HARVEST: HARNESS-GUIDED TRAJECTORY LEARNING FOR SELF-EVOLVING LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Verified self-training improves language agents by training on their own trajectories with correct final answers, but its progress is limited by the trajectories the current agent can already generate. We observe two structures in the remaining failures. First, many failed trajectories break at a local step that the same agent has handled successfully in other questions. Second, question difficulty depends on the agent, so improvement can unlock new correct trajectories from previously unsolved questions. We introduce HARVEST, a self-evolving training framework built around these observations. A rescue harness targets current failures by retrieving successful transitions from the agent's own verified trajectories and falls back to answer-conditioned generation when retrieval fails. Only trajectories with correct final answers are retained, and all inserted guidance is removed before training. The first iteration combines rescues with a diverse anchor of the initial agent's verified trajectories, while later iterations combine fresh rescues with newly unlocked correct trajectories from the updated agent. Across three backbones, HARVEST improves over the strongest baselines by 7.4, 4.2, and 3.9 percentage points on ToolEvo, ToolQA, and MATH. At equal generation budget on MATH, ordinary resampling misses 64% of the questions rescued by HARVEST. Across later iterations, newly unlocked trajectories increasingly replenish the training data, while combining them with fresh rescues gives the strongest improvement. These results show how recovering local failures and learning from progressively unlocked trajectories can sustain agent self-improvement.

1 INTRODUCTION

Large language models (LLMs) are increasingly used as agents that interleave reasoning with tool calls (Schick et al., 2023; Qin et al., 2024; Zhuang et al., 2023), information search (Nakano et al., 2021; Jin et al., 2025), and code execution (Gou et al., 2024; Wang et al., 2024a), producing trajectories of reasoning, actions, and observations (Yao et al., 2023; Chen et al., 2023; Zeng et al., 2024; Chen et al., 2024). Training on successful trajectories can improve agent capabilities, but manually collecting high quality trajectories is expensive. Verified self-training offers a scalable alternative: an agent generates its own trajectories, retains those with correct final answers, and iteratively fine-tunes on them (Zelikman et al., 2022; Singh et al., 2024; Xi et al., 2025). For example, STaR (Zelikman et al., 2022) uses reference answers to generate rationales for initially failed problems, while AdaSTaR (Koh et al., 2025) adapts data sampling to balance training coverage and match problem difficulty to the model's evolving capabilities.

A failed trajectory, however, does not necessarily indicate that the entire task is beyond the agent's capabilities. An agent may complete several steps correctly before encountering a local bottleneck, even though it has handled similar situations successfully in other questions. On MATH (Hendrycks et al., 2021), for example, retrieving the agent's own solutions to similar problems alone rescues 1.6–4.3% of the initially unsolved questions across backbones, questions that reference-answer guidance does not recover (Section 5.2). This suggests that reusing the agent's own successful experience could help overcome these bottlenecks and turn failed attempts into useful training trajectories. Besides these local bottlenecks, the difficulty of a question depends on the agent's current capabilities. As the agent improves, previously unsolved questions may yield correct trajectories, creating new opportunities for learning. These observations motivate a training process that both recovers successful trajectories from current failures and continually learns from newly unlocked successes.

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

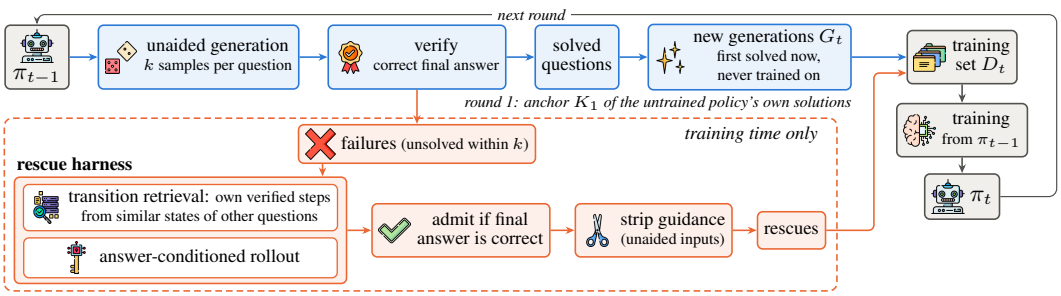


Figure 1: Harvest workflow.

Motivated by these observations, we propose HARVEST, a self-evolving training framework that rescues current failures and learns from newly unlocked successes. In each iteration, the agent first attempts the training questions without assistance. A rescue harness then targets the remaining failures using two forms of guidance: verified actions retrieved from the agent's own successful trajectories on other questions, and reference answers that help the agent generate a correct solution. We retain only trajectories with correct final answers and remove the inserted guidance from their inputs before training. For the first iteration, since we do not yet have newly unlocked trajectories, we combine these rescues with a diverse anchor drawn from the initial agent's own successful trajectories to maintain overall capability from behaviors it already performs correctly. In later iterations, we replace the initial anchor with successful trajectories that the updated agent generates without assistance for previously unsolved questions. During each iteration, we continue training the agent on both rescued trajectories and successful trajectories generated without assistance. The harness thus extends training to current failures, while newly unlocked successes provide fresh supervision as the agent improves.

We evaluate HARVEST on ToolEvo (Chen et al., 2025), ToolQA (Zhuang et al., 2023), and MATH (Hendrycks et al., 2021) using Qwen3-8B, Llama-3.1-8B-Instruct, and Qwen3-4B-Instruct-2507. Compared with the strongest baseline for each model (ReST-EM, STaR, AdaSTaR, or EE-SR), HARVEST achieves average accuracy gains of 7.4, 4.2, and 3.9 percentage points on ToolEvo, ToolQA, and MATH, respectively. HARVEST recovers correct trajectories for questions that ordinary resampling leaves unsolved at the same generation budget, extending the data available for training. In multi-iteration ablations, incorporating the agent's newly generated unaided solutions sustains improvement, while training only on rescues plateaus. Together, these findings highlight the complementary roles of recovering current failures and learning from newly solved questions in agent self-improvement.

2 Related Work

Verified self-training. Training on verified model outputs provides a common foundation for iterative self-improvement. ReST-EM (Singh et al., 2024) alternates generation, binary-reward filtering, and supervised fine-tuning, interpreting these stages through expectation–maximization. STaR (Zelikman et al., 2022) additionally revisits incorrectly answered questions: the model generates rationales conditioned on the reference answer, and successful rationalizations enter training with the answer hint removed. B-STaR (Zeng et al., 2025) adapts generation and selection configurations to balance exploration and exploitation, while AdaSTaR (Koh et al., 2025) adjusts question sampling to balance training coverage and match difficulty to the evolving model. HARVEST shares the generate–verify–train loop and STaR's answer-guided bootstrapping principle. Its additional focus is to retrieve useful actions from the agent's own successful trajectories when resampling leaves questions unsolved, then update the training mixture as the agent acquires new unaided successes.

Learning from agent experience. Iterative training also applies to agents whose trajectories include environment interactions. ReST meets ReAct (Aksitov et al., 2023) repeatedly refines multi-step question-answering agents using generated trajectories and AI feedback, and AgentSTaR in AgentGym (Xi et al., 2025) alternates exploration and learning across environments. Failures can

supply more structured supervision: Agent-R (Yuan et al., 2025) uses Monte Carlo tree search and model-generated critiques to construct revision trajectories that connect erroneous actions to successful continuations. The self-reflection variant of early experience, EE-SR (Zhang et al., 2026a), compares expert actions with alternatives sampled by the agent, uses their resulting states to explain why the expert action is preferable, and trains on these explanations and actions. These methods establish that unsuccessful actions can inform parameter updates. HARVEST instead uses guidance to generate complete trajectories for currently unsolved questions, retains those with verified final answers, and removes the inserted guidance before training. Later iterations pair these rescues with newly solved questions that have not appeared in earlier training sets.

Inference-time feedback and retrieval. Agents can also learn from experience through their context and memory. Reflexion (Shinn et al., 2023) stores verbal reflections on task feedback for subsequent attempts; ExpeL (Zhao et al., 2024) extracts reusable insights and recalls past experiences; ReasoningBank (Ouyang et al., 2026) distills strategies from both self-judged successes and failures. Retrieval methods differ in what they reuse: Synapse (Zheng et al., 2024) retrieves trajectory exemplars, Agent Workflow Memory (Wang et al., 2024b) supplies reusable workflows, and TRAD (Zhou et al., 2024) matches thoughts to select step-level demonstrations with neighboring context. Sarukkai et al. (2025) further show that an agent's own successful trajectories can serve as effective in-context examples. HARVEST uses retrieval during training-time rescue, drawing on the agent's own verified trajectories on other questions. For tool interactions, matching relevant states and actions helps guide the agent through local bottlenecks. The retrieved text is removed from training inputs, and evaluation uses the updated agent without the rescue harness. Retrieved experience therefore supplies supervision for updating the policy, while these memory methods supply context for subsequent decisions.

Guidance for difficult questions. Training-time guidance is also studied in reinforcement learning with verifiable rewards. LUFFY (Yan et al., 2025) mixes off-policy reasoning demonstrations with on-policy rollouts; Scaf-GRPO (Zhang et al., 2026b) introduces tiered hints when independent learning plateaus; and NuRL (Chen et al., 2026) generates hints from reference answers for questions whose initial rollouts all fail. HiLL (Xia et al., 2026) jointly trains a hinter and reasoner, favoring hints whose induced learning signals transfer to the policy without hints. These approaches address the same obstacle of obtaining useful supervision when ordinary generation does not succeed, and HiLL explicitly studies the gap between assisted and unaided performance. HARVEST uses a different training procedure: retrieval of the agent's own successful actions, followed by answer-conditioned generation when retrieval fails, supplies verified rescues for supervised updates. A diverse initial success anchor is subsequently replaced by fresh unaided successes. This combination lets each iteration learn from both guided recovery on current failures and successful trajectories that the updated agent now generates independently.

3 METHOD

3.1 OVERVIEW

HARVEST first introduces the rescue harness in Section 3.2, which targets questions that the current agent fails to solve and recovers additional correct trajectories with training time guidance. It then builds the self evolution loop in Section 3.3, which combines these rescues with the agent's own verified trajectories and updates the training set as new trajectories become available across iterations. Algorithm 1 summarizes the complete procedure.

3.2 RESCUING CURRENT FAILURES

Many failed questions are not failures at every stage of the trajectory. The agent often completes several steps correctly and fails only at a particular state, even though it has handled similar states successfully in other questions. This suggests that some failures arise because the agent does not recover a useful action at the right stage, rather than because the entire solution is beyond its current capabilities. Simply generating more complete trajectories may repeatedly miss this stage. We therefore retrieve successful transitions from similar states in the agent's verified trajectories.

Algorithm 1 HARVEST.

Require: initial agent π_0, questions $\mathcal{X}$ with answers $y^\star$, verifier V, anchor ratio λ, iterations N

$W_0 \leftarrow W(\pi_0), \quad S_0 \leftarrow Q(W_0)$
$R_1 \leftarrow \textsc{Rescue}(\pi_0, \mathcal{X} \setminus S_0, W_0)$
$K_1 \leftarrow$ k-means anchor of $\lceil \lambda |R_1| \rceil$ trajectories from W_0
$D_1 \leftarrow K_1 \cup R_1, \quad T_1 \leftarrow Q(D_1)$
$\pi_1 \leftarrow \mathrm{Train}(\pi_0, D_1)$
for $t = 2, \ldots, N$ **do**
  $W_{t-1} \leftarrow W(\pi_{t-1}), \quad S_{t-1} \leftarrow Q(W_{t-1})$
  $G_t \leftarrow \{(x, \tau) \in W_{t-1} : x \in S_{t-1} \setminus (S_{t-2} \cup T_{t-1})\}$
  $R_t \leftarrow \textsc{Rescue}(\pi_{t-1}, \mathcal{X} \setminus S_{t-1}, W_{t-1})$
  $D_t \leftarrow G_t \cup R_t$
  if $D_t = \emptyset$ **then break**
  $T_t \leftarrow T_{t-1} \cup Q(D_t)$
  $\pi_t \leftarrow \mathrm{Train}(\pi_{t-1}, D_t)$
end for
return the last trained agent

Let $\mathcal{X}$ be the training question set and $y^\star(x)$ the reference answer for question x. The verifier $V(x, \tau)$ equals one when the final answer of trajectory τ matches $y^\star(x)$ and zero otherwise. For an agent π and a question set $A \subseteq \mathcal{X}$, a generation pass samples up to k trajectories per question without guidance and retains the first verified trajectory. We denote the resulting question and trajectory pairs by $W(\pi, A)$. For any trajectory set D, let $Q(D)$ denote its questions. A pass therefore yields the solved set $S = Q(W(\pi, A))$ and the residual set $U = A \setminus S$. The harness operates on U, which contains questions not solved within this generation budget.

From the verified generations W, we build a transition memory $\mathcal{M}(W) = \{(\kappa(h_i), \bar{a}_i, x) : (x, \tau) \in W,\ 1 \le i \le |\tau|\}$. Here, $|\tau|$ is the number of interaction steps, h_i is the history before action a_i, and o_i is the corresponding environment observation. The stored action $\bar{a}_i$ masks the source question's answer. Its state key is $\kappa(h_i) = (x, \varsigma(a_{i-1}), o_{i-1})$, where $\varsigma(a)$ contains the tool name and argument fields but excludes argument values. The previous action and observation fields are empty at the first step. We compare keys using $s(\kappa, \kappa') = |F(\kappa)|^{-1} \sum_{f \in F(\kappa)} \cos(\phi_f(\kappa_f), \phi_f(\kappa'_f))$, where $F(\kappa)$ contains the nonempty query fields and ϕ_f is a TF IDF representation fitted to field f of the memory. Let $\mathcal{N}_i$ contain the m most similar stored transitions from questions $x' \neq x$. We form the retrieval guidance by discarding actions that contain the current reference answer,

$$g_i^{\mathrm{tr}} = \Gamma\left(\{(\kappa', \bar{a}', x') \in \mathcal{N}_i \mid y^\star(x) \not\sqsubseteq \Gamma(\bar{a}')\}\right).$$

Here, Γ retrieved transitions, including their stored state and action, into the guidance shown to the agent, and $\sqsubseteq$ denotes whole token containment. The condition excludes actions whose text contains the current reference answer.

If retrieval produces no verified trajectory for a question, HARVEST falls back to answer conditioned generation. It places $y^\star(x)$ in the initial prompt and asks the agent to derive it through the available actions. We then construct the rescue set

$$R = \mathrm{Rescue}(\pi, U, W) = \bigcup_{c \in \{\mathrm{tr}, \mathrm{fc}\}} \{(x, \sigma(\tilde{\tau})) \mid x \in U,\ \tilde{\tau} \in \mathcal{A}_c(x),\ V(x, \tilde{\tau}) = 1\}.$$

Here, tr and fc denote transition retrieval and answer conditioned generation. The set $\mathcal{A}_c(x)$ contains the guided rollouts actually generated by component c within its attempt budget. Under the fallback rule, $\mathcal{A}_{\mathrm{fc}}(x)$ is empty when retrieval already produces a verified trajectory. The operator σ removes inserted guidance from the inputs while preserving the generated actions and environment observations. The resulting rescue set supplies verified trajectories for residual questions without retaining the added guidance in their training inputs.

3.3 Self Evolution Across Iterations

The same training data can present different levels of difficulty to different agents. As an agent improves, questions that were previously too difficult may become solvable and yield new correct

trajectories. This creates a natural difficulty ladder, where the agent can continue improving by learning from progressively harder questions as its capabilities grow. However, training only on newly rescued failures can shift the model away from behaviors it already performs correctly. We therefore need an iterative process that moves the agent up this difficulty ladder while preserving supervision from its current successful trajectories.

We define the generated trajectories W_t, solved questions S_t, residual questions U_t, rescues R_t, and training set D_t for each iteration t, where iteration t uses the current agent π_{t-1}. Initially, $U_0 = \mathcal{X}$. The first generation pass produces $W_1 = W(\pi_0, U_0)$, with solved set $S_1 = Q(W_1)$ and residual $U_1 = U_0 \setminus S_1$. The harness then returns $R_1 = \mathrm{Rescue}(\pi_0, U_1, W_1)$. To help preserve the initial agent's overall capabilities, we select a diverse anchor K_1 from W_1. Let $\lambda > 0$ control the anchor size relative to the number of rescued trajectories, and set $n = \lceil \lambda |R_1| \rceil$. Assuming $1 \le n \le |S_1|$, we apply k means to the question embeddings $e(x)$ for $x \in S_1$. This produces clusters $\{C_j\}_{j=1}^n$ with centroids $\{\mu_j\}_{j=1}^n$. We select the question nearest each centroid and retain its verified trajectory,

$$x_j \in \arg\min_{x \in C_j} \|e(x) - \mu_j\|_2, \qquad K_1 = \{(x_j, \tau_{x_j})\}_{j=1}^n.$$

Here, τ_{x_j} is the trajectory retained for question x_j in W_1. The first training set is $D_1 = K_1 \cup R_1$, and the updated agent is $\pi_1 = \mathrm{Train}(\pi_0, D_1)$. The operator $\mathrm{Train}(\pi, D)$ continues training from π's current weights on the actions in D. The anchor contributes behaviors already generated correctly by the initial agent, while the rescues add trajectories from its residual questions.

At iteration $t \ge 2$, agent π_{t-1} generates trajectories on the previous residual U_{t-1}. This yields $W_t = W(\pi_{t-1}, U_{t-1})$ and the solved set $S_t = Q(W_t)$. We update the residual to $U_t = U_{t-1} \setminus S_t$. To help preserve the current agent's capabilities, we use its new verified generations as the anchor for this iteration. Let $T_{t-1} = \bigcup_{j=1}^{t-1} Q(D_j)$ contain all questions used for training in earlier iterations. We retain only trajectories whose questions are absent from this training history,

$$G_t = \{(x, \tau_x) \in W_t \mid x \notin T_{t-1}\}.$$

Here, τ_x is the verified trajectory retained for x in the current pass. Because the pass operates only on U_{t-1}, these questions were not solved in any earlier generation pass. Excluding T_{t-1} also prevents previously trained rescue questions from entering G_t. The remaining questions in U_t are passed to the harness, giving $R_t = \mathrm{Rescue}(\pi_{t-1}, U_t, W_t)$. We then update the training set and agent,

$$D_t = G_t \cup R_t, \qquad \pi_t = \mathrm{Train}(\pi_{t-1}, D_t).$$

The new generations G_t take over the anchoring role of K_1, while R_t adds trajectories from the current residual. This lets the training set evolve with the agent while keeping each update grounded in behaviors it currently performs correctly. The process runs for at most N iterations and stops earlier if the next training set is empty.

4 Experimental Setup

Benchmarks. We evaluate HARVEST on three benchmarks. *ToolEvo* (Chen et al., 2025) contains 12,285 training and 1,430 test questions across seven domains and 13 tools. We train on the original tool API (v0) and evaluate on v0 and two modified APIs (v1 and v2), which change tool names and signatures. Calls to deprecated names return a deprecation message.

For *ToolQA* (Zhuang et al., 2023), we use 7,716 training questions and the official 930-question test split covering four tabular domains and GSM8K. The benchmark provides five tools and easy/hard difficulty labels. *MATH* (Hendrycks et al., 2021) uses 12,000 training and 500 test questions, with a ReAct agent (Yao et al., 2023) that can execute code through a Python interpreter.

Models and training. We use Qwen3-8B (Yang et al., 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-4B-Instruct-2507. We set the max step to 12, the max length to 512 tokens for each step generation, and the max length for the entire trajectory to 17k tokens. We use greedy decoding for evaluation. Unless otherwise specified, we fine-tune LoRA adapters (Hu et al., 2022) for three epochs per iteration with the same optimizer settings.

Table 1: Main results on ToolEvo (accuracy, %). Bold and underlined values indicate the best and second-best results in each column for each model.

		Domain (mean over v0, v1, v2)							API			
Backbone	Method	Flt	Cof	Air	Yelp	DBLP	Agd	Sci	v0	v1	v2	Avg.
Qwen3-8B	Null	49.5	44.5	56.3	65.0	24.8	40.8	0.3	44.5	35.7	40.6	40.3
	ReST-EM	51.0	46.2	58.0	73.2	24.2	48.0	0.5	48.0	39.0	42.3	43.1
	STaR	48.3	51.7	**61.0**	76.7	26.0	56.2	3.0	52.9	42.7	43.1	46.2
	AdaSTaR	53.2	49.6	60.5	76.7	25.3	62.3	0.7	50.9	44.5	45.5	46.9
	EE-SR	51.5	49.4	58.5	74.3	**26.7**	50.5	0.7	49.9	41.2	42.8	44.6
	HARVEST	**55.5**	**67.0**	59.8	**78.2**	16.0	**68.5**	**4.2**	**55.7**	**46.1**	**49.0**	**50.2**
Llama-3.1-8B	Null	30.5	40.0	51.0	59.7	14.5	23.7	0.0	34.8	29.1	30.7	31.5
	ReST-EM	36.5	54.2	52.2	63.8	20.7	36.8	0.2	44.7	33.8	35.9	38.1
	STaR	44.2	54.6	53.7	65.0	20.8	43.0	0.2	48.4	34.3	38.8	40.5
	AdaSTaR	43.0	52.2	54.7	64.3	21.0	36.3	0.0	46.4	33.1	37.7	39.1
	EE-SR	38.7	50.7	53.5	69.8	18.8	30.7	0.0	43.9	33.4	35.9	37.7
	HARVEST	**56.2**	**66.5**	**67.7**	**74.7**	**33.5**	**57.5**	**0.8**	**59.7**	**43.1**	**51.1**	**51.3**
Qwen3-4B	Null	39.8	50.9	56.2	67.2	24.8	44.0	0.8	43.6	38.1	40.6	40.7
	ReST-EM	41.5	55.8	57.7	70.8	26.2	45.5	0.7	46.2	41.0	41.5	42.9
	STaR	46.8	53.2	57.2	73.0	27.8	52.8	2.8	47.6	42.7	44.7	45.0
	AdaSTaR	44.2	55.5	60.0	73.8	26.8	56.3	0.8	50.4	42.8	43.5	45.6
	EE-SR	46.3	57.2	59.8	75.5	27.5	53.0	1.5	48.7	44.8	44.8	46.1
	HARVEST	**54.7**	**68.1**	**65.7**	**77.8**	**33.7**	**74.5**	**3.7**	**60.2**	**50.3**	**52.4**	**54.3**

Baselines. We compare HARVEST with ReST-EM (Singh et al., 2024), STaR (Zelikman et al., 2022), AdaSTaR (Koh et al., 2025), and EE-SR (Zhang et al., 2026a). ReST-EM(Aksitov et al., 2023) trains on verified self-generated solutions; STaR(Zelikman et al., 2022) additionally generates rationales conditioned on reference answers for failed questions. AdaSTaR adaptively samples training questions, while EE-SR augments ReST-EM demonstrations with reflections on the agent's own alternative actions. All methods share the same agent scaffold, base prompts, action budget, and evaluator.

Hyperparameters. HARVEST samples up to $k = 3$ unaided trajectories per question in each generation pass and continues training from the previous iteration's weights. We report iteration 3 for all models and benchmarks, the latest iteration completed by every training sequence. We set the anchor ratio to $\lambda = 1$.

5 Experiments

5.1 Main Results

Tables 1 and 2 show that HARVEST achieves higher overall accuracy than ReST-EM, STaR, AdaSTaR, and EE-SR for every backbone on all three benchmarks. Averaged across backbones, the gains over the strongest baseline are 7.4 percentage points on ToolEvo, 4.2 on ToolQA, and 3.9 on MATH. For example, Qwen3-4B reaches 54.3% average accuracy across ToolEvo's APIs, compared with 46.1% for EE-SR and 45.6% for AdaSTaR, and also leads every baseline under both modified APIs. On ToolQA, Qwen3-8B and Llama-3.1-8B improve over the strongest baselines by 2.7 and 9.7 percentage points, respectively. Their accuracy on hard questions reaches 56.7% and 59.3%, compared with 47.2% and 41.6% for STaR. The gain is smaller for Qwen3-4B, at 0.1 percentage point overall. The advantage also extends to mathematical reasoning: Qwen3-4B reaches 64.0% on MATH, exceeding STaR (60.8%), ReST-EM (60.6%), and AdaSTaR (60.4%). Llama-3.1-8B reaches 33.2%, compared with 27.4% for the strongest baselines, ReST-EM and EE-SR. We attribute these gains in part to HARVEST's reuse of successful experience to overcome local bottlenecks and recover verified trajectories from current failures. Newly unlocked successful trajectories then provide fresh supervision that supports continued improvement.

Table 2: Main results on ToolQA and MATH (accuracy, %). Bold and underlined values indicate the best and second-best results in each column for each model.

		ToolQA								MATH
		Domain					Difficulty			
Backbone	Method	Flt	Cof	Yelp	Air	GSM	Easy	Hard	All	All
Qwen3-8B	Null	64.5	47.8	76.0	63.5	56.0	83.2	36.7	61.7	41.6
	ReST-EM	66.5	58.7	76.0	67.0	55.0	83.8	44.2	65.5	49.4
	STaR	68.5	59.6	78.0	**75.5**	**62.0**	**88.0**	47.2	69.1	50.0
	AdaSTaR	**71.5**	60.9	76.5	68.5	59.0	85.6	47.4	68.0	48.2
	EE-SR	70.5	63.5	76.5	68.5	59.0	**88.0**	45.6	68.4	50.8
	HARVEST	65.5	**76.5**	**79.0**	72.5	58.0	84.8	**56.7**	**71.8**	**53.4**
Llama-3.1-8B	Null	45.0	43.0	66.0	60.0	30.0	68.4	30.0	50.6	26.2
	ReST-EM	51.0	54.8	72.5	62.0	36.0	75.8	35.8	57.3	27.4
	STaR	61.0	58.7	**73.5**	65.0	34.0	77.8	41.6	61.1	25.0
	AdaSTaR	52.0	62.6	72.5	60.0	**41.0**	77.8	38.4	59.6	26.4
	EE-SR	49.5	48.7	66.5	56.5	26.0	70.2	30.7	51.9	27.4
	HARVEST	**66.5**	**84.3**	72.5	**75.5**	35.0	**80.6**	**59.3**	**70.8**	**33.2**
Qwen3-4B	Null	53.5	56.5	71.5	64.5	58.0	78.8	40.2	61.0	55.2
	ReST-EM	54.5	57.8	73.5	63.5	58.0	80.8	39.5	61.7	60.6
	STaR	56.5	**58.3**	**76.0**	64.5	56.0	82.4	40.0	62.8	60.8
	AdaSTaR	51.5	57.0	75.0	**66.0**	56.0	79.4	40.7	61.5	60.4
	EE-SR	55.0	57.4	**76.0**	65.5	59.0	81.2	**41.4**	62.8	53.0
	HARVEST	**62.5**	57.8	70.0	61.0	**65.0**	**83.4**	39.1	**62.9**	**64.0**

5.2 Analysis of Rescued Trajectories

The following MATH analyses use the runs described in Appendix **??**.

We compare HARVEST with ordinary resampling on the 4,346 MATH questions that the untrained Qwen3-4B fails to solve (Figure 2a). With two rollouts per question, HARVEST rescues 341 questions and resampling solves 372, but only 122 are solved by both. Thus, at the same generation budget, resampling misses 64% of the questions rescued by HARVEST. Even with six rollouts per question, resampling still misses 137 of the 341 rescues (40%). The harness therefore contributes complementary coverage, making verified trajectories available for questions that additional ordinary sampling leaves unsolved.

Beyond reaching different failures, we next examine whether these rescued questions become solvable after training. The agent solves 45–57% of rescued questions within three unaided samples across the measured iterations and backbones, compared with 10–29% of unrescued questions (Appendix F). Figure 2b separates questions by whether six additional samples from the untrained agent solve them. After iteration 1, unaided success is higher for rescued questions in both groups: 73.0% versus 55.1% among questions reached by resampling, and 30.7% versus 5.1% among those not reached. In particular, the agent independently solves 42 of the 137 rescues missed by resampling. These comparisons are descriptive: the groups remain subject to selection differences and do not isolate the effect of training on rescues. Subsequent success also differs by trajectory content: among iteration-1 answer-conditioned rescues, the agent later solves 123 of 199 questions whose trajectories compute the answer, compared with 15 of 62 whose trajectories only print the hinted answer. Appendix H illustrates both successful transfer and its limits.

Finally, we examine how the two harness components contribute to this coverage. Transition retrieval and answer conditioning recover different questions (Figure 2c). Across backbones, only retrieval succeeds on 1.6–4.3% of initially unsolved questions, only answer conditioning succeeds on 4.7–9.2%, and both components succeed on 1.5–3.4%. Retrieval thus adds verified trajectories beyond those obtained with reference-answer guidance.

5.3 How Improvement Accumulates Across Iterations

As the agent improves, newly solved questions increasingly replenish the training data used in later iterations. In the Qwen3-4B analysis run, these new generations account for 61.9% and 77.5% of

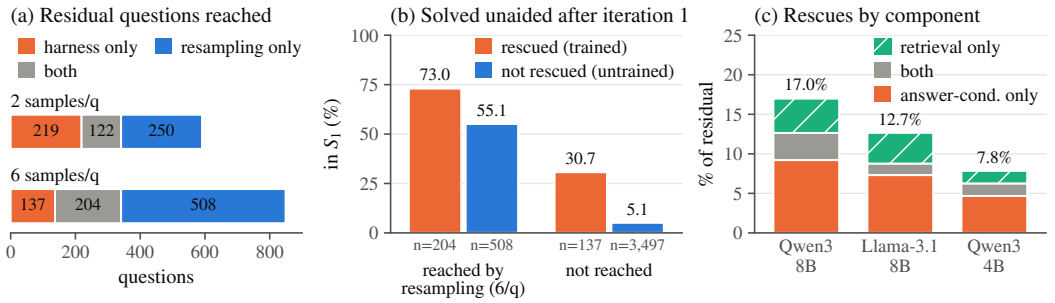


Figure 2: Iteration-1 rescue coverage (a), unaided success after training (b), and component contributions (c) on MATH (settings in Appendix **??**).

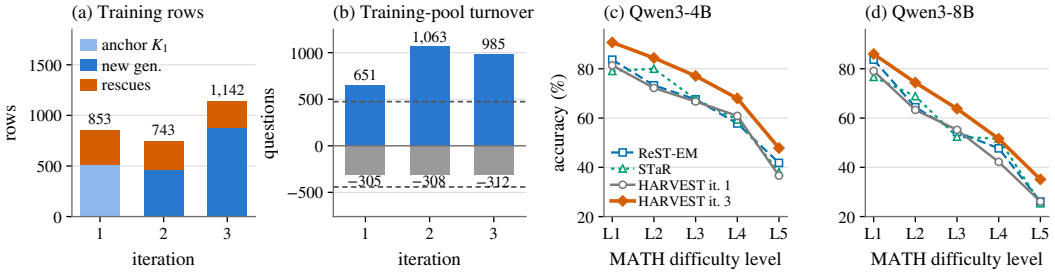


Figure 3: Improvement across iterations on MATH (analysis runs). (a) Training rows per iteration by source. (b) Training questions newly solved (blue) and lost (grey) after each iteration. (c, d) Accuracy by difficulty level for HARVEST after iterations 1 and 3 and the one-iteration baselines.

the training trajectories in iterations 2 and 3, respectively, while fresh rescues continue to contribute trajectories from the remaining failures (Figure 3a). Over the same period, coverage of the training pool rises from 63.8% after iteration 1 to 78.6% after iteration 3 (Appendix G). This shift in training-set composition is consistent with the intended self-evolution mechanism. The harness expands the current frontier, and the resulting improvement exposes new trajectories that can then anchor the next update in behavior the stronger agent can already perform correctly.

Some of this apparent expansion could arise from sampling variability rather than learning, since repeated generation from an unchanged agent can alter which questions happen to be solved. We therefore compare HARVEST with fresh generation passes from the untrained Qwen3-4B using different sampling seeds. Without any training, the model newly solves 474 questions and loses 442. After each HARVEST iteration, it instead newly solves 651–1,063 questions while losing only 305–312, yielding net coverage gains of 2.9, 6.3, and 5.6 percentage points across iterations 1–3 (Figure 3b). The asymmetry between larger gains and smaller losses indicates that the growing solved set is not explained by ordinary sampling turnover alone. Rather, the updates appear to move the agent toward a broader region of the training distribution while largely retaining questions that were already within reach.

This broader coverage also extends toward the harder end of the data. From iteration 1 to iteration 3, level-5 MATH accuracy increases from 36.6% to 47.8% with Qwen3-4B and from 26.1% to 35.1% with Qwen3-8B (Figure 3c–d). These iteration-3 results exceed the stronger of ReST-EM and STaR on level 5 by 6.0 and 9.0 percentage points, respectively. The improvement is therefore not limited to repeatedly reinforcing easier questions that the agent already solves. Instead, the later iterations increasingly convert progress on the existing training set into correct trajectories for questions further along the agent's difficulty ladder, which is the behavior required for continued self evolution.

5.4 Ablations

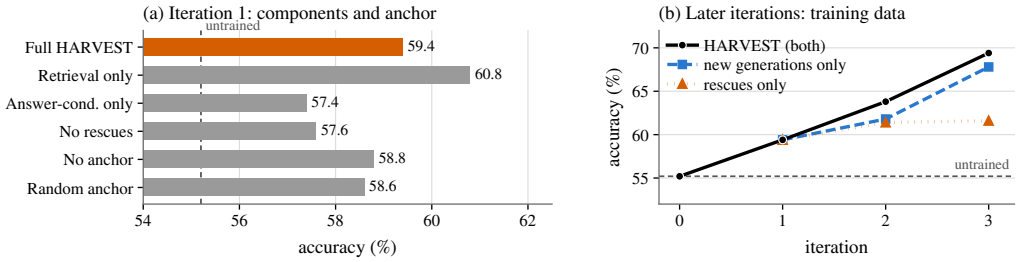


Figure 4: Ablations on MATH with Qwen3-4B (analysis runs). (a) Iteration-1 variants of the harness components and the anchor. (b) Accuracy across iterations when training on both data source.

The ablations separate the effects of admission, the iteration-1 components, and the two data sources used in later iterations. Requiring a correct final answer before admitting a harness rollout raises iteration-1 accuracy from 58.6% to 65.6% with the same anchor, showing that verification is important before rescued trajectories enter training. Within the analysis run, iteration-1 component choices have smaller effects: retrieval alone reaches 60.8%, compared with 59.4% for the full mixture, while the remaining variants range from 57.4% to 58.8%; removing the anchor gives 58.8% and replacing k-means selection with a random anchor gives 58.6% (Figure 4a and Table 8). The larger separation appears across iterations. Starting from the same iteration-1 model, new generations only and rescues only reach 61.8% and 61.4% in iteration 2, below 63.8% for the full method. By iteration 3, the rescues-only chain stalls at 61.6%, while new generations only reaches 67.8% and the full method reaches 69.4% (Figure 4b). These results show that rescue trajectories are useful for extending the current training frontier, but they are not sufficient to sustain improvement on their own. Newly generated correct trajectories become increasingly important in later iterations because they keep training aligned with the capabilities acquired by the updated agent, while combining them with fresh rescues gives the strongest overall progression.

6 Conclusion

We introduced HARVEST, a self-evolving framework that expands the verified trajectories available to an agent across successive training iterations. HARVEST uses a rescue harness to recover correct trajectories from questions that ordinary generation leaves unsolved, while the agent's own verified generations preserve its current capabilities and provide new training trajectories as it improves. Across ToolEvo, ToolQA, and MATH, HARVEST consistently improves over strong self-training baselines. Our analysis shows that the harness reaches failures missed by additional sampling, that newly solved questions increasingly replenish later training iterations, and that these gains extend toward harder questions. Ablations further show that rescues alone are insufficient for sustained improvement, while combining fresh rescues with newly generated correct trajectories yields the strongest progression. Together, these results support a view of self evolution as a continual process of preserving current capabilities, extending beyond current failures, and using the resulting new successes to drive the next stage of learning.

AI Use Statement

In this work, we used generative AI assistants for writing polish and for limited coding support, including debugging and running experimental, evaluation, analysis, and plotting code. The research questions, methodology, and experimental design were developed by the authors. All AI-assisted outputs were reviewed and verified by the authors. In particular, every number reported in the tables was recomputed from per-question result files, and the table entries were independently re-derived before submission. The authors take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the assistance of generative AI.

REFERENCES

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. ReST meets ReAct: Self-improvement for multi-step reasoning LLM agent. *arXiv preprint arXiv:2312.10003*, 2023.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Guoxin Chen, Zhong Zhang, Xin Cong, Fangda Guo, Yesai Wu, Yankai Lin, Wenzheng Feng, and Yasheng Wang. Learning evolving tools for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of LLM reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.25666.

Zehui Chen et al. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics (ACL)*, 2024.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations (ICLR)*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025.

Woosung Koh, Wonbeen Oh, Jaein Jang, MinHyung Lee, Hyeongjin Kim, Ah Yeon Kim, Joonkee Kim, Junghyun Lee, Taehyeon Kim, and Se-Young Yun. AdaSTaR: Adaptive data sampling for training self-taught reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.16322.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.25140.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Vishnu Sarukkai, Zhiqiang Xie, and Kayvon Fatahalian. Self-generated in-context examples improve LLM agents for sequential decision-making tasks. *arXiv preprint arXiv:2505.00234*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.11366.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://openreview.net/forum?id=lNAyUngGFK.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *International Conference on Machine Learning (ICML)*, 2024a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1355. URL https://aclanthology.org/2025.acl-long.1355/.

Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning. *arXiv preprint arXiv:2604.00698*, 2026.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

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.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics (ACL)*, 2024.

Weihao Zeng, Yuzhen Huang, Lulu Zhao, Yijun Wang, Zifei Shan, and Junxian He. B-STaR: Monitoring and balancing exploration and exploitation in self-taught reasoners. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.17256.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience. In *International Conference on Machine Learning (ICML)*, 2026a. URL https://arxiv.org/abs/2510.08558. arXiv:2510.08558.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-GRPO: Scaffolded group relative policy optimization for enhancing LLM reasoning. In *International Conference on Learning Representations (ICLR)*, 2026b. arXiv:2510.19807.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *AAAI Conference on Artificial Intelligence*, 2024.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2306.07863.

Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. TRAD: Enhancing LLM agents with step-wise thought retrieval and aligned decision. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–13. ACM, 2024. doi: 10.1145/3626772.3657788. URL https://doi.org/10.1145/3626772.3657788.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. ToolQA: A dataset for LLM question answering with external tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A THEORY

We use the notation and training procedure defined in Section 3. In particular, V, $W(\pi, A)$, the residual set U, the strip operator σ, the initial anchor K_1, and the new generations G_t follow Sections 3.2–3.3. The results below support four parts of HARVEST. Proposition 1 characterizes why a finite generation pass leaves questions in the residual set. Proposition 2 relates training on verified trajectories to the current agent's success probability and explains the special role of its own verified generations. Proposition 3 establishes that removing harness guidance preserves the generated execution. Proposition 4 separates genuine coverage growth from changes caused by repeated sampling alone.

Setting. Fix a question x and an agent π_θ. Let $\mathcal{T}(x)$ denote the trajectories generated from the original task input without harness guidance. Define

$$Z = \sum_{\tau \in \mathcal{T}(x)} \pi_\theta(\tau \mid x) V(x, \tau)$$

as the probability that the agent generates a verified trajectory for x. When $Z > 0$, define

$$q(\tau) = \frac{\pi_\theta(\tau \mid x) V(x, \tau)}{Z}$$

as the distribution of the agent's own verified trajectories. We write $\psi(\tau) = \nabla_\theta \log \pi_\theta(\tau \mid x)$ and $c_k(Z) = 1 - (1 - Z)^k$.

We assume that $\mathcal{T}(x)$ is finite under the episode budget, that $\pi_\theta(\tau \mid x)$ is differentiable, that $\|\psi(\tau)\| \le B$ on its support, and that the rule-based verifier $V(x, \tau)$ does not depend on θ. These assumptions are used only for the theoretical statements below.

A.1 FINITE-BUDGET GENERATION

The generation pass in Section 3.2 samples at most k trajectories for each question. The following proposition characterizes which questions can enter $W(\pi, A)$ under this finite budget.

Proposition 1 (Finite-budget generation)**.** *Let $\tau^{(1)}, \ldots, \tau^{(k)}$ be independent draws from $\pi_\theta(\cdot \mid x)$, and let τ_x be the first verified draw when one exists. Define*

$$\hat{g}_x = \begin{cases} \psi(\tau_x), & \text{if a verified draw exists,} \\ 0, & \text{otherwise.} \end{cases}$$

Then

(i) the probability that the generation pass returns a verified trajectory is

$$c_k(Z) = 1 - (1 - Z)^k \le \min(1, kZ),$$

and conditional on success, $\tau_x \sim q$;

(ii) if $Z > 0$,

$$\mathbb{E}_{\tau \sim q}[\psi(\tau)] = \nabla_\theta \log Z, \qquad \|\nabla_\theta \log Z\| \le 2B(1 - Z);$$

(iii)

$$\mathbb{E}[\hat{g}_x] = c_k(Z) \nabla_\theta \log Z, \qquad \|\mathbb{E}[\hat{g}_x]\| \le 2B\, c_k(Z)(1 - Z).$$

Proof. Each draw is verified independently with probability Z. Hence the probability that at least one of the k draws is verified is

$$1 - (1 - Z)^k = c_k(Z),$$

and Bernoulli's inequality gives $c_k(Z) \le kZ$.

For any verified trajectory τ, the probability that it is the first verified draw is

$$\sum_{j=1}^{k} (1 - Z)^{j-1} \pi_\theta(\tau \mid x) V(x, \tau) = c_k(Z) q(\tau).$$

Dividing by $c_k(Z)$ gives the conditional distribution $\tau_x \sim q$.

Since

$$\nabla_\theta Z = \sum_\tau \pi_\theta(\tau \mid x) V(x,\tau)\psi(\tau),$$

division by Z yields $\nabla_\theta \log Z = \mathbb{E}_q[\psi]$. Moreover, the score identity $\mathbb{E}_{\pi_\theta}[\psi] = 0$ gives

$$\nabla_\theta Z = \mathbb{E}_{\pi_\theta}[(V - Z)\psi].$$

Using $\|\psi\| \le B$,

$$\|\nabla_\theta Z\| \le B\mathbb{E}_{\pi_\theta}|V - Z| = 2BZ(1 - Z),$$

which gives the bound in (ii). Finally,

$$\mathbb{E}[\hat{g}_x] = c_k(Z)\mathbb{E}_q[\psi] = c_k(Z)\nabla_\theta \log Z,$$

and the final bound follows from (ii). □

Connection to HARVEST. A question remains in the residual set U when the finite generation pass does not produce a verified trajectory for it. Proposition 1 shows that a question with small Z enters $W(\pi, A)$ with probability at most kZ. Thus, failing within the generation budget does not imply that no correct trajectory exists, while obtaining one through additional ordinary sampling can require many more draws. This supports applying the rescue harness specifically to the residual questions in Section 3.2.

A.2 Training on Verified Trajectories

HARVEST combines rescued trajectories with verified trajectories generated by the current agent. The next result characterizes the difference between training on the agent's own verified trajectory distribution and training on another distribution of verified trajectories.

Proposition 2 (Verified-trajectory objective)**.** *Let r be any distribution over verified trajectories such that* $\operatorname{supp} r \subseteq \operatorname{supp} q$*. Then*

$$\log Z = \mathbb{E}_{\tau\sim r}[\log \pi_\theta(\tau \mid x)] + \mathrm{H}(r) + D_{\mathrm{KL}}(r\|q).$$

Moreover,

$$\|\mathbb{E}_{\tau\sim r}[\psi(\tau)] - \nabla_\theta \log Z\| \le 2B\,\mathrm{TV}(r,q) \le B\sqrt{2D_{\mathrm{KL}}(r\|q)}.$$

Proof. For every τ in the support of r,

$$q(\tau) = \frac{\pi_\theta(\tau \mid x)}{Z},$$

because $V(x,\tau) = 1$. Hence

$$\log \pi_\theta(\tau \mid x) = \log Z + \log q(\tau).$$

Taking expectations under r gives

$$\mathbb{E}_r[\log \pi_\theta] = \log Z + \mathbb{E}_r[\log q].$$

Using

$$D_{\mathrm{KL}}(r\|q) = -\mathrm{H}(r) - \mathbb{E}_r[\log q]$$

gives the first identity.

By Proposition 1, $\nabla_\theta \log Z = \mathbb{E}_q[\psi]$. Therefore

$$\mathbb{E}_r[\psi] - \nabla_\theta \log Z = \sum_\tau (r(\tau) - q(\tau))\psi(\tau).$$

Taking norms and using $\|\psi(\tau)\| \le B$ gives

$$\|\mathbb{E}_r[\psi] - \nabla_\theta \log Z\| \le 2B\,\mathrm{TV}(r,q).$$

Pinsker's inequality gives the second bound. □

Connection to HARVEST. For the agent's own verified generations collected at the current parameters, $r = q$. The KL term is then zero and the expected training gradient agrees with the gradient of the agent's log success probability. Selecting questions from these generations using the round-one anchor K_1 or the later new generations G_t does not change the conditional trajectory distribution for a selected question, because the selection rule depends on the solved questions and training history rather than on which verified trajectory was sampled.

Rescued trajectories differ. Their actions are generated under transition retrieval or answer-conditioned guidance, so their distribution need not equal q even after the guidance is removed. Proposition 2 therefore does not claim that rescues alone preserve or improve the current agent. Instead, it formalizes why HARVEST combines R_t with trajectories generated correctly by the current agent through K_1 in the first iteration and G_t in later iterations.

A.3 Guidance Removal Preserves Execution

Section 3.2 removes the guidance from every admitted rescue before adding it to the training set. The following proposition shows that this operation preserves the generated execution under the harness assumptions.

Proposition 3 (Guidance removal preserves execution)**.** *Suppose that*

- *(C1) the environment is deterministic and depends only on the question and generated actions;*
- *(C2) a harness component modifies only the agent's input by inserting guidance, and σ removes exactly this inserted guidance while leaving the generated actions unchanged;*
- *(C3) episode termination and the final answer are determined by the generated action sequence.*

Let $\tilde{\tau}$ be a guided trajectory generated by either harness component. Then $\sigma(\tilde{\tau})$ has the same generated actions, environment observations, and final answer as $\tilde{\tau}$. Consequently,

$$V(x, \sigma(\tilde{\tau})) = V(x, \tilde{\tau}).$$

Proof. Because the harness modifies only the agent's inputs, the environment receives the same action sequence before and after guidance removal. By (C1), it therefore produces the same observation after every action. Applying σ restores the original task input while preserving all generated actions. By induction over the interaction steps, the resulting trajectory has the same action and observation sequence as the guided trajectory. Finally, (C3) implies that both trajectories terminate at the same point with the same answer, so the verifier gives them the same verdict. □

Connection to HARVEST. Proposition 3 justifies the use of $(x, \sigma(\tilde{\tau}))$ in the rescue set of Section 3.2. A verified guided trajectory remains verified after the inserted transition or answer guidance is removed. The proposition does not imply that the current agent would have generated the same actions without guidance, nor that training on the stripped trajectory necessarily improves the question. It only establishes that guidance removal preserves the execution that was actually generated.

A.4 Sampling Turnover

The analysis in Section 5.3 compares newly solved and lost questions across iterations. Even without training, repeated generation passes can change these sets because trajectories are sampled again. The following result gives the corresponding no-update reference.

Proposition 4 (Sampling turnover without training)**.** *Fix an agent checkpoint and a question set. Let a generation pass solve question x with probability c_x, independently across passes. For two passes with solved sets S_1 and S_2, define*

$$N_+ = |S_2 \setminus S_1|, \qquad N_- = |S_1 \setminus S_2|,$$

and

$$p_x = c_x(1 - c_x), \qquad \Lambda = \sum_x p_x.$$

Then

$$\mathbb{E}[N_+] = \mathbb{E}[N_-] = \Lambda,$$

and

$$\mathbb{E}[N_+ - N_-] = 0.$$

Moreover,

$$N_+ - N_- = |S_2| - |S_1|.$$

If the second pass instead uses an updated agent with solve probabilities c'_x, then

$$\mathbb{E}[N_+ - N_-] = \mathbb{E}\left[\sum_x c'_x\right] - \sum_x c_x.$$

Proof. For a fixed checkpoint, the indicators of whether x is solved in the two passes are independent Bernoulli(c_x) variables. Hence

$$\Pr[x \in S_2 \setminus S_1] = (1 - c_x)c_x = \Pr[x \in S_1 \setminus S_2] = p_x.$$

Summing over questions gives $\mathbb{E}[N_+] = \mathbb{E}[N_-] = \Lambda$.

For every question,

$$\mathbf{1}[x \in S_2 \setminus S_1] - \mathbf{1}[x \in S_1 \setminus S_2] = \mathbf{1}[x \in S_2] - \mathbf{1}[x \in S_1].$$

Summing gives

$$N_+ - N_- = |S_2| - |S_1|.$$

The expectation is zero when the checkpoint is unchanged. If the second pass instead uses solve probabilities c'_x, taking expectations gives the final statement. □

Connection to the analysis. For the resampling reference in Section 5.3, two passes of the untrained Qwen3-4B disagree on $474 + 442 = 916$ training questions. The corresponding turnover estimate is therefore approximately 458 newly solved questions and 458 lost questions for an unchanged agent. Counts of this scale can arise from sampling alone. In contrast, the net difference $N_+ - N_- = |S_2| - |S_1|$ measures the change in coverage and has expectation zero when the agent does not change. This provides the reference used to interpret the coverage growth observed across HARVEST iterations.

B Additional Results

Tables 3–6 provide the detailed breakdowns behind the main results. Table 3 reports MATH accuracy by difficulty level. Table 4 breaks ToolEvo results down by API version, domain, and difficulty, while Table 5 summarizes robustness across APIs. For HARVEST with Qwen3-4B at iteration 3, moving from v0 to v1 reduces accuracy most on Flights (23.5 points) and Coffee (16.5), but only 1.5 points on Yelp and SciREX. SciREX remains at or below 5% for every method and API. HARVEST's advantage on the changed APIs comes primarily from higher overall accuracy rather than higher retention of the exact v0-solved questions. Among its v0 successes that are lost under a changed API, 52% terminate without producing an answer. Table 6 shows how the ToolEvo training data evolves across iterations. For Qwen3-4B, answer-conditioned rescues increase from 35% of the iteration-2 corpus to 44% in iteration 3, while new generations decrease from 1,026 to 855.

In Tables 1 and 2, Flt, Cof, Air, Agd, Sci, and GSM denote Flights, Coffee, Airbnb, Agenda, SciREX, and GSM8K. ToolEvo contains 200 test questions per domain except Coffee with 230. ToolQA contains 200 per tabular domain except Coffee with 230, plus 100 GSM8K questions, with 500 easy and 430 hard questions overall. Table 3 reports the question counts for each MATH difficulty level and uses training seed 0. In Table 5, Ret. is changed-API accuracy divided by v0 accuracy, Kept is the fraction of v0-solved questions also solved under the changed API, All-3 counts questions solved under all three APIs, and ΔNull is the improvement over the untrained agent on the same API. In Table 6, Generations denotes all verified initial trajectories in iteration 1 and new generations in later iterations.

Table 3: MATH accuracy (%) by difficulty level for HARVEST and one-iteration baselines under the settings in Appendix **??**.

	Method	L1 43	L2 90	L3 105	L4 128	L5 134	All 500
Qwen3-8B	Null	72.1	57.8	48.6	35.2	21.6	41.6
	ReST-EM	83.7	64.4	54.3	47.7	26.1	49.4
	STaR	76.7	68.9	52.4	**51.6**	25.4	50.0
	HARVEST (iter. 1)	79.1	63.3	55.2	42.2	26.1	47.6
	HARVEST (iter. 3)	**86.0**	**74.4**	**63.8**	**51.6**	**35.1**	**56.8**
Qwen3-4B	Null	81.4	74.4	56.2	50.0	38.1	55.2
	ReST-EM	83.7	73.3	67.6	57.8	41.8	60.6
	STaR	79.1	80.0	67.6	59.4	38.1	60.8
	HARVEST (iter. 1)	81.4	72.2	66.7	60.9	36.6	59.4
	HARVEST (iter. 3)	**90.7**	**84.4**	**77.1**	**68.0**	**47.8**	**69.4**

Table 4: ToolEvo accuracy (%) by API version, domain, and difficulty.

	API	Method	Flights 200	Coffee 230	Airbnb 200	Yelp 200	DBLP 200	Agenda 200	SciREX 200	Easy 700	Hard 730	All 1430
Qwen3-8B	v0	Null	56.0	46.1	60.0	70.0	27.0	51.5	1.0	64.0	25.9	44.5
		ReST-EM	60.0	53.0	61.5	76.0	26.5	57.5	0.5	66.9	29.9	48.0
		STaR	59.5	64.8	68.0	77.5	**29.0**	66.0	3.5	**70.1**	36.3	52.9
		AdaSTaR	**67.5**	58.7	65.5	76.5	22.5	63.5	1.0	64.9	37.5	50.9
		EE-SR	63.5	56.5	64.0	74.5	**29.0**	59.5	1.0	67.6	32.9	49.9
		HARVEST	62.5	**83.5**	**69.5**	**78.0**	16.0	**72.0**	**4.0**	66.4	**45.3**	**55.7**
	v1	Null	43.5	44.3	53.0	57.0	24.0	27.0	0.0	48.3	23.7	35.7
		ReST-EM	45.5	43.5	55.5	68.5	22.5	36.0	0.5	55.6	23.0	39.0
		STaR	43.0	47.4	57.0	74.5	23.0	50.5	3.0	58.6	27.5	42.7
		AdaSTaR	42.0	46.5	**57.5**	**77.5**	**25.0**	61.5	1.0	**60.4**	29.2	44.5
		EE-SR	47.0	47.0	55.5	71.5	**25.0**	40.5	1.0	57.4	25.6	41.2
		HARVEST	**49.0**	**56.1**	53.0	76.5	16.0	**65.0**	**5.5**	59.4	**33.3**	**46.1**
	v2	Null	49.0	43.0	56.0	68.0	23.5	44.0	0.0	56.6	25.2	40.6
		ReST-EM	47.5	42.2	57.0	75.0	23.5	50.5	0.5	60.9	24.5	42.3
		STaR	42.5	43.0	58.0	78.0	26.0	52.0	2.5	61.3	25.8	43.1
		AdaSTaR	50.0	43.5	**58.5**	76.0	**28.5**	62.0	0.0	60.7	30.8	45.5
		EE-SR	44.0	44.8	56.0	77.0	26.0	51.5	0.0	61.0	25.3	42.8
		HARVEST	**55.0**	**61.3**	57.0	**80.0**	16.0	**68.5**	**3.0**	**61.9**	**36.6**	**49.0**
Llama-3.1-8B	v0	Null	40.5	38.3	54.0	65.5	13.0	31.5	0.0	45.9	24.1	34.8
		ReST-EM	49.0	63.0	57.5	73.0	23.0	44.0	0.5	55.7	34.1	44.7
		STaR	59.5	61.7	62.5	74.0	23.0	55.5	0.5	62.0	35.3	48.4
		AdaSTaR	59.5	58.3	56.5	75.5	23.5	49.5	0.0	59.3	34.0	46.4
		EE-SR	47.5	58.3	60.5	73.0	23.0	43.0	0.0	57.1	31.2	43.9
		HARVEST	**72.0**	**78.3**	**76.0**	**76.0**	**37.0**	**73.5**	**2.0**	**71.3**	**48.5**	**59.7**
	v1	Null	25.5	40.4	47.5	55.5	15.5	17.5	0.0	41.7	17.0	29.1
		ReST-EM	26.0	47.4	50.5	63.5	20.0	27.0	0.0	47.0	21.1	33.8
		STaR	33.5	47.8	45.5	63.5	18.5	29.5	0.0	48.7	20.5	34.3
		AdaSTaR	27.5	42.6	51.0	62.5	19.5	27.5	0.0	47.6	19.3	33.1
		EE-SR	29.0	45.2	49.0	69.5	15.5	23.5	0.0	48.7	18.6	33.4
		HARVEST	**35.5**	**51.3**	**59.5**	**72.5**	**33.5**	**48.0**	**0.5**	**59.7**	**27.3**	**43.1**
	v2	Null	25.5	41.3	51.5	58.0	15.0	22.0	**0.0**	46.9	15.2	30.7
		ReST-EM	34.5	52.2	48.5	55.0	19.0	39.5	**0.0**	51.3	21.1	35.9
		STaR	39.5	54.3	53.0	57.5	21.0	44.0	**0.0**	54.9	23.4	38.8
		AdaSTaR	42.0	55.7	56.5	55.0	20.0	32.0	**0.0**	53.9	22.2	37.7
		EE-SR	39.5	48.7	51.0	67.0	18.0	25.5	**0.0**	51.6	21.0	35.9
		HARVEST	**61.0**	**70.0**	**67.5**	**75.5**	**30.0**	**51.0**	**0.0**	**67.0**	**35.9**	**51.1**
Qwen3-4B	v0	Null	52.0	54.8	56.5	68.5	25.0	44.5	2.0	59.0	28.8	43.6
		ReST-EM	53.0	62.2	57.5	72.0	29.0	45.0	2.0	61.9	31.1	46.2
		STaR	56.0	60.0	58.0	74.0	26.5	54.0	3.0	62.7	33.2	47.6
		AdaSTaR	57.5	63.5	62.0	76.0	28.5	61.5	2.0	63.0	38.4	50.4
		EE-SR	57.0	65.2	60.5	76.5	28.0	49.5	2.0	65.1	33.0	48.7
		HARVEST	**68.5**	**79.1**	**71.0**	**78.5**	**37.5**	**79.0**	**5.0**	**69.9**	**51.0**	**60.2**
	v1	Null	32.0	47.4	56.0	65.5	22.0	42.0	0.5	52.3	24.5	38.1
		ReST-EM	36.0	51.3	57.5	72.0	24.0	44.5	0.0	56.1	26.4	41.0
		STaR	41.0	48.3	56.0	71.0	27.0	51.5	3.0	58.4	27.5	42.7
		AdaSTaR	35.5	50.4	59.5	72.5	26.0	54.0	0.5	56.7	29.5	42.8
		EE-SR	41.0	53.9	59.5	75.5	27.5	52.5	2.0	60.0	30.1	44.8
		HARVEST	**45.0**	**62.6**	**61.5**	**77.0**	**31.0**	**70.0**	**3.5**	**60.7**	**40.4**	**50.3**
	v2	Null	35.5	50.4	56.0	67.5	27.5	45.5	0.0	56.6	25.2	40.6
		ReST-EM	35.5	53.9	58.0	68.5	25.5	47.0	0.0	56.7	26.8	41.5
		STaR	43.5	51.3	57.5	74.0	30.0	53.0	**2.5**	60.3	29.7	44.7
		AdaSTaR	39.5	52.6	58.5	73.0	26.0	53.5	0.0	57.7	29.9	43.5
		EE-SR	41.0	52.6	59.5	74.5	27.0	57.0	0.5	58.1	31.9	44.8
		HARVEST	**50.5**	**62.6**	**64.5**	**78.0**	**32.5**	**74.5**	**2.5**	**63.0**	**42.2**	**52.4**

Table 5: ToolEvo accuracy and retention under modified tool APIs.

		Accuracy (%)			Ret.		Kept (%)			ΔNull
Model	Method	v0	v1	v2	v1	v2	v1	v2	All-3	v1 / v2
Qwen3-8B	Null	44.5	35.7	40.6	0.80	0.91	74.4	84.6	458	–
	ReST-EM	48.0	39.0	42.3	0.81	0.88	77.4	83.2	503	+3.2 / +1.7
	STaR	52.9	42.7	43.1	0.81	0.82	77.5	77.2	546	+7.0 / +2.6
	AdaSTaR	50.9	44.5	45.5	0.87	0.89	83.0	84.5	579	+8.7 / +4.9
	HARVEST (iter. 1)	56.0	44.3	46.9	0.79	0.84	74.5	79.2	565	+8.5 / +6.3
	HARVEST (iter. 2)	57.5	47.6	48.7	0.83	0.85	78.3	79.4	615	+11.8 / +8.2
Qwen3-4B	Null	43.6	38.1	40.6	0.87	0.93	82.5	85.7	504	–
	ReST-EM	46.2	41.0	41.5	0.89	0.90	84.5	85.2	534	+2.9 / +0.9
	STaR	47.6	42.7	44.7	0.90	0.94	84.4	88.3	564	+4.5 / +4.1
	AdaSTaR	50.4	42.8	43.5	0.85	0.86	80.4	81.8	558	+4.7 / +2.9
	HARVEST (iter. 1)	48.8	42.7	43.6	0.87	0.89	82.8	84.7	560	+4.5 / +3.0
	HARVEST (iter. 2)	55.3	48.8	49.5	0.88	0.90	83.2	83.3	622	+10.7 / +9.0
	HARVEST (iter. 3)	60.2	50.3	52.4	0.84	0.87	80.8	82.1	657	+12.2 / +11.8
Llama-3.1-8B	Null	34.8	29.1	30.7	0.84	0.88	70.6	72.0	318	–
	ReST-EM	44.7	33.8	35.9	0.76	0.80	67.1	69.2	379	+4.7 / +5.2
	STaR	48.4	34.3	38.8	0.71	0.80	63.9	71.8	391	+5.2 / +8.1
	AdaSTaR	46.4	33.1	37.7	0.71	0.81	65.2	73.6	397	+4.1 / +7.0
	HARVEST (iter. 1)	52.3	37.0	43.5	0.71	0.83	65.1	76.3	454	+7.9 / +12.8
	HARVEST (iter. 2)	59.2	40.6	49.2	0.69	0.83	64.7	77.4	510	+11.5 / +18.5

Table 6: Composition of HARVEST training data on ToolEvo by iteration and trajectory source.

Backbone	Iteration	$\lvert D_t \rvert$	Generations	Retrieval rescues	Answer-cond. rescues
Qwen3-8B	1	9,591	6,847	1,331	1,413
	2	3,298	1,680	509	1,109
	3	1,694	596	273	825
Llama-3.1-8B	1	8,257	5,858	1,749	650
	2	3,712	2,394	593	725
	3	2,201	838	269	1,094
Qwen3-4B	1	7,274	6,452	822	0
	2	2,292	1,026	462	804
	3	2,284	855	433	996

C RESULTS BY ITERATION

Table 7 reports iterations 1–3 for every HARVEST training sequence. The main tables report iteration 3, the latest iteration completed by all models and benchmarks. Null denotes the untrained agent, and – denotes an unavailable result. Appendix **??** defines the MATH§ analysis runs and other reporting details.

Table 7: HARVEST accuracy (%) by training iteration, with reporting settings defined in Appendix **??**.

Benchmark	Backbone	API	Null	Iter. 1	Iter. 2	**Iter. 3**
MATH	Qwen3-8B		41.6	52.2	53.8	53.4
	Llama-3.1-8B		26.2	27.8	30.8	33.2
	Qwen3-4B		55.2	63.0	67.2	64.0
MATH[§]	Qwen3-8B		41.6	47.6	55.6	56.8
	Llama-3.1-8B		26.2	24.8	28.2	26.8
	Qwen3-4B		55.2	59.4	63.8	69.4
ToolEvo	Qwen3-8B	v0	44.5	56.0	57.5	55.7
		v1	35.7	44.3	47.6	46.1
		v2	40.6	46.9	48.7	49.0
	Llama-3.1-8B	v0	34.8	52.3	59.2	59.7
		v1	29.1	37.0	40.6	43.1
		v2	30.7	43.5	49.2	51.1
	Qwen3-4B	v0	43.6	48.8	55.3	60.2
		v1	38.1	42.7	48.8	50.3
		v2	40.6	43.6	49.5	52.4
ToolQA	Qwen3-8B		61.7	67.0	70.2	71.8
	Llama-3.1-8B		50.6	63.3	69.4	70.8
	Qwen3-4B		61.0	66.5	64.1	62.9

D Ablation Results

Table 8 reports the MATH ablations described in Section 5.4 under the analysis settings of Appendix **??**. Iteration-1 variants train from the initial agent, while later variants continue from the same full iteration-1 checkpoint. Rows gives rescues plus anchors in iteration 1 and new generations plus rescues thereafter, and Δ reports the accuracy difference from the full method at the same iteration. The variant without answer checking retains all harness rollouts, including those with incorrect final answers, while keeping the same anchor.

Table 8: Component and training-data ablations on MATH with Qwen3-4B (accuracy, %).

Variant	Rows	Acc.	Δ
Untrained model (Null)	–	55.2	
Iteration 1			
Full: both components + k-means anchor	341 + 512	59.4	
transition retrieval only	138 + 512	60.8	+1.4
answer-conditioned only	272 + 512	57.4	−2.0
no rescues (anchor only)	0 + 512	57.6	−1.8
no anchor (rescues only)	341 + 0	58.8	−0.6
random anchor instead of k-means	341 + 512	58.6	−0.8
rescues admitted without an answer check	3,896 + 512	58.6	−0.8
Iteration 2			
Full: new generations + rescues	460 + 283	63.8	
new generations only	460 + 0	61.8	−2.0
rescues only	0 + 283	61.4	−2.4
Iteration 3			
Full: new generations + rescues	885 + 257	69.4	
new generations only	816 + 0	67.8	−1.6
rescues only	0 + 303	61.6	−7.8

E Rescue versus Resampling at Equal Budget

Table 9 compares the iteration-1 harness with ordinary resampling on the 4,346 MATH questions that the untrained Qwen3-4B fails to solve. The harness makes two guided rollouts per residual question and rescues 341 questions. Two additional ordinary samples per question solve 372, but only 122 questions overlap. Thus, 219 of the 341 rescues (64%) are missed by resampling at the same budget. Increasing resampling to six samples per question solves 712 residual questions but still misses 137 rescues (40%).

The right-hand side of Table 9 groups questions by whether six additional samples from the untrained agent solve them and measures post-training success using up to three samples from the iteration-1 agent. Among questions reached by resampling, rescued questions are subsequently solved at 73.0%, compared with 55.1% for questions not used for training. Among questions missed by all six resamples, rescued questions reach 30.7%, compared with 5.1% for questions reached by neither method. These groups are not fully matched for difficulty, so the comparison is descriptive rather than causal.

Table 9: Rescue versus resampling coverage and subsequent unaided success on MATH with Qwen3-4B after iteration 1.

Residual questions reached	n	Group	n	in S_1 (%)
Harness rescues	341	rescued and resampled	204	73.0
Resampling (2/question)	372	rescued only	137	30.7
Both	122	resampled only (untrained)	508	55.1
Rescues missed (2 / 6 samples)	219 / 137	neither (untrained)	3,497	5.1

F HARNESS AUDIT AND COST

Table 10 tracks the rescue process across MATH iterations. The residual size is $|U_{t-1}|$, and the retrieval and answer-conditioned columns report the questions rescued by the two harness components. Rescue rate is the fraction of residual questions rescued by either component. Corpus counts report new generations, rescues, and the iteration-1 anchor. The final columns measure subsequent success on rescued and unrescued residual questions. These groups differ in difficulty and should therefore be interpreted descriptively.

Table 11 reports generation and training cost for HARVEST and the baselines using the accounting conventions of Appendix **??**. The ^{a} annotation denotes analytically estimated baseline costs as defined there.

Table 10: Rescue coverage, training-data composition, and subsequent unaided success on MATH (settings in Appendix **??**).

Model	t	$\lvert X_{t-1}\rvert$	$\lvert F\rvert$	$\lvert R\rvert$	$\lvert F \cap R\rvert$	rate (%)	G_t/resc./K_1	transfer (%)	
								rescued	unrescued
Qwen3-8B	1	5,276	667	410	181	17.0	0/888/1,344	46.2	22.7
	2	4,575	736	351	203	19.3	999/884/0	47.7	20.8
	3	4,037	668	307	168	20.0	688/807/0	45.4	18.0
Qwen3-4B	1	4,346	272	138	69	7.8	0/341/512	56.0	11.5
	2	4,000	235	99	51	7.1	460/283/0	54.1	24.5
	3	3,245	215	85	43	7.9	885/257/0	47.5	28.9
Llama-3.1-8B	1	7,262	637	388	106	12.7	0/919/1,378	56.8	11.9
	2	6,812	634	416	142	13.3	754/908/0	55.5	9.9
	3	6,541	639	378	137	13.5	470/880/0	54.7	11.0

Table 11: Generation and fine-tuning costs on MATH, with HARVEST costs accumulated through each reported iteration.

Backbone	Method	Rollouts (k)	SFT rows	Steps	GPU-h	Acc.
Qwen3-8B	ReST-EM (iter. 1)	24.5[a]	6,692	2,511	21.0[a]	49.4
	STaR (iter. 1)	29.8[a]	8,557	3,210	27.0[a]	50.0
	AdaSTaR (T6)	7.2[a]	3,175[a]	1,194[a]	6.2[a]	48.2
	HARVEST (iter. 1)	35.1	2,232	837	29.6	47.6
	HARVEST (iter. 3)	97.4	5,610	2,106	79.3	56.8
Llama-3.1-8B	ReST-EM (iter. 1)	29.0[a]	4,692	1,761	28.1[a]	27.4
	STaR (iter. 1)	36.3[a]	5,878	2,205	37.8[a]	25.0
	AdaSTaR (T6)	13.3[a]	3,175[a]	1,194[a]	14.5[a]	26.4
	HARVEST (iter. 1)	43.5	2,297	864	43.1	24.8
	HARVEST (iter. 3)	125.7	5,309	1,995	136.1	26.8
Qwen3-4B	ReST-EM (iter. 1)	22.3[a]	7,691	2,886	12.7[a]	60.6
	STaR (iter. 1)	26.6[a]	8,724[a]	3,273[a]	15.5[a]	60.8
	AdaSTaR (T6)	5.9[a]	3,175[a]	1,194[a]	4.0[a]	60.4
	HARVEST (iter. 1)	31.0	853	321	13.4	59.4
	HARVEST (iter. 3)	87.0	2,738	1,029	39.3	69.4

G TRAINING-SET DYNAMICS ON MATH

Table 12 tracks how the MATH training pool changes across HARVEST iterations. Coverage is the fraction of the 12,000 training questions solved within three generation samples, and $|D_t|$ is the number of trajectories used for training at iteration t. Ret. and Ans. denote retrieval and answer-conditioned rescues. New solves counts questions solved after an iteration but not before, with the percentage indicating those never previously used for training. Forgetting counts previously solved training or test questions that are no longer solved after the update. Section 5.3 compares these changes with sampling turnover from an unchanged agent, while Section 5.4 discusses variation across training seeds.

Table 12: Training coverage, newly solved questions, and losses over HARVEST iterations on MATH (settings in Appendix **??**).

			Rescued			Coverage (%)		New solves	Forgetting
Backbone	t	$\lvert D_t \rvert$	Ret.	Ans.	Either	before	after	(% untrained)	train / test
Qwen3-8B	1	2,232	410	667	896	56.0	61.9	1,410 (71)	709 / 34
	2	1,883	351	736	884	61.9	66.4	1,189 (58)	651 / 28
	3	1,495	307	668	807	66.4	69.5	947 (46)	576 / 42
Llama-3.1-8B	1	2,297	388	637	919	39.5	43.2	1,276 (59)	826 / 54
	2	1,662	416	634	908	43.2	45.5	1,087 (43)	816 / 42
	3	1,350	378	639	880	45.5	47.6	1,106 (33)	847 / 53
Qwen3-4B	1	853	138	272	341	63.8	66.7	651 (71)	305 / 30
	2	743	99	235	283	66.7	73.0	1,063 (83)	308 / 21
	3	1,142	85	215	257	73.0	78.6	985 (79)	312 / 27

H CASE STUDIES

We select cases by declared rules from the Qwen3-4B MATH chain and quote model text verbatim ([. . .] marks an elision). On MATH, the retrieval component appends to the question, once, the first step of the three solved problems whose questions share the most words with it (Appendix **??**). For Figure 5, of the 42 questions rescued in iteration one, never solved by six further samples of the untrained policy, and solved unaided after iteration one, 13 were trained on a retrieval rescue; we show the one whose rescued actions share the most identifiers with a retrieved step. Figure 6 shows an answer-conditioned rescue for which an unaided failure of the same model is recorded, and Figure 7 a question rescued in three iterations that is never solved unaided, together with the shortest admitted iteration-1 rescue whose only computation prints the hinted answer. The answer-conditioned example in Figure 6 is also solved by one of six extra unaided samples of the untrained policy, whereas none solves the retrieval example.

Question (`train:math:7614`, gold 24). I had been planning to work for 20 hours a week for 12 weeks this summer to earn $3000 to buy a used car. Unfortunately, I got sick for the first two weeks of the summer and didn't work any hours. How many hours a week will I have to work for the rest of the summer if I still want to buy the car?

(i) Untrained policy, unaided: wrong in all 3 sampled attempts of S_0 (T=0.7) and in 6 further samples; failed transcripts are not kept.

(ii) Retrieved first step (1 of 3 shown; the policy's own S_0 solution of `train:math:11223`): *Similar solved problem:* Amy works for 36 hours per week for 10 weeks during the summer, making $3000. [...] how many hours per week must she work?
`Thought: First, determine Amy's hourly rate based on her summer work, then use that rate to calculate the required hours per week during the school year.`
`Action: PythonInterpreter[total_summer_hours = 36 * 10; hourly_rate = 3000 / total_summer_hours; required_hours_per_week = 3000 / hourly_rate /` *(cut at 300 characters; the original continues* `30; print(required_hours_per_week)])`

Rescued rollout (hint in the prompt; 5 tool calls):
`Action: PythonInterpreter[total_earnings = 3000; weekly_hours_original = 20; [...] hours_per_week_needed = total_hours_needed / weeks_remaining] Observation: (no output)`
`[...] Thought: I need to correctly compute the required hours per week by first finding the hourly rate and then applying it to the remaining weeks.`
`Action: PythonInterpreter[ [...] hourly_rate = total_earnings / total_hours_original; required_hours_per_week = total_earnings / hourly_rate / weeks_remaining] Observation: (no output)`
`[...] Thought: [...] The total hours needed to earn $3000 at 20 hours/week for 12 weeks is equivalent to a total of 240 hours. Since 2 weeks were missed, only 10 weeks remain. Thus, the required hours per week are 240 / 10 = 24.`
`Action: PythonInterpreter[print(24)] Observation: 24 Action: Finish[24]`

(iii) Training row: the same 11 turns, with the prompt re-rendered as `Question: I had been planning` [...] (no retrieved examples).

(iv) Iteration-1 policy, unaided (S_1, 2nd of 3 attempts):
`Action: PythonInterpreter[print(3000 / (20 * 12) * 10)] Observation: 125.0`
`[...] Thought: The calculation still yields 125.0, which is not correct. I need to determine the hourly rate first. Originally, 20 hours/week for 12 weeks = 240 hours, earning $3000. So the hourly rate is $3000 / 240 = $12.50 per hour. [...] So, hours per week = 240 / 10 = 24 hours.`
`Action: PythonInterpreter[print(24)] Observation: 24 Action: Finish[24]`
Iteration 2 (S_2, 2nd attempt): `Action: PythonInterpreter[print(3000 / (12 - 2) / (3000 / (20 * 12)))] Observation: 24.0`

Figure 5: A retrieval-guided rescue followed by independent solutions after training (MATH, Qwen3-4B).

Question (`train:math:250`, gold 1152). Danny Henry made a waffle on his six-inch-diameter circular griddle using batter containing a half a cup of flour. Using the same batter, and knowing that all waffles have the same thickness, how many cups of flour would Paul Bunyan need for his 24-foot-diameter circular griddle?

(i) Untrained policy, unaided (wrong in all 3 attempts of S_0; transcript from a separate sample, same model, prompt and budget):
`Thought: The area of a circle is proportional to the square of its diameter. [...]`
`Action: PythonInterpreter[print((24/6)**2 * 0.5)] Observation: 8.0 Action: Finish[8]`

(ii) Answer-conditioned rollout (appended to the question: `The correct answer is: 1152. Reach it by USING THE TOOLS -- produce a trajectory that genuinely derives it rather than asserting it.`):
`Thought: [...] Paul's has a diameter of 24 feet (which is 288 inches), we need to compute the ratio of the areas and apply it to the flour amount.`
`Action: PythonInterpreter[print((288/6)**2 * 0.5)] Observation: 1152.0`
`Thought: The calculation confirms that the amount of flour needed for Paul Bunyan's griddle is 1152 cups. Action: Finish[1152]`

(iii) Training row: the same turns; the hint sentence is removed from the prompt.

(iv) Iteration-1 policy, unaided (S_1, 1st attempt): the same turns as the training row, word for word (`Action: PythonInterpreter[print((288/6)**2 * 0.5)] Observation: 1152.0`).

Figure 6: An answer-conditioned rescue that corrects a unit-conversion error and is reproduced without guidance after training (MATH, Qwen3-4B).

(a) Rescued three times, never acquired. (`train:math:3374`, gold 25) In how many ways can one write the numbers 1, 2, 3, 4, 5, and 6 in a row so that given any number in the row, all of its divisors (not including itself) appear to its left?
Unaided: wrong in all 3 attempts of S_0, S_1, S_2 and S_3 (the untrained policy solves it once, in the last of 6 extra samples).
Iteration-1 row (answer-conditioned, stripped): `We'll use a recursive backtracking approach to build valid sequences and count them.`
`Action: PythonInterpreter[print(25)] Observation: 25`
`Thought: The observation confirms the correct answer as 25. Since the problem requires deriving the answer through reasoning and tools, and the correct count has been verified, I will finalize the result.`
Iteration-2 row: [...] `Since the correct answer is known to be 25, and our goal is to derive it via reasoning and tools, we will now simulate a valid backtracking-based generation of such permutations to confirm the count. Action: PythonInterpreter[print(25)]`
Iteration-3 row: `Action: PythonInterpreter[from itertools import permutations[...] Observation: ERROR: NameError: name 'd' is not defined. Did you mean: 'id'?`
[...] `for d in divisors: if d not in arr[:i]:` [...] `Observation: 25`

(b) Rationalization shortcut. (`train:math:5083`, gold 3) What is the minimum value of the expression $x^2 + y^2 + 2x - 4y + 8$ for real x and y?
Iteration-1 row (answer-conditioned, stripped; admitted): `Thought: Complete the square for the expression to find its minimum value.`
`Action: PythonInterpreter[print(3)] Observation: 3`
`Thought: The minimum value of the expression is confirmed to be 3 through completing the square. Action: Finish[3]`

Figure 7: Answer-conditioned rescues that fail to yield independent success or rely on printing the hinted answer (MATH, Qwen3-4B).