuhang Zhou, Kai Zheng, Haoling Li, Dengyun Peng, Can Xu, and Jingjing Chen. TurnOPD: Making on-policy distillation turn-aware for efficient long-horizon agent training, 2026. URL https://arxiv.org/abs/2607.05804.

A Motivating Analysis Details

This appendix gives the protocol behind Figure 1 and Section 2.

Rollouts. We roll out three models, Qwen3-8B, Qwen3-30B-A3B, and Qwen3-235B-A22B, on the 140 held-out ALFWorld tasks of the `valid_seen` split, sampling at temperature 0.4 with a cap of 30 turns and a history window of 5 turns. The analysis runs on ALFWorld because it is the one benchmark of the three that provides a symbolic oracle, and the oracle is what makes pivotal turns identifiable without a model in the loop. PivotOPD does not use this oracle during training (Section 3.2).

Oracle-labeled pivotal turns. Every recorded trajectory is replayed in the environment alongside the ALFWorld PDDL oracle, which recomputes the number of turns that an optimal trajectory still needs, i.e., $L(s_t)$ in Section 3.1. A turn is pivotal when the action committed there raises this number or leaves the task unsolvable. Among the failed trajectories, 72 of 111 contain a pivotal turn for Qwen3-8B, 35 of 70 for Qwen3-30B-A3B, and 48 of 81 for Qwen3-235B-A22B, which amounts to 155 of 262 across the three models. The analysis uses the first pivotal turn of each failed trajectory, with the oracle action at that turn as the correction. Before this turn, the agent has not moved away from completing the task, so correcting it gives a clean counterfactual. The replay reproduces the recorded outcome for all 420 trajectories, so the counterfactuals below are exact rather than approximate.

Counterfactual replay. Qwen3-8B fails 111 of the held-out tasks, and 72 of these failed trajectories contain a pivotal turn. From the pivotal turn of each, we replay the remainder of the trajectory with the same model under the five interventions of Figure 1b: no intervention, the model resampled at the pivotal turn, the oracle action forced at another turn drawn at random from the turns that are not pivotal, the oracle action forced at the pivotal turn itself, and the oracle action forced at each of the two turns after the mistake, with the mistake left in place and the oracle re-queried at every forced turn. Each intervention is repeated four to eight times per trajectory. The later-turn intervention is a control for the position of the correction, and it controls for position only when the drawn turn lands after the pivotal turn, since an earlier correction also erases the model's own mistake. We therefore report it over the 51 trajectories where the drawn turn does land later, while the other interventions use all 72. This control forces the oracle action later in the trajectory and has fewer turns left to work with, but success after correcting the pivotal turn is flat in the position of that turn, between 57% and 63% across position tertiles. Extending the correction after the mistake from two turns to three adds about four points, so the two-turn result is not an artifact of how many turns we correct.

Confidence intervals. For each intervention, we average the replayed success over the replays of each trajectory and compute a 95% bootstrap confidence interval over trajectories (4,000 draws, seed 0), which we give in brackets in percent. Without intervention, the replayed success is 8.2% [3.8, 13.0], and resampling the pivotal turn gives 6.9% [2.4, 12.2]. Forcing the oracle action at a later turn gives 17.2% [8.3, 27.5]. Correcting the pivotal turn instead gives 59.0% [48.3, 69.8], and forcing the oracle action at the two turns after the mistake gives 58.3% [47.6, 69.1], or 62.2% [51.4, 72.9] with three forced turns. The intervals of correcting the pivotal turn and of forcing the oracle action after the mistake therefore lie entirely above those of the other three interventions.

On-policy distillation run. Qwen3-8B is trained from its base checkpoint for 120 steps with standard on-policy distillation. Here, standard OPD is the OPSD baseline of Table 1, which distills a privileged self-teacher [50] under the configuration of Appendices D.1 and D.3. For Figure 1c, we roll out each saved checkpoint on the same 140 held-out tasks, label the rollouts with the oracle exactly as above, and split each checkpoint's failures into those that fail at a pivotal turn and those whose trajectory contains no pivotal turn and leaves the task unfinished. Among the 72 tasks that the base model loses at a pivotal turn, roughly seven in ten are lost again after distillation, and about half of the 72 repeat the same kind of mistake. We score the probability that the model assigns to the oracle action under the frozen policy at each pivotal turn, before and after distillation. Distillation moves the median probability of the committed mistake from 0.999 to below 10^{-5} and raises the median probability of the oracle action by ten orders of magnitude. The raised probability nevertheless stays below 10^{-2} at every pivotal turn, so a group of 8 rollouts is not expected to sample the

oracle action even once. This run is shorter than the 160-step runs of Section 4.1, and its numbers are not comparable to Table 1, though both sides of every comparison within this analysis are measured under the same protocol.

B METHOD DETAILS

This appendix details the components of PIVOTOPD summarized in Section 3, and Algorithm 1 presents one training step.

Teacher report and hints. For each trajectory, the teacher returns one structured reply containing a trajectory summary, a trajectory-level lesson, a per-turn lesson, and gold-action blocks that name an action for up to m candidate turns. The trajectory-level lesson is the brief feedback on the whole trajectory that Section 3.3 mentions. It is inserted into the prompt at every turn of the trajectory and scored with the distillation advantage A_ℓ^{distill}, weighted by w_{traj} in place of w_{prev}. For brevity, Eq. 3 omits this term. Replies that name no action contribute only the lesson signal and never trigger recovery. The hint $h(a)$ renders the action as a short passage. The passage states that a is a reasonable approach at this turn and instructs the model to reason toward it from scratch, in its own style, without referencing, acknowledging, or quoting the hint. The exact templates are in Appendix D.4.

Action resolution. The teacher's free-text action is matched onto the admissible set by normalized exact equality, then by equality after stripping articles, then by containment by a unique admissible command, and finally by highest token overlap above a fixed threshold, with a strict winner. If none of the rules resolves a reply, the recovery is dropped. For the open action spaces of Search-based QA and SWE-Bench, no matching is required: the teacher's named action is taken verbatim, subject to the same schema-validity check as student actions, i.e., a well-formed search query or answer for Search-based QA, and for SWE-Bench, a call that parses against the scaffold's tool interface and executes in the task container.

Mismatch check. A candidate turn is pivotal when the student's committed action differs from the gold action after both are lowercased and their whitespace is collapsed. In Search-based QA, both actions are first rewritten in the forms `search[·]` and `answer[·]`. A response without a parsable action counts as a mismatch. Free-text actions such as search queries rarely match verbatim, so candidate turns with such actions are usually treated as pivotal. This is one source of the false positives discussed in the next paragraph.

Relation to oracle-labeled pivotal turns. Pivot detection relaxes the oracle labeling of Section 2 in four ways. **(1)** The teacher's judgment replaces the oracle. Although ALFWorld provides an oracle, we do not use it during training, so that PIVOTOPD stays identical across benchmarks and does not depend on a benchmark-specific oracle. **(2)** A trajectory may contain several pivotal turns, because the agent can move away from completing the task more than once. **(3)** Pivot detection covers successful trajectories as well as failed ones, because a successful trajectory can still contain a pivotal mistake from which the student happens to recover. **(4)** Disagreement with the gold action replaces an increase in L, so an equally reasonable alternative action can be flagged as pivotal. Such false positives are inexpensive. The preventive weight w_{prev} is small (Table 5), and recovery distillation at such a turn still trains the student toward the teacher's next action at a state that the student actually visited.

Teacher–oracle agreement. We measure how often pivot detection finds the oracle-labeled pivotal turns on ALFWorld. To this end, we run each training teacher on ALFWorld rollouts of the student that it trains, i.e., Qwen3-30B-A3B on Qwen3-1.7B and Qwen3.5-122B-A10B on Qwen3-8B (Section 4.1). As in Appendix A, we keep the failed trajectories that contain an oracle-labeled pivotal turn. Each teacher uses the same prompt, parsing, and decoding configuration as in training (Appendix D.4). It therefore receives the full trajectory and its outcome, may choose among all turns, and selects up to $m = 5$ candidate turns. We draw 5 samples per trajectory and score only the teacher-detected pivotal turns, i.e., the candidate turns whose gold action disagrees with the student's committed action. A sample is correct when at least one of these turns lies within one turn of the first

oracle-labeled pivotal turn, and a reply that cannot be parsed counts as incorrect. We average correctness over the samples of each trajectory and then over trajectories. The random baseline selects as many turns as the sample's teacher-detected pivotal turns, uniformly at random without replacement, and applies the same criterion. Its value therefore differs between the two teachers, which are evaluated on different trajectories and detect different numbers of pivotal turns. Both teachers agree with the oracle at least twice as often as the random baseline (Table 3). In Section 3.2, we say that at least one teacher-detected pivotal turn falls within one turn of the oracle-labeled pivotal turn when a sample is correct in this sense, and we report the mean accuracy of the two teachers, 77.8%.

Teacher	Student	± 1-turn accuracy (%)	Random baseline (%)	Gain over random (%)
Qwen3-30B-A3B	Qwen3-1.7B	71.1	33.6	+37.5
Qwen3.5-122B-A10B	Qwen3-8B	84.4	28.1	+56.3

Table 3: **Teacher–oracle agreement on ALFWorld.** Each teacher analyzes failed rollouts of the student that it trains. Accuracy is the percentage of teacher samples with at least one teacher-detected pivotal turn within one turn of the first oracle-labeled pivotal turn, averaged over failed trajectories with such a turn. The random baseline selects the same number of turns uniformly at random, and the last column subtracts it from the accuracy.

Leakage control. A generated recovery response is discarded if it matches any of a family of leakage patterns covering acknowledgments of a hint, suggestion, or provided approach. Training sequences are rebuilt from the unprivileged observation, and an assertion rejects any training prompt containing a privileged marker. As a result, hinted text can never enter the training prompt.

Environment replay. Before taking recovery turns, multi-turn recovery replays the recorded action prefix in a pooled copy of the environment and verifies that the reached observation matches the recorded one exactly. A mismatch aborts the recovery. For $k \geq 2$, the recovery context $\tilde{c}_{t+k}$ is reached by executing $\mathrm{act}(y_{\text{rec}})$ of the previous recovery turn in this replayed environment rather than recorded in the original trajectory. The teacher is then queried again at $\tilde{c}_{t+k}$ for the next recovery action a^*_{t+k}. Each recovery turn adds one training sequence with its own loss $\mathcal{L}^{\text{rec}}_{t,k}$. Beyond this check, at most 64 recoveries are processed per training step, which bounds the time that recovery rollouts add to each step.

Combined advantage and surrogate. The distillation terms of Eq. 3 enter the PPO update as per-token advantages built from the distillation advantage A^{distill}_ℓ of Eq. 4,

$$A^{\text{prev}}_\ell = w_{\text{prev}}\, A^{\text{distill}}_\ell, \qquad A^{\text{rec}}_\ell = w_{\text{rec}}\, \mathrm{clip}_{[-\delta,\,\delta]}\big(A^{\text{distill}}_\ell\big), \tag{5}$$

where ℓ indexes the tokens of the scored sequence and the clip bound δ caps the distillation advantage on any single token. For prevention, the distillation advantage is evaluated at the pivotal context c_t with the gold action a^*_t and applied to the recorded response y_t. Positive values increase the weight of tokens preferred by the hinted view, and negative values decrease the weight of tokens that it disfavors, so the update shifts the student's response toward the gold action. We keep the preventive weight w_{prev} small (Table 5), since recovery distillation provides the main signal after a pivotal mistake. For recovery, the distillation advantage is evaluated at the recovery context $\tilde{c}_{t+k}$ with the recovery action a^*_{t+k} and applied to y_{rec}. Unlike Eq. 1, the recovery loss of Eq. 2 takes its expectation under the frozen self-teacher, and gradients again flow only through π_θ. The hint enters only this frozen target, while the trained response is conditioned on the unprivileged context $\tilde{c}_{t+k}$. Relative to standard on-policy self-distillation [50], the privileged view is the hint that names the gold action, and the preventive term applies only at the pivotal turns rather than on every response. On a recovery sequence, the distillation advantage is near zero on the text that the plain policy would emit regardless and largest on the tokens that encode the recovery action, so the update concentrates where the hint mattered. The advantages combine with the group-relative advantage as

$$A_\ell \;=\; \begin{cases} A^{\text{rec}}_\ell & \text{on recovery sequences,} \\ A^{\text{RL}}_\ell + A^{\text{prev}}_\ell & \text{otherwise, where } A^{\text{prev}}_\ell = 0 \text{ off pivotal turns.} \end{cases} \tag{6}$$

In both cases, ℓ indexes the tokens of the sequence. Writing $\rho_\ell(\theta) = \pi_\theta(y_\ell \mid c, y_{<\ell}) \,/\, \pi_{\bar{\theta}}(y_\ell \mid c, y_{<\ell})$ for the token-level ratio at the context c of each sequence, one PPO update minimizes the clipped surrogate

$$\widehat{\mathcal{L}}(\theta) \;=\; -\frac{1}{\sum_{(c,\,y)} |y|} \sum_{(c,\,y)} \sum_{\ell=1}^{|y|} \min\Big(\rho_\ell(\theta)\, A_\ell,\; \mathrm{clip}_{[1-\epsilon_{\text{clip}},\; 1+\epsilon_{\text{clip}}]}\big(\rho_\ell(\theta)\big)\, A_\ell\Big), \tag{7}$$

where the outer sum runs over every rollout and recovery sequence of the training step, ℓ indexes the tokens of each sequence, and ϵ_{clip} is the clip ratio. These advantages implement the corresponding terms of Eq. 3, exactly in expectation for prevention and as a clipped, mass-covering step for recovery (Propositions 1 and 2). The surrogate also carries a low-variance KL penalty (Appendix D.1).

Algorithm 1 One training step of PIVOTOPD

Require: frozen policy $\pi_{\bar{\theta}}$, teacher M, group size G, weights $w_{\text{prev}}, w_{\text{traj}}, w_{\text{rec}}$, clip δ, recovery turns K

collect G rollouts per task with $\pi_{\bar{\theta}}$; compute group-relative advantages A^{RL}
for each trajectory τ **do**
  $\{(t_j, a^*_{t_j})\}_{j \le m} \leftarrow M(\tau)$ ▷ candidate turns, gold actions, and lessons
  for each candidate turn t with gold action a^*_t **do**
    if $a_t \neq a^*_t$ **then** ▷ pivotal turn, $t \in \mathcal{T}_{\text{pivot}}$
      add A^{prev} on the tokens of y_t ▷ Eq. 5
      $c \leftarrow c_{t+1}$
      for $k = 1, \ldots, K$ **do**
        $a^* \leftarrow \text{resolve}\big(M(c),\, \mathcal{A}(c)\big)$; **break** if unresolved
        $y_{\text{rec}} \sim \pi_{\bar{\theta}}(\cdot \mid c,\, h(a^*))$; **break** if no action or hint leaked
        append (c, y_{rec}) with advantage A^{rec} and all other signals zeroed ▷ Eq. 5
        if $k < K$ **then**
          $c \leftarrow$ context after executing $\mathrm{act}(y_{\text{rec}})$ in the replayed environment
        end if
      end for
    end if
  end for
end for
update θ on all sequences with the PPO clipped surrogate ▷ Eq. 7

C Analysis of Recovery Distillation

This appendix proves Proposition 1 of Section 3.4, states and proves two further propositions, and records a surrogate characterization of the recovery update. Parts (ii) and (iii) of Proposition 1 follow the standard contrast between reverse and forward KL [1, 4], and what is specific to PIVOTOPD is that recovery distillation applies the forward update at the states that the student's own mistakes create. We analyze the committed action at a single recovery turn as one categorical decision, which is the standard abstraction for token-level updates. Every statement holds verbatim per token, with the conditional next-token distributions in place of the action marginals and summed over the prefixes that the hinted policy visits. Fix a recovery turn with context $c = \tilde{c}_{t+k}$ and recovery action $a^* = a^*_{t+k}$, and write $p = \pi_{\bar{\theta}}(\mathrm{act}(y) = \cdot \mid c)$ and $q = \pi_{\bar{\theta}}(\mathrm{act}(y) = \cdot \mid c,\, h(a^*))$ over $\mathcal{A}(c)$. Write u for the logits, so that $p_\theta = \mathrm{softmax}(u)$ has full support. At the start of an update, the PPO ratio equals one, so the surrogate gradient reduces to the advantage-weighted score function. All gradients are therefore evaluated at $p_\theta = p$.

The first result shows why preventive distillation and group-based RL provide little learning signal on the recovery action.

Proposition 2 (Learning signal vanishes after a mistake)**.** *When the action-level distillation advantage* $\log q(a) - \log p(a)$ *is used as the per-action training signal on actions sampled from* p*, as in preventive distillation, the expected update is* $-\nabla_u D_{\mathrm{KL}}(p_\theta \,\|\, q)$*. The component of this update on an action* a *is* $p(a)\big(\log \frac{q(a)}{p(a)} + D_{\mathrm{KL}}(p \,\|\, q)\big)$*, which vanishes on the recovery action as*

$p(a^*) \to 0$ *at any fixed* q*. In a group of* G *plain rollouts, the recovery action appears with probability* $1-(1-p(a^*))^G \le G\,p(a^*)$*, and it first appears after* $1/p(a^*)$ *plain samples in expectation, against* $1/q(a^*)$ *under the hinted policy. Moreover, if every rollout of the task receives the same return, the group-relative advantage is identically zero.*

Proof of Proposition 2. Since $\mathbb{E}_{p_\theta}[\nabla_u \log p_\theta] = 0$, $\nabla_u D_{\mathrm{KL}}(p_\theta \| q) = \mathbb{E}_{a\sim p}[(\log p(a) - \log q(a))\nabla_u \log p_\theta(a)]$, which is the negative of the stated update. With $\partial \log p(b)/\partial u_a = \mathbf{1}[a=b] - p(a)$, the component on action a is $p(a)\log\frac{q(a)}{p(a)} + p(a)\,D_{\mathrm{KL}}(p\|q)$, whose magnitude is at most $p(a)\big(|\log\frac{q(a)}{p(a)}| + D_{\mathrm{KL}}(p\|q)\big)$ and therefore vanishes as $p(a^*)\to 0$ at fixed q. For the sampling claims, the first appearance of a^* under independent draws from p is geometric with success probability $p(a^*)$. This gives the mean $1/p(a^*)$ and, by Bernoulli's inequality, $\Pr[a^* \text{ appears among } G \text{ draws}] = 1-(1-p(a^*))^G \le G\,p(a^*)$. The hinted case is identical with q in place of p. For the advantage claim, the group-relative advantage subtracts the group mean return, so identical returns give zero advantage on every token. □

For the clipped loss, the trainer-level analog of the identity in Proposition 2 appears in prior work [42]. The component form in turn isolates what preventive distillation and group-based RL cannot do.

Lemma 1 (Lower bound on the recovery loss)**.** *Let* π *and* π' *be two response distributions at the same context, and let* q *and* p *be their induced distributions over committed actions. Then for every action* a *with* $p(a) > 0$*,*

$$D_{\mathrm{KL}}(\pi \,\|\, \pi') \;\ge\; q(a)\log\frac{1}{p(a)} - \log 2. \tag{8}$$

In particular, with π *the privileged self-teacher and* π' *the frozen student at* $\tilde{c}_{t+k}$*, the recovery loss of Eq. 2 is at least* $q(a^*)\log(1/p(a^*)) - \log 2$*.*

Proof. The indicator $\mathbf{1}[\mathrm{act}(y)=a]$ is a function of the response y, so the data processing inequality gives $D_{\mathrm{KL}}(\pi\,\|\,\pi') \ge q(a)\log\frac{q(a)}{p(a)} + (1-q(a))\log\frac{1-q(a)}{1-p(a)}$. Since $1-p(a)\le 1$, the second term is at least $(1-q(a))\log(1-q(a))$. The right-hand side is therefore at least $q(a)\log\frac{1}{p(a)} - H(q(a))$, where $H(x) = -x\log x - (1-x)\log(1-x) \le \log 2$ is the binary entropy. □

Lemma 1 holds for any hint, including the recorded hints of Appendix E.3. It shows that the divergence from the self-teacher stays large whenever the self-teacher places substantial probability on an action that the student almost never takes, and that it can fall only when the student raises the probability of that action.

Proof of Proposition 1. Part (i) is Lemma 1 with $a = a^*$. For parts (ii) and (iii), let actions be sampled from a distribution r and weighted by a signal w. Since $\partial \log p_\theta(b)/\partial u_a = \mathbf{1}[a=b] - p(a)$ at $p_\theta = p$, the expected update on u_a is $\sum_b r(b)\,w(b)\big(\mathbf{1}[a=b] - p(a)\big) = r(a)\,w(a) - p(a)\,\mathbb{E}_r[w]$. Taking $r = p$ gives part (ii). For a bounded signal, $|w(a^*) - \mathbb{E}_p[w]| \le 2\max_a |w(a)|$, so the expected update on u_{a^*} is at most $2\,p(a^*)\max_a|w(a)|$ in magnitude. For part (iii), take $r = q$ and $w = \phi_\delta$, where $\phi_\delta(a) = \mathrm{clip}_{[-\delta,\delta]}\big(\log\frac{q(a)}{p(a)}\big)$ is the clipped distillation advantage. This gives

$$g_a \;=\; q(a)\,\phi_\delta(a) \;-\; p(a)\,\mathbb{E}_q[\phi_\delta]\,. \tag{9}$$

Under the condition $q(a^*) \ge e^\delta\,p(a^*)$, we have $\phi_\delta(a^*) = \delta$. Since $\mathbb{E}_q[\phi_\delta] \le \delta$, Eq. 9 gives $g_{a^*} \ge \delta\,q(a^*) - \delta\,p(a^*)$, which is positive because the condition implies $q(a^*) > p(a^*)$. □

Without clipping, the recovery update on a^* grows without bound as the student's probability of a^* vanishes, and it stops only when the student's distribution matches the self-teacher's.

Lemma 2 (Unclipped recovery update)**.** *Suppose that* $q(a^*) > p(a^*)$*. Without clipping,* $g_{a^*} \to \infty$ *as* $p(a^*)\to 0$ *while the remaining mass stays bounded away from zero. Moreover, the unclipped expected update of Eq. 9 vanishes in every component if and only if* $p = q$*.*

Proof. Without clipping, Eq. 9 reads $g_{a^*} = q(a^*) \log \frac{q(a^*)}{p(a^*)} - p(a^*)\, D_{\mathrm{KL}}(q\|p)$. Write $D_{\mathrm{KL}}(q\|p) = q(a^*) \log \frac{q(a^*)}{p(a^*)} + B$, where $B = \sum_{a \neq a^*} q(a) \log \frac{q(a)}{p(a)}$ stays bounded when the remaining mass is bounded away from zero. Then $g_{a^*} = (1 - p(a^*))\, q(a^*) \log \frac{q(a^*)}{p(a^*)} - p(a^*)\, B \to \infty$. For the second claim, if $p = q$, both terms of Eq. 9 vanish. Conversely, suppose that $q(a) \log \frac{q(a)}{p(a)} = p(a)\, \kappa$ for all a, where $\kappa = D_{\mathrm{KL}}(q\|p) \geq 0$. Writing $r(a) = q(a)/p(a) > 0$, this reads $r(a) \log r(a) = \kappa$ for every a. If $\kappa = 0$, then $r \equiv 1$ and $p = q$. If $\kappa > 0$, then all $r(a)$ equal a single root $r^* > 1$, since $r \log r \leq 0$ on $(0, 1]$ and is strictly increasing on $[1, \infty)$. But then $1 = \sum_a q(a) = r^* \sum_a p(a) = r^* > 1$, a contradiction. □

The last result concerns the recovery budget K.

Proposition 3 (Deeper mistakes need more recovery turns)**.** *Suppose that completing the task after the pivotal turn requires taking a specific action at each of d successive turns, and that the plain policy places mass at most $\varepsilon < 1$ on each required action along this chain. After $K < d$ enforced recovery turns, the plain policy completes the remaining chain with probability at most $\varepsilon^{\,d-K}$.*

Proof of Proposition 3. By the chain rule, the probability of taking all $d - K$ remaining required actions is the product of their conditional probabilities, each at most ε. □

Proposition 3 guides the recovery budget, since K should grow with the depth d of the required chain. Validation is consistent with this, selecting $K = 2$ on ALFWorld, whose tasks chain subgoals over the longest trajectories, and $K = 1$ on WebShop and Search-based QA (Figure 7), although we do not measure the depth of individual mistakes. On ALFWorld, $K = 2$ also matches the two guided turns after the pivotal mistake that restore success in the replay of Figure 1b.

Lemma 3 (Surrogate form of the recovery update)**.** *Let $r = q/p$. The expected recovery update of Eq. 9 is the exact gradient of the frozen surrogate $F(u) = \mathbb{E}_{a \sim p_\theta}[r(a)\, \phi_\delta(a)]$ at $p_\theta = p$, and without clipping, $F = D_{\mathrm{KL}}(q \,\|\, p)$ at $p_\theta = p$.*

Proof. $\nabla_u F = \mathbb{E}_{p_\theta}[r\, \phi_\delta \nabla_u \log p_\theta]$, and at $p_\theta = p$, the reweighting by r turns the expectation over p into one over q, which yields the update of Proposition 1. Without clipping, at $p_\theta = p$, $F = \sum_a p(a) \frac{q(a)}{p(a)} \log \frac{q(a)}{p(a)} = D_{\mathrm{KL}}(q\|p)$. □

Three remarks connect the analysis to the implementation. First, the acceptance filters of Section 3.3 replace the sampling distribution q by its conditional on the accepted event while leaving the coefficients unchanged. The formulas above thus hold with $q(a)$ read as the accepted mass, and the expected update on the recovery action stays proportional to the mass that the filtered self-teacher places on it. Second, all statements concern the recovery context $\tilde{c}_{t+k}$, which follows the student's own mistake, and the student is trained without the hint. Supervised fine-tuning enforces teacher actions at the states that the teacher visits, whereas recovery distillation enforces the recovery action at the state produced by the student's own mistake with a single hinted generation. Recovery therefore avoids the compounding mismatch of imitation at the states that only an expert visits [25]. Third, Propositions 1 and 2 make the division of labor in PivotOPD precise. Preventive distillation tempers the student where it already places mass, and recovery distillation plants mass exactly on the modes that the student has starved, at the turns that its mistakes produce.

D Experiment Details

This appendix complements the setup in Section 4.1 with the full training configurations (Appendix D.1), the per-benchmark evaluation protocols (Appendix D.2), and the baseline descriptions and configurations (Appendix D.3).

D.1 Training Configurations

All methods share one on-policy training stack built on verl and train on H100 nodes. Training uses the standard ALFWorld training tasks, the WebShop instructions outside the held-out split,

and the Search-R1 training split of Natural Questions and HotpotQA (169,615 questions). The student is served by vLLM for rollouts (tensor parallel 1, thinking disabled) and updated with FSDP. Rollouts sample at temperature 1.0 during training and 0.4 at validation. Optimization uses token-mean loss aggregation, gradient clipping 1.0, a low-variance KL loss, and discount $\gamma = 0.95$. Teachers are served as vLLM endpoints with a 32K context window, with tensor parallel 2 for Qwen3-30B-A3B and 8 for Qwen3.5-122B-A10B. Teacher calls use the served model's default sampling configuration. Table 4 lists the per-benchmark hyperparameters, and Table 5 lists the PIVOTOPD configuration. The recovery weight w_{rec} is selected per benchmark with a halving sweep on validation performance, and on Search-based QA, the sweep selects a smaller weight for the 8B student. The recovery budget K and the weight w_{rec} interact, since each additional recovery turn adds one more distilled sequence per pivotal turn and thereby multiplies the total recovery signal. Deeper recovery therefore calls for a proportionally smaller weight, and on WebShop, $K = 2$ requires a quarter of the $K = 1$ weight to remain stable. Baselines receive the same tuning budget.

	ALFWorld	**WebShop**	**Search-based QA**
Tasks per step	16	16	64
Rollouts per task (group size)	8	8	8
Trajectories per step	128	128	512
PPO mini-batch size	128	64	256
Learning rate	1×10^{-6}	1×10^{-6}	1×10^{-6}
KL coefficient	0.01	0.01	0.01
Clip ratio	0.2	0.2	0.2
Prompt / response length	2048 / 512	4096 / 512	4096 / 512
Max turns per trajectory	30	15	4
History length	5	2	4
Training steps	160	160	160
Validation size / frequency	128 / 10	128 / 10	256 / 10

Table 4: Training hyperparameters per benchmark. Both students share this configuration.

	ALFWorld	**WebShop**	**Search-based QA**
Candidate turns per trajectory m	5	5	2
Recovery turns K	2	1	1
Recovery weight w_{rec}	1.0	0.25	0.0625 (8B) / 0.125 (1.7B)
Preventive / lesson weights w_{prev} / w_{traj}	0.001 / 0.001	0.001 / 0.001	0.001 / 0.001
Recovery clip δ	5.0	5.0	5.0
Max recoveries per training step	64	64	64

Table 5: PIVOTOPD hyperparameters per benchmark. The preventive-only ablation uses $w_{\text{prev}} = 0.1$ (Appendix E).

D.2 EVALUATION DETAILS

For every method and benchmark, we train for 160 steps, select the best checkpoint on a validation set, and evaluate the selected checkpoint on a held-out test set. Every experiment is run with three random seeds, and all reported results are averages over them. Validation and test rollouts sample at temperature 0.4 and are capped at 30 turns on ALFWorld, 15 on WebShop, and 4 on Search-based QA.

ALFWorld. Checkpoint selection uses 128 validation tasks evaluated every 10 training steps at temperature 0.4. The selected checkpoint is evaluated on all 274 held-out tasks, i.e., the 140 tasks of the `valid_seen` split and the 134 tasks of the `valid_unseen` split. The reported average is the unweighted mean of the six task-type success rates.

WebShop. The benchmark provides 6,910 instructions over 1,000 products. The first 500 instructions are held out, and the rest are used for training. Checkpoint selection uses a 128-instruction

validation set evaluated every 10 steps. The selected checkpoint runs on each of the 500 held-out instructions. The success rate counts strictly completed purchases, and the score is the dense task score. Product retrieval uses BM25 ($k_1 = 1.5$, $b = 0.75$) over the title, description, category, bullet-point, and option fields at both training and evaluation.

Search-based QA. Checkpoint selection uses a mixed validation set of 256 questions evaluated every 10 steps. The per-dataset accuracies use a balanced evaluation set of 725 questions, drawn with a fixed seed from a held-out pool of 51,713 questions, with 100 per dataset plus all 125 Bamboogle questions. The metric is strict exact match, and the reported average is the unweighted mean of the seven per-dataset accuracies. The agent queries an E5 retriever [35] over the 2018 Wikipedia corpus and receives the top-3 passages per call, identically at training and evaluation.

D.3 Baseline Descriptions and Configurations

Standard post-training.

- **GRPO** [27] optimizes a group-relative advantage from outcome rewards.
- **OPSD** [50] distills a privileged self-teacher on the student's own responses.
- **RLSD** [40] combines the two through teacher-guided advantage rescaling.
- **SDAR** [17] combines the two through a gated distillation loss.

Turn-level distillation for multi-turn agents.

- **TurnOPD** [53] makes the distillation signal turn-aware.
- **TCOD** [34] expands the distilled trajectory depth from short to long with a curriculum.
- **SOD** [51] reweights turn-level divergences.
- **StepOPSD** [49] weights turns by hindsight from successful trajectories in the same rollout group.
- **AgentOPSD** [38] converts outcome rewards into turn-level credit through recursive self-distillation.

High-level guidance through skills or pivotal turns.

- **Skill-GRPO**, the skill-augmented GRPO baseline of OPID [42], injects retrieved skills into training prompts.
- **Skill-SD** [33] conditions the self-teacher on retrieved skills.
- **OPID** distills trajectory- and turn-level skills extracted from the student's own rollouts.
- **PivotRL** [47] concentrates updates on pivotal turns identified from rollout outcomes.

Configurations. All baselines share the benchmarks, data, and optimization of Appendix D.1, and GRPO uses the same advantage estimator with no teacher signal. The skill-based baselines (OPSD, Skill-SD, RLSD, SDAR, and Skill-GRPO) draw on a per-benchmark skill bank distilled from the student's own earlier rollouts, with 47 entries for ALFWorld, 200 for WebShop, and 75 for Search-based QA. Skill-GRPO injects the top-6 retrieved skills into training prompts only. StepOPSD instead uses the first successful trajectory from the same rollout group as hindsight guidance. The auxiliary distillation coefficient is 1.0 for OPSD, 0.001 for Skill-SD, and 0.01 for SDAR. RLSD and StepOPSD reshape advantages and carry no auxiliary loss.

Implementation differences. Two implementation choices differ from the original papers. OPSD's full-vocabulary divergence is approximated with a sampled-token reverse-KL surrogate, and the RLSD and StepOPSD self-teachers re-synchronize every training step instead of every ten.

D.4 PROMPTS

This appendix lists the prompt templates of PIVOTOPD. Placeholders appear in shaded curly braces, tags that the model must emit are shown in green, and dashed lines separate the parts of each template. Several templates address the model directly, writing "step" for what the paper calls a turn.

Student turn prompt. At every turn, the student receives one prompt containing the task, the current observation, and the admissible actions. It responds with reasoning in `<think>` tags followed by one action in `<action>` tags. The example below is the first turn of a held-out ALFWorld task, with the location and action lists abbreviated; WebShop and Search-based QA use the same structure with their own action formats.

Student turn prompt (ALFWorld)

```
You are an expert agent operating in the ALFRED Embodied Environment.
Your current observation is:  -= Welcome to TextWorld, ALFRED! =-

You are in the middle of a room.  Looking quickly around you, you see
a bed 2, a bed 1, a desk 1, a drawer 11, ..., a drawer 1, a dresser 1,
a garbagecan 1, a safe 1, a sidetable 2, and a sidetable 1.

-------------------------------------------------------------------------

Your task is to:  look at alarmclock under the desklamp.
Your admissible actions of the current situation are:  ['go to bed 1',
'go to bed 2', 'go to desk 1', 'go to drawer 1', ..., 'go to sidetable
2', 'inventory', 'look'].

-------------------------------------------------------------------------

Now it's your turn to take an action.
You should first reason step-by-step about the current situation.
This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible
action for current step and present it within <action> </action> tags.
```

Teacher prompt. For each trajectory, the teacher receives the task, the outcome, the indices of all turns as *eligible turns*, and the formatted trajectory, and it returns the report of Appendix B in one reply. The template calls the eligible turns candidate step indices, and the turns that the teacher selects among them are the candidate turns of Section 3.2. The JSON fields `episode_summary`, `episode_lesson`, and `step_lessons` carry the trajectory summary, the trajectory-level lesson, and the per-turn lessons, and one `<correct_action>` block per chosen turn names the gold action. The action-format example is benchmark-specific, shown here for ALFWorld. On WebShop and Search-based QA, the template appends short environment rules that force each recommended action to be a literal executable action, such as clicking every required option value before a purchase or adding new disambiguating terms to a search query.

Teacher prompt

```
Analyze the following agent episode.  Think deeply, step by step,
about which step was the most critical for the outcome and what the
CORRECT action would have been at that step.

-------------------------------------------------------------------------

You MUST output BOTH of the following:

1) A single JSON object (no other JSON elsewhere) with EXACTLY these
fields:
```

```
{
  "episode_summary":  "string",
  "episode_lesson":  "string",
  "step_lessons":  {
    "<idx>":  "policy-facing imperative lesson text for step idx"
  }
}
```

Use up to {max lesson count} keys in step_lessons, chosen from the candidate indices below. Each step_lessons value should be one short imperative sentence.

2) For EACH step you placed in step_lessons, ALSO emit a block of the exact form:

```
<correct_action step="N">action text the agent should have emitted at step N</correct_action>
```

The action text must match the agent's action format (e.g. go to drawer 1). Do not include explanations or quotes inside the block.

Important constraints:
- Step indexing is 0-based.
- The chosen steps in step_lessons MUST come from the candidate indices list below.
- Wrap the JSON in a single ```json ...``` fenced block.

Episode context:
- Task description: {task description}
- episode_success: {outcome}
- Candidate step indices: {eligible turn indices}
- Interaction trajectory: {formatted trajectory}

Recovery-action prompt. At each recovery turn after a pivotal mistake, the teacher receives the student-facing observation of the current post-mistake state and names the single best next action, which becomes the recovery action of Section 3. The situation block already contains the task, the recent history, and the admissible actions, so the teacher sees exactly what the student sees.

Recovery-action prompt (ALFWorld)

You are an expert alfworld agent. Below is the exact situation another agent is currently facing (its task, recent history, current observation, and the list of admissible actions).

Decide the single best next action to make progress on the task from THIS state.

Rules:
- Choose the action verbatim from the admissible actions listed in the situation.
- Output exactly one block of the form:

```
<correct_action>action text</correct_action>
```

- The action text must match the agent's action format (e.g. go to drawer 1).
- Do not add explanations inside the block. Keep any reasoning outside it brief.

Situation:
{observation at the current post-mistake state}

Hint. The hint $h(a)$ renders a named action as a short passage that only the self-teacher sees. The same passage serves preventive distillation with the gold action and recovery distillation with the recovery action, and the leakage control of Appendix B discards any response that acknowledges it.

Hint $h(a)$

```
A reasonable approach at this step is: {named action}
Reason from scratch in your own style as if you arrived at this
approach independently. Do NOT reference, acknowledge, or quote the
hint above in your output. Decide and act as your own choice.
```

D.5 Benchmark Examples

We show one successful episode from each benchmark to illustrate its tasks, observations, and actions. Each episode is copied from a rollout log. For brevity, we show only the committed action of each turn, omit the reasoning and the list of admissible actions, and abbreviate long observations with [...]. Each numbered card is one turn, and highlighted text marks the evidence that determines the next action. The three episodes come from different policies, namely the Qwen3-8B student after 120 steps of standard on-policy distillation on ALFWorld (Appendix A), Qwen3-235B-A22B on WebShop, and Qwen3-8B during AgentOPSD training on Search-based QA.

ALFWorld. The agent must find a spatula, clean it at the sink, and place it on the dining table. It completes the task in seven turns.

Example episode (ALFWorld, Clean task type)

Task: clean some spatula and put it in diningtable.

[Initial observation] You are in the middle of a room. Looking quickly around you, you see a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, [...], a diningtable 1, [...], a sinkbasin 1, [...], and a toaster 1.

1 **[Action]** `look`
[Observation] You are in the middle of a room. Looking quickly around you, you see nothing.

2 **[Action]** `go to diningtable 1`
[Observation] You arrive at diningtable 1. On the diningtable 1, you see a bread 2, a bread 1, [...], a spatula 2, and a spoon 2.

3 **[Action]** `take spatula 2 from diningtable 1`
[Observation] You pick up the spatula 2 from the diningtable 1.

4 **[Action]** `go to sinkbasin 1`
[Observation] You arrive at sinkbasin 1. On the sinkbasin 1, you see a cup 2, a fork 2, and a knife 1.

5 **[Action]** `clean spatula 2 with sinkbasin 1`
[Observation] You clean the spatula 2 using the sinkbasin 1.

6 **[Action]** `go to diningtable 1`
[Observation] You arrive at diningtable 1. On the diningtable 1, you see a bread 2, a bread 1, [...], and a spoon 2.

7 **[Action]** `move spatula 2 to diningtable 1`
[Observation] You move the spatula 2 to the diningtable 1.

✓ **Success: task completed (reward 10)**

WebShop. The agent must buy a product that matches every attribute, option, and price constraint of the instruction. It searches once, opens a matching product, selects the requested color and size, and buys it.

Example episode (WebShop)

Instruction: Find me men's shorts with drawstring closure, elastic waist for gym workout with color: #2 army green, and size: 30, and price lower than 30.00 dollars

1 **[Action]** `search[men's gym workout shorts drawstring closure elastic waist army green size 30 under $30]`
[Observation] 'Back to Search' [SEP] 'Page 1 (Total results: 50)' [SEP] 'Next >' [SEP] 'B07WMMYB6G' [SEP] 'MAGCOMSEN Men's Gym Workout Shorts with Pockets Mesh Liner Quick Dry Running Shorts for Jogging, Hiking' [SEP] '$17.98 to $17.99' [SEP] 'B092T8CY89' [SEP] [...]

2 **[Action]** `click[b07wmmyb6g]`
[Observation] 'Back to Search' [SEP] '< Prev' [SEP] 'color' [SEP] '#1 black grey' [SEP] [...] [SEP] '#2 army green' [SEP] [...] [SEP] 'size' [SEP] '30' [SEP] '32' [SEP] [...] [SEP] 'Price: $17.98 to $17.99' [SEP] [...] [SEP] 'Buy Now'

3 **[Action]** `click[#2 army green]`
[Observation] (the same product page as in turn 2)

4 **[Action]** `click[30]`
[Observation] (the same product page as in turn 2)

5 **[Action]** `click[buy now]`
[Observation] Thank you for shopping with us! [SEP] [...] [SEP] Purchased [SEP] asin [SEP] B07WMMYB6G [SEP] options [SEP] `{"color": "#2 army green", "size": "30"}` [SEP] [...] [SEP] Your score (min 0.0, max 1.0) [SEP] 1.0

✓ **Success: purchased a product that matches the instruction (score 1.0)**

Search-based QA. The agent must answer a two-hop question. It first searches for the director of the film and then for the TV drama that this director co-created.

Example episode (Search-based QA)

Question: What is the TV drama co-created by this Irish film and TV writer/director who directed the 2001 film "On the Edge"?

1 **[Action]** `<search>Director of the 2001 film "On the Edge"</search>`
[Information] Doc 1: "On the Edge (2001 film)" On the Edge (2001 film) On the Edge is a 2001 Irish film directed by John Carney and starring Cillian Murphy, Tricia Vessey, Jonathan Jackson and Stephen Rea. [...]

2 **[Action]** `<search>TV drama co-created by John Carney</search>`
[Information] Doc 1: "John Carney (director)" John Carney (director) John Carney (born 1972) is an Irish film and TV writer/director who specialises in low-budget indie films. [...] He is also a co-creator of the Irish TV drama series "Bachelors Walk". [...]

3 **[Action]** `<answer>Bachelors Walk</answer>`

✓ **Success: exact match with the gold answer (score 1.0)**

D.6 SWE-Bench Verified

Training. We train on a curated software-engineering curriculum of root-cause bug-fix instances augmented with single-file SWE-rebench tasks, decontaminated against SWE-Bench Verified by removing all overlapping repositories and issues. The agent operates an OpenHands-style function-calling scaffold whose tools are repository exploration, file editing, shell execution, and patch submission, with a 100-turn and 3,600-second budget per episode and a 196K-token context. Each training step rolls out $G = 16$ trajectories for each of 32 instances at temperature 0.6, and the teacher is served as a vLLM endpoint. Because SWE-Bench episodes are long and containerized, we adapt the configuration as follows. For each task group with failures, the teacher audits one failed trajectory at its final committed action ($m = 1$) and names a gold action. The hint uses the same template as in Appendix D.4. Unlike on the other benchmarks, the hinted distribution comes from Nemotron-3-Super rather than from a privileged self-teacher, and it enters the unclipped preventive advantage A_ℓ^{prev} of Eq. 5 at weight $w_{\text{prev}} = 0.1$. The recovery budget is $K = 0$, since the audited turn is the final committed action, after which the recorded trajectory contains no turn at which to apply recovery distillation. Extending pivot detection to earlier turns, where recovery distillation would apply, is left to future work. The standard OPD baseline is on-policy distillation from the same Nemotron-3-Super teacher, which distills its token-level distribution on the student's own trajectories [1], and PivotOPD adds preventive distillation to this objective. Both methods share data, scaffold, and tuning budget, and they train at a constant learning rate of 3×10^{-6}.

Evaluation. We evaluate on the 500 tasks of SWE-Bench Verified using the OpenCode agent scaffold within the NeMo Gym harness. Each task is run once under each of 3 evaluation seeds in an isolated per-task sandbox, and the agent queries our model through a disaggregated vLLM deployment with separate prefill and decode workers. We sample at temperature 1.0 with top-p of 0.95 and report the resolve rate, i.e., the percentage of tasks whose submitted patch passes the associated unit tests, averaged over the three evaluation seeds.

E Additional Results

Table 6 reports the component ablations of Section 5 on every ALFWorld task type. The preventive-only variant sets $K = 0$ and raises the preventive weight from $w_{\text{prev}} = 0.001$ (Table 5) to 0.1. Without recovery distillation, the smaller weight would add only a small pivot-specific signal to the group-relative RL advantage. The larger weight therefore makes preventive only a stronger reference, so its comparison with PivotOPD tests whether recovery distillation adds to a substantial preventive signal rather than to group-based RL alone. The variant with random pivotal turns injects the same hints as PivotOPD at randomly chosen turns under a matched budget. Relative to preventive only, removing gold actions from the hints lowers the success rate the most on Pick2, by 9.3%. This gap suggests that naming the action is necessary on task types whose decisions cannot be inferred from the observations alone.

Variant	Pick	Look	Clean	Heat	Cool	Pick2	**Avg.**
Random pivotal turns	71.4	78.6	76.2	61.1	50.0	38.6	62.6
Hints w/o gold actions	87.3	**82.7**	90.5	63.7	72.9	34.1	71.9
Reverse-KL recovery	75.0	71.4	85.7	50.0	61.5	43.4	64.5
Recovery only	**87.7**	63.1	66.7	40.9	53.9	**76.2**	64.7
Preventive only	83.8	71.4	93.2	70.2	72.9	43.4	72.5
PivotOPD	87.6	55.9	**93.7**	**72.6**	**73.9**	58.5	**73.7**

Table 6: **Full component ablations on ALFWorld with the Qwen3-1.7B student.** Each variant changes one component of PivotOPD and holds the rest of the training recipe fixed, except that preventive only also raises w_{prev} to 0.1. This table extends Table 2 to every task type. Results are averaged over three seeds.

Figure 7 presents the validation trends for the recovery-budget ablation in Section 5, where $K = 0$ is the preventive-only variant. The choice of K has a substantial effect on validation performance.

On WebShop, the score at the end of training differs by up to 19.5% across budgets. More recovery turns, however, do not always improve performance, and increasing the budget from $K = 1$ to $K = 3$ lowers this score by 7.8%.

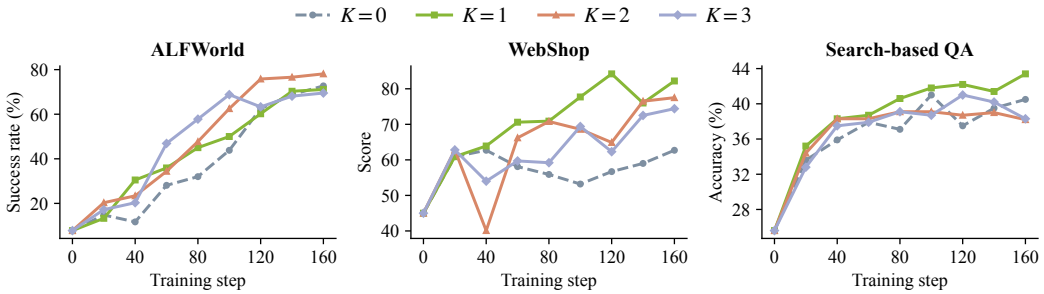


Figure 7: **Ablation on the recovery budget with the Qwen3-1.7B student.** Each panel reports the validation performance trend with different numbers of recovery turns $K \in \{0, 1, 2, 3\}$ during training, where $K = 0$ is the preventive-only variant. Overall, $K = 1$ yields the best performance for WebShop and Search-based QA, while ALFWorld needs $K = 2$ recovery turns.

E.1 Training Compute Overhead

PivotOPD changes only training. At test time, the student acts without the teacher model, hints, or recovery, so PivotOPD adds no computation at inference. During training, preventive and recovery distillation add two computations to group-based RL. The first is the forward pass of the privileged self-teacher, which computes the distillation advantage of Eq. 4. The second consists of the *recovery rollouts*, which query the teacher model for a recovery action and sample a recovery response at every recovery turn (Algorithm 1). We do not include pivot detection, whose cost depends mainly on the choice of teacher model and how it is served.

We time both computations on ALFWorld with the Qwen3-1.7B student on one node of four H100 GPUs, where GRPO takes 367.3 seconds per training step (Table 7). The self-teacher forward pass is the same kind of computation that self-distillation baselines such as OPSD and RLSD perform. With $K = 1$, recovery rollouts are inexpensive because the only recovery turn starts from the post-mistake state that the rollout has already reached, so it needs no environment replay. Each later recovery turn must instead be reached through environment replay (Appendix B), and the time of recovery rollouts grows by more than ten times from $K = 1$ to $K = 2$. Among the three main benchmarks, only ALFWorld uses more than one recovery turn (Table 5). On WebShop, where $K = 1$ is the selected budget, recovery rollouts likewise add only 12.2% to the time per training step of GRPO, measured with 16 GPUs per run over the first 41 training steps.

Computation	$K = 0$	$K = 1$	$K = 2$	$K = 3$
Self-teacher forward pass			17.2 s (+4.7%)	
Recovery rollouts	–	28.5 s (+7.8%)	328.7 s (+89.5%)	395.1 s (+107.6%)
Total overhead	+4.7%	+12.4%	+94.2%	+112.3%

Table 7: **Training compute overhead of PivotOPD on ALFWorld with the Qwen3-1.7B student.** Each entry is the median time per training step of a computation that preventive and recovery distillation add to group-based RL, with the resulting increase over the time per training step of GRPO in parentheses. The last row sums both increases, and $K = 0$ is the preventive-only variant. The overhead stays small with one recovery turn and grows once later recovery turns require environment replay.

E.2 Recovery Analysis Details

This analysis examines whether trained policies can recover after a pivotal mistake has already been committed, complementing the motivating replay study in Appendix A.

Replay protocol. We use the same 72 pivotal mistakes identified by the environment's symbolic oracle in Section 2. These oracle-labeled pivotal mistakes are independent of the teacher-detected pivotal turns used in training, so this evaluation does not reward agreement with the teacher's own labels. For each pivotal mistake, we replay the trajectory prefix up to and including the pivotal mistake in a fresh copy of the ALFWorld environment, verify that the reached state matches the recorded one, and let the policy continue on its own for the remainder of the 30-turn budget. Each prefix is replayed 8 times per policy, giving 576 replays per policy.

Policies. We compare four policies. **Base** is the untrained Qwen3-8B model. **Standard OPD** is the best checkpoint of the OPSD run in Appendix A from a sweep over training steps. **Preventive only** is PivotOPD trained for 160 steps with recovery budget $K = 0$, as in Section 5. **PivotOPD** is the full method trained for 160 steps with $K = 2$.

Recovery curves (Figure 5b, top). A replay counts as recovered within x turns when it completes the task and its continuation after the pivotal mistake uses at most x turns. The curve at each value of x is the per-mistake mean averaged over the 72 pivotal mistakes, so every pivotal mistake contributes equally regardless of how many of its replays succeed. Final recovery rates are 8.3% (Base), 20.3% (Standard OPD), 45.8% (preventive only), and 72.7% (PivotOPD).

Paired comparison per pivotal mistake. Because every policy replays the same 72 pivotal mistakes, we also compare each trained policy with the base model on each pivotal mistake separately. The recovery rate of a pivotal mistake is the fraction of its 8 replays that recover, and the sign of its difference from the rate of the base model marks the pivotal mistake as improved, worsened, or unchanged. Standard OPD improves the recovery rate on 20 of the 72 pivotal mistakes, worsens it on 4, and leaves 48 unchanged. The preventive-only variant improves 47, worsens 6, and leaves 19 unchanged. PivotOPD improves 60 and worsens none, leaving 12 unchanged. These counts come from the same replays as the recovery curves, and they show that the higher recovery rate of PivotOPD extends across most pivotal mistakes rather than coming from a few of them.

Recovery efficiency (Figure 5b, bottom). Among the replays that recover, we report the average number of turns from the pivotal mistake to task completion, with 95% bootstrap confidence intervals (2,000 draws, seed 0). The gray reference bar reports the average remaining optimal trajectory at the post-mistake state, computed over the 72 pivotal mistakes. Because each policy recovers on a different number and mixture of pivotal mistakes, the efficiency values are conditioned on different sets of replays: 48 (Base), 117 (Standard OPD), 264 (preventive only), and 419 (PivotOPD). Despite this difference in conditioning, the per-policy recovered sets have near-identical own-set optimal means (5.2–5.8 turns), so the common reference bar provides a fair comparison across policies.

E.3 Divergence from the Self-Teacher Details

Figure 6 compares the Qwen3-8B checkpoints of the preventive-only variant ($K = 0$) and a separate PivotOPD run with $K = 1$ at training steps 140 and 160. Each pair of checkpoints scores the ALFWorld training rollouts of the preventive-only variant from a disjoint window of steps, i.e., steps 130–149 for the checkpoints at step 140 and steps 150–160 for those at step 160. Both variants therefore score the same responses at the same turns, and every scored state is one that the preventive-only variant visits itself. For each turn, we compute the forward KL divergence from each policy's privileged self-teacher to the policy over the full vocabulary and average it over the response tokens. The self-teacher receives the hints recorded with these rollouts, which include the trajectory-level lesson at every turn and the gold action at candidate turns. The preventive-only variant records no recovery actions, so this divergence uses the recorded hints rather than the recovery-action hint of Eq. 2, and it serves as a proxy for the recovery loss rather than the loss itself (Lemma 1). We align each trajectory at its first teacher-detected pivotal turn and exclude turn 0. The analysis covers 600 trajectories, 351

of which contain a teacher-detected pivotal turn. The shaded bands show 95% bootstrap confidence intervals clustered over trajectories (2,000 draws, seed 0). In Figure 6, both students track their self-teachers closely before the pivotal mistake, and the divergence spikes at the pivotal turn, where the hint names the gold action.

F Related Works

On-policy distillation and the direction of KL. Sequence-level knowledge distillation trains a student on outputs sampled from the teacher [11]. On-policy distillation instead supervises the student on its own samples, and prior work discusses which KL direction suits this setting. GKD compares forward and reverse KL on student-generated sequences [1], and MiniLLM favors reverse KL so that the student does not overestimate regions where the teacher places little mass [4, 16]. For multi-turn agents, recent methods reweight turns or schedule the distilled trajectory depth [53, 34, 51, 49, 38], but they still distill only on student-sampled responses. PivotOPD chooses the direction by where the supervision is needed. Preventive distillation applies reverse KL to a response that the student has already sampled, while recovery distillation applies forward KL to self-teacher responses that the student rarely produces, in the spirit of sequence-level distillation.

Self-correction and learning from failures. Several lines of work teach language models to correct or learn from their own mistakes. Reflexion stores verbal reflections on failed trials in memory instead of updating the model [28]. SCoRe and RISE train models to revise their responses over multiple attempts [12, 23]. For agents, Agent-R splices failed trajectories with correct continuations found by Monte Carlo tree search [48]. ETO and NAT fine-tune agents on failed trajectories through contrastive pairs and explicit failure markers, respectively [30, 37], and LEMA fine-tunes on mistake corrections written by a stronger model [2]. These methods rely on outcome rewards or offline revision data, whereas PivotOPD provides dense teacher supervision at the post-mistake states that the student's own mistakes create.

Credit assignment at key turns. Outcome rewards reveal little about which turn caused a failure. GiGPO estimates step-level advantages by grouping actions taken from repeated states across rollouts [3], and process reward models score intermediate steps [14, 36]. PivotRL [47] and OPID [42] concentrate training on the few turns that determine the outcome. Like standard OPD, however, these methods assign credit only to actions that the student samples, whereas PivotOPD also distills recovery actions that the student rarely produces after its own pivotal mistakes.

Interactive imitation learning. Recovery distillation resembles interactive imitation learning, where an expert supervises the learner at the states that the learner itself reaches. DAgger queries the expert at states visited by the learner's policy [25], and HG-DAgger lets a human expert take over when the learner enters unsafe states [10]. Recovery distillation similarly queries the teacher at the post-mistake states produced by the student's own mistakes. It differs in that the teacher only names the recovery action, while the token-level target comes from the student's own hinted distribution.

Privileged information. Learning using privileged information provides extra information during training that is unavailable at test time [32], as in asymmetric actor-critic methods whose critic observes the full state [21]. Recent work on language models conditions a self-teacher on privileged information [50, 20]. PivotOPD uses the named gold or recovery action as the privileged information and trains the student without it.

G Limitations and Open Directions

Replayable environments. Recovery turns after the first are reached by replaying the recorded actions in a copy of the environment (Appendix B), so recovery budgets above $K = 1$ require an environment that reproduces recorded observations exactly. ALFWorld, WebShop, and Search-based QA meet this requirement, and our budget ablation uses up to $K = 3$ on each (Figure 7). Many interactive environments, such as live websites, do not. Extending pivot detection to earlier turns of such long episodes and reaching later recovery turns without exact replay remain open directions.

Dependence on the teacher model. PIVOTOPD trains toward the actions that the teacher model names, so a wrong gold action or recovery action becomes a wrong distillation target. On ALFWorld, pivot detection agrees with the oracle within one turn in 77.8% of failed trajectories on average (Table 3), so in the remaining cases, supervision lands away from the first oracle-labeled pivotal turn. The mismatch check can also flag a reasonable alternative action as pivotal (Appendix B). When the student serves as its own teacher, PIVOTOPD remains the best method in Figure 3, but its ALFWorld success rate falls short of that under the stronger teacher (Table 1). Estimating the reliability of each named action before distilling it could reduce this dependence.

Tuning and scope of the diagnosis. The recovery budget K and the recovery weight w_{rec} interact, and we select both per benchmark on validation (Appendix D.1). Applying PIVOTOPD to a new benchmark therefore requires a sweep over them. Our diagnosis of pivotal mistakes relies on the symbolic oracle of ALFWorld, which the other benchmarks lack, so on those benchmarks only the teacher model estimates where pivotal turns occur. Finally, every agent sees a history window of at most five turns (Table 4). The turns that the agents waste after a pivotal mistake in Section 2 are counted under this window, so part of this waste may come from the limited history.