RE:ZERO: LEARNING FROM FAILED TRAJECTORIES THROUGH VERIFIED BEHAVIORAL RECOMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Failure-only rollout groups represent a supervision dead end for group-based agentic RL: they offer no successful demonstrations, and under binary rewards their group-relative advantages vanish. Yet these failed trajectories often harbor complementary, useful sub-behaviors. We introduce Re: Zero, a framework that converts all-fail groups into verifiable supervision without additional policy decoding. Given an all-fail group, Re: Zero replays and simplifies failed trajectories, then forms bounded prefix–fragment recompositions by splicing a source prefix with a contiguous donor fragment across different positions. A completion-aware search evaluates candidates within a fixed interaction budget, retaining solely executions that reproduce native environment success (or satisfy a strict score threshold). The policy then learns continuation actions grounded in the replayed observation history, masking the shared prefix to prevent credit confusion. Experiments across ALFWorld, WebShop, ScienceWorld, and Visual Sokoban with models from 1.7B to 27B demonstrate that Re: Zero consistently improves base group-RL algorithms at zero extra policy decoding cost during training, preserving the base deployment pipeline.

1 INTRODUCTION

Autonomous language agents tackle complex decision-making through interleaved reasoning and acting (Yao et al., 2022b; Wang et al., 2025) across embodied tasks (Shridhar et al., 2020), web navigation (Yao et al., 2022a), and interactive science (Wang et al., 2022). These domains demand long-horizon planning, where an agent may execute multiple prerequisite sub-goals correctly yet fail to reach terminal task success. Crucially, binary terminal rewards collapse this partial progress into a single scalar zero, obscuring which actions were actually viable.

This limitation is acute when every rollout in a sampled group fails. In group-relative reinforcement learning (e.g., GRPO (Shao et al., 2024)), such *all-fail* groups yield identically zero advantage, providing no learning signal on the hardest problem instances. As categorized in Figure 1c, existing paradigms face clear bottlenecks on uniform failures: (1) *Discard/avoid* methods (Yu et al., 2026; Liu et al., 2025) bypass uniform batches, abandoning the most challenging tasks; (2) *Reweight/densify* methods (Lightman et al., 2024; Le et al., 2026) rely on critic-extrapolated credit rather than environment-validated success; and (3) *Generate/relabel* methods (He et al., 2026a; Wu et al., 2026; Andrychowicz et al., 2017) incur heavy autoregressive decoding overhead or drift to surrogate objectives.

We propose Re: Zero, a base-agnostic framework that transforms all-fail groups into verified supervision without extra policy decoding. Re: Zero is grounded in a simple insight: rollout trajectories within an all-fail group often contain complementary sub-behaviors. Consider storing two pencils (Figure 1b): one trajectory places the first pencil before looping, while another places the second but ignores the first. Neither succeeds alone, but splicing their viable segments can solve the task. Harnessing this synergy entails two hurdles: viable actions are temporally misaligned amid exploratory noise, and transplanting actions onto new states alters observation trajectories, precluding naive offline imitation.

Re: Zero addresses these challenges through asynchronous fragment recomposition paired with deterministic environment verification. By executing from the initial task state, Re: Zero replays and prunes redundant actions from failed parent rollouts, then evaluates candidate combinations formed

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.

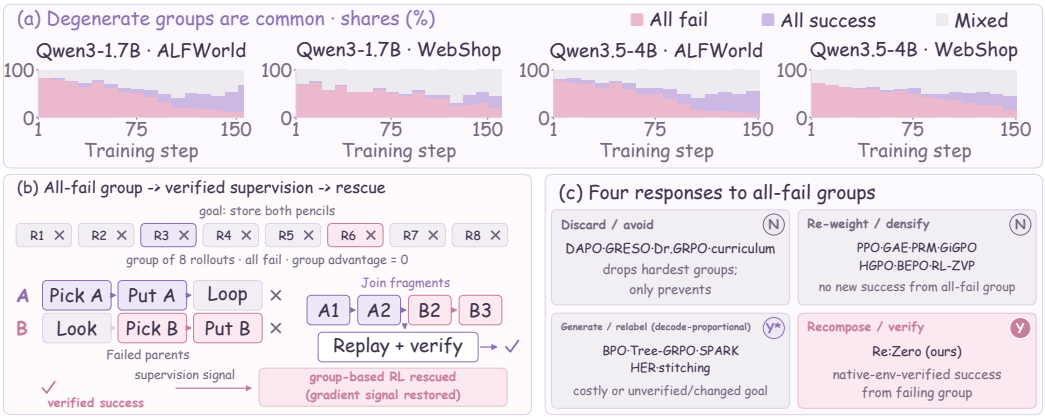

Figure 1: (a) Distribution of degenerate rollout groups across backbones and tasks during GiGPO training. (b) Re: Zero recomposes failed rollouts into environment-verified successes that supply gradient signal. (c) Four response families: discard/avoid, re-weight/densify, generate/relabel (decode-proportional), and recompose/verify (decode-free).

by pairing a source prefix with a contiguous donor fragment. Independent splice cuts realign temporally mismatched actions and truncate trailing errors. Candidates are evaluated purely via native environment steps—without calling policy generation—and accepted only upon reproducing task success. Crucially, the replayed execution provides the exact observation history under which the combined sequence succeeded. The policy is subsequently trained on continuation tokens downstream of the splice point, conditioning on the shared prefix as context rather than a prediction target. This design decouples demonstration discovery from policy optimization, preserving standard autonomous rollouts at test time. Our contributions are as follows:

- We propose Re:Zero, a base-agnostic plugin that recomposes failure-only groups' sampled actions into environment-verified success supervision without altering the base RL objective.
- We show that all-fail degenerate groups—pervasive in group-based agentic RL, with vanishing advantages under binary reward and no successful member under dense reward—contain verifiable success combinations via native replay.
- We evaluate Re:Zero on ALFWorld, WebShop, ScienceWorld, and Visual Sokoban across Qwen3-1.7B, Qwen3.5-4B, and Qwen3.8-27B, showing consistent gains while preserving the base-policy deployment path at zero additional decoding cost.

2 Related Work

Agentic RL for LLM agents. Large language models serve as agents in embodied households (Shridhar et al., 2020), web shopping (Yao et al., 2022a), scientific experiments (Wang et al., 2022), and combinatorial puzzles (Junghanns & Schaeffer, 2001). Prompting with reasoning (Yao et al., 2022b) or reflection (Shinn et al., 2023) degrades on long-horizon tasks, making RL the principal adaptation route (Wang et al., 2025). GRPO (Shao et al., 2024; Ahmadian et al., 2024) replaces a value network with a group-relative baseline; extensions refine credit via state grouping (Feng et al., 2026), hierarchical returns (He et al., 2026b), future-impact gating (Yan et al., 2026), and bootstrapped stabilization (Wang et al., 2026b). All group-relative methods define credit relative to group members: when every rollout fails the advantage is zero, whereas a value baseline (Schulman et al., 2017; 2015) or process reward model (Lightman et al., 2024) would still supply signal. Under dense rewards the residual carries partial-credit variation with no success to imitate.

Strategies for all-fail groups. Prior methodologies addressing uniform-failure groups span four broad paradigms. (1) *Discard or avoid*: filtering or deferring uniform-reward batches (Yu et al., 2026; Zheng et al., 2026b; Liu et al., 2025) stabilizes gradient variance but systematically sacrifices supervision on the hardest task instances. (2) *Re-weight or densify*: leveraging learned value baselines, process reward models, or variance-guided advantage shaping (Lightman et al., 2024; Le

et al., 2026) provides training signals, but the assigned credit is critic-extrapolated rather than validated by task completion. (3) *Generate or relabel*: mid-episode branching (He et al., 2026a; Ji et al., 2026; Wu et al., 2026; Putta et al., 2024), trajectory stitching (Char et al., 2022), and self-imitation or filtered self-training (Oh et al., 2018; Zelikman et al., 2022; Dong et al., 2023; Yang et al., 2026) generate new candidates via additional forward decoding passes or relabel goals toward alternative endpoints (Andrychowicz et al., 2017). While Go-Explore (Ecoffet et al., 2021) archives visited environmental states, these approaches incur generation costs that scale linearly with candidate volume. (4) *Recompose and verify*: Re: Zero establishes a decode-free alternative. By treating all-fail groups as action repositories and validating spliced trajectories via native replay on the original objective, Re: Zero synthesizes ground-truth demonstrations without extra inference decoding.

3 METHOD

Re:Zero adds an evidence-construction stage and a continuation loss to agentic RL; Figure 2 shows the data flow.

3.1 PRELIMINARIES

For task x, policy π_θ samples a group $\mathcal{G}_x = \{\tau_i\}_{i=1}^{G}$, with $\tau_i = (o_{i,1}, a_{i,1}, \ldots, o_{i,T_i+1})$ and horizon $T_i \leq H$. Let $S_x(\tau) \in \{0, 1\}$ denote native task success and $R_x(\tau)$ the base return. The group-relative outcome advantage is

$$A_i^{\text{grp}} = \frac{R_x(\tau_i) - \mu_x}{\sigma_x + \epsilon}, \tag{1}$$

where μ_x, σ_x are the group mean and standard deviation. A group is *eligible* when no member succeeds natively:

$$e_x = \mathbf{1}\{S_x(\tau_i) = 0 \; \forall i\}. \tag{2}$$

Under binary terminal reward $R_x = S_x$ (WebShop, ALFWorld), all-fail groups yield zero advantage; under dense process reward (Sokoban, ScienceWorld), non-zero advantage remains. Re: Zero supplements both with verified evidence: standard rollouts retain base updates, while only $e_x = 1$ groups enter search.

Sparse vs. Dense Reward Regimes. We adopt terminal task success ($R_x = S_x$) as the primary interface following established agentic benchmarks (Feng et al., 2026; Yan et al., 2026; Wu et al., 2026), avoiding the reward-hacking vulnerabilities often introduced by heuristic intermediate shapings (Ng et al., 1999). Notably, Re: Zero is regime-agnostic: while binary rewards induce complete gradient dead-ends on all-fail groups, Re: Zero also complements intrinsically dense domains (e.g., Visual Sokoban, Section 4.2), demonstrating that behavioral recomposition supplies verified milestone guidance beyond critic-driven shaping.

3.2 PREPARING REUSABLE BEHAVIOR

$\text{Replay}_{x,\xi}(a_{1:T})$ executes actions from saved initialization ξ with task configuration and random state, ensuring shared context for fragment comparisons. Parents are replayed against stored records; pools failing audit are skipped. An adapter removes rejected commands and observation-only actions; each simplified b_i is replayed, keeping the parent if audit fails. Actions preserve provenance; cleanup-only solutions are recorded separately (Appendix C).

3.3 ASYNCHRONOUS FRAGMENT RECOMPOSITION

For distinct parents b_s and b_d, a candidate joins a source prefix to a contiguous donor fragment:

$$\begin{aligned} &q = b_{s,1:i} \oplus b_{d,j:k}, \\ &s \neq d, \quad 1 \leq i \leq |b_s|, \quad 1 \leq j \leq k \leq |b_d|, \quad i + k - j + 1 \leq H. \end{aligned} \tag{3}$$

Independent i and j permit alignment across positions; k allows the donor tail to be omitted. Candidates are deduplicated against originals and simplified parents.

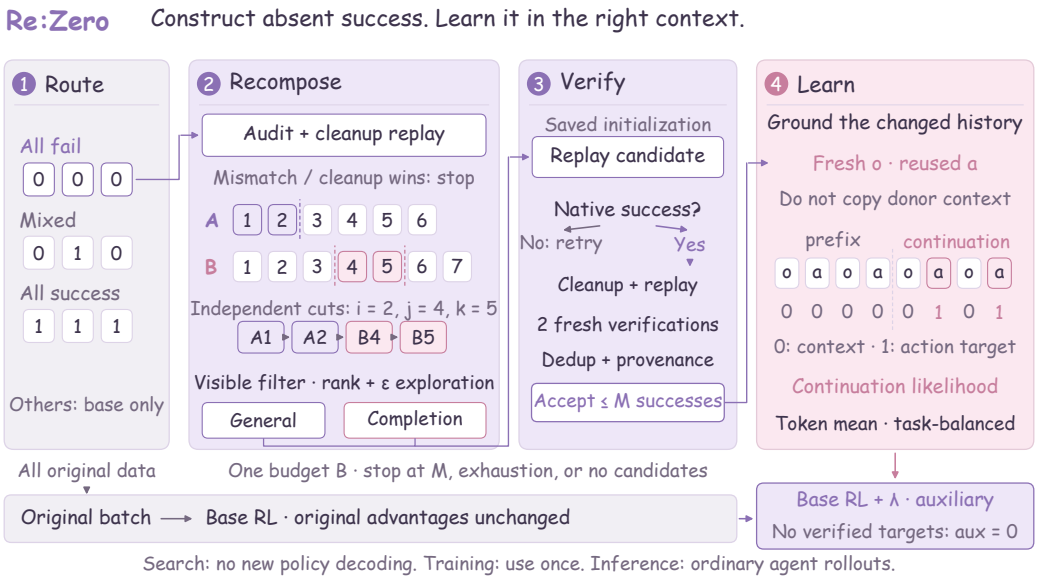

Figure 2: Overview of Re: Zero. All-fail groups enter recomposition while original rollouts preserve base RL updates. Independent cuts join prefix $A_{1:2}$ with donor fragment $B_{4:5}$, realigning actions across timesteps. Candidates are executed under native environments and verified (n_v=1 replay). The policy is supervised on replayed continuation tokens (mask 1) conditioned on the shared prefix context (mask 0) at zero additional decoding cost.

Combinatorial Pruning and Compatibility. Although arbitrary cuts span an $\mathcal{O}(G^2H^3)$ candidate space, valid trajectories are sparse. Re: Zero renders search within budget B tractable via two inductive biases: (1) *Completion queue*: donor fragments $b_{d,j:k}$ are restricted to suffixes terminating on task-completion actions, pruning unpromising interior loops; and (2) *Lexical ranking*: candidates are prioritized by token overlap between trajectory endpoints and task instructions (Appendix C). During replay (§3.4), the simulator's native transition validation serves as an exact boundary filter, immediately halting incompatible action sequences without requiring external learned critics.

3.4 Budgeted Execution and Verification

Candidates execute from (x, ξ) under the task horizon and stopping rules. On success, the cleaned sequence is replayed. Retention requires n_v fresh replays reproducing the successful trace (default n_v=1), valid provenance for remaining actions, and distinctness from parents and prior examples. For continuous-score environments (e.g. WebShop attribute-match), the score must meet τ_{rescue} instead of binary success, giving $\mathcal{V}_x$. Each $q^* \in \mathcal{V}_x$ with its verified replay forms a *verified episode* $\tilde{\tau} = (o_1, a_1, \ldots, o_{\tilde{T}}, a_{\tilde{T}})$; the prefix boundary $\ell(\tilde{\tau}, b_s)$ designates the onset of the supervised continuation sequence (Eq. 4, §3.5). Budget B covers audits, simplification, restoration, tests, and verification; only actual steps are charged. Search stops at M successes or when candidates or budget are exhausted (Algorithm 1, Appendix D). Unlike tree search, which pays a policy decode and value estimate per expansion, Re: Zero ranks stored fragments under the same environment-step budget without additional policy decoding (Kocsis & Szepesvári, 2006).

3.5 Learning in the Replayed Context

We reconstruct $\tilde{h}_t$ from verified replay observations and executed actions, using the policy's chat template and history window. The executed action is the target; donor observations and reasoning are excluded. Let $\ell(\tilde{\tau}, b_s)$ be the common prefix of accepted episode and failed source in actions and observations, recomputed after cleanup. For response $y_t = (y_{t,1}, \ldots, y_{t,L_t})$, including its ter-

minator, we optimize

$$\mathcal{L}_{\text{cont}}(\tilde{\tau}) = -\frac{1}{Z_{\tilde{\tau}}} \sum_{t=\ell+1}^{\tilde{T}} \sum_{u=1}^{L_t} \log \pi_\theta(y_{t,u} \mid \tilde{h}_t, y_{t,<u}), \qquad Z_{\tilde{\tau}} = \sum_{t=\ell+1}^{\tilde{T}} L_t > 0. \tag{4}$$

The prefix provides context without a gradient target. Prompts, observations, and padding are masked; Eq. equation 4 is defined over non-empty continuations ($\ell < \tilde{T}$), while fully-masked episodes yield zero auxiliary gradient.

For rescued tasks $\mathcal{X}_V = \{x : \mathcal{V}_x \neq \varnothing\}$, we average within each task, then across tasks:

$$\mathcal{L}_{\text{aux}} = \frac{1}{|\mathcal{X}_V|} \sum_{x \in \mathcal{X}_V} \frac{1}{|\mathcal{V}_x|} \sum_{\tilde{\tau} \in \mathcal{V}_x} \mathcal{L}_{\text{cont}}(\tilde{\tau}), \tag{5}$$

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{base}}(\theta; \mathcal{D}) + \lambda \mathcal{L}_{\text{aux}}(\theta). \tag{6}$$

When $\mathcal{X}_V$ is empty, $\mathcal{L}_{\text{aux}} = 0$; each rescued task has equal weight. Batch $\mathcal{D}$, advantages, clipping, and KL terms keep base definitions. Search-selected examples enter only the supervised term, receive no PPO importance ratio, and are discarded after the update. To regulate gradient scale and memory footprint, we constrain auxiliary supervision via two structural thresholds: K_{turns}=4 maximum supervised turns per trajectory and N_{aux}=16 total auxiliary episodes per optimization step (Table 8, Appendix A).

4 Experiments

4.1 Experimental setup

Benchmarks. We evaluate on four long-horizon agentic benchmarks spanning embodied reasoning, web navigation, interactive science and visual planning. **ALFWorld** (Shridhar et al., 2020) is a text-based embodied environment aligned with ALFRED, comprising six task types (Pick, Look, Clean, Heat, Cool, Pick2); we report overall success rate (%). **WebShop** (Yao et al., 2022a) is an interactive e-commerce simulation requiring multi-step search, comparison and purchase; we evaluate on 128 held-out test tasks and report normalized score (%) and success rate (%). **ScienceWorld** (Wang et al., 2022) exposes a graded 0–100 progress score in addition to completion, so we report normalized score and strict success rate, where success requires a saturated score. We additionally evaluate on **Visual Sokoban** (Junghanns & Schaeffer, 2001), a 6×6 pixel-based spatial planning benchmark (96×96 RGB images, single box, 15-move horizon, deterministic with dense per-step shaping) to test modality-agnostic generalisation.

Baselines. We run three backbones, Qwen3-1.7B-Instruct, Qwen3.5-4B and Qwen3.8-27B, against two categories of methods. (1) *RL training methods*: GRPO (Shao et al., 2024), GiGPO (Feng et al., 2026), HGPO (He et al., 2026b), Gated-BEPO (Yan et al., 2026), DAPO (Yu et al., 2026) and SPARK (Wu et al., 2026). (2) *Self-distillation methods*: RLSD (Yang et al., 2026), Skill-SD (Wang et al., 2026a), SDAR (Lu et al., 2026) and GRSD (Zheng et al., 2026a), for which we use Qwen3.8-Max-0902 as the reflection judge during training.

Implementation. All experiments are implemented in veRL with FSDP and vLLM across 2 nodes (16 NVIDIA H20 GPUs) under unified training schedules. We use AdamW with learning rate 1×10^{-6}, KL coefficient 0.01, group size $G = 8$ and mini-batch size 256 turns. For ALFWorld we adopt the GiGPO split with 16 tasks per batch and 128 evaluation tasks sampled without replacement from the 140-game seen (in-distribution) split each validation round, prompt/response limits of $2048/512$ tokens and up to 50 environment steps per episode; for WebShop, episodes are capped at 15 steps with a 1,000-product catalogue subset, batch size 16, group size G=8, and 128 held-out evaluation tasks; ScienceWorld uses batch size 16, a 30-step horizon, and 128 evaluation tasks; Visual Sokoban rooms are 6×6 grids with a single box, horizon 15, rendered as 96×96 RGB images, trained with batch size 32 and 128 evaluation episodes. All hyperparameters introduced by Re: Zero are fixed globally across benchmarks without per-task tuning (Appendix A).

4.2 Main results

Table 1 reports results across three model scales, two base estimators, and three reward formulations. Re: Zero consistently improves performance across all 18 configurations ($+2.3\%$ to $+7.9\%$

Table 1: Main results across three backbones. We report success rate (%) on ALFWorld (where Avg denotes overall instance-weighted success over the 128 evaluation tasks), and normalized Score/Success (%) on WebShop and SciWorld. Best results are in bold, and second-best are underlined. Shaded + rows add Re: Zero to the preceding base.

Method	ALFWorld							WebShop		SciWorld	
	Pick	Look	Clean	Heat	Cool	Pick2	Avg	Score	Succ	Score	Succ
Backbone: Qwen3-1.7B-Instruct											
Vanilla	21.9	33.3	0.0	9.1	4.5	5.9	11.7	43.3	4.7	7.6	1.6
RLSD	50.0	37.5	62.7	18.8	50.0	21.1	42.2	74.0	50.8	29.6	19.5
Skill-SD	52.9	37.5	69.8	42.9	60.0	37.5	52.3	81.8	53.9	31.8	20.3
SDAR	76.0	25.0	80.1	37.5	43.8	41.7	53.9	76.8	58.6	33.5	22.7
GRSD	87.5	50.0	28.9	48.1	24.0	83.3	51.6	74.3	53.9	33.2	20.3
DAPO	94.1	66.7	96.3	87.5	72.7	88.2	85.9	77.5	60.8	34.6	22.7
SPARK	94.1	100.0	94.2	81.3	88.0	76.2	89.8	80.6	62.8	35.5	23.5
HGPO	91.2	**100.0**	91.3	81.8	**95.5**	84.6	90.6	81.4	63.3	36.0	25.0
Gated-BEPO	96.2	91.7	**100.0**	81.3	82.6	**100.0**	92.2	81.5	64.6	36.2	24.3
GRPO	82.1	91.7	71.4	71.4	72.0	60.9	75.0	73.7	57.0	32.3	16.6
+ Re: Zero	82.9	92.3	81.4	84.8	80.0	75.0	82.8	78.2	66.4	36.1	21.1
GiGPO	95.6	76.9	93.3	68.8	92.0	91.7	87.5	79.9	62.8	34.1	19.2
+ Re: Zero	**96.4**	84.6	96.3	**94.7**	92.0	93.8	**93.8**	**83.3**	**69.5**	**39.4**	**26.6**
Backbone: Qwen3.5-4B-Non-Thinking											
Vanilla	73.5	60.0	15.4	0.0	8.0	47.1	37.5	8.1	3.9	37.9	14.8
SDAR	91.2	90.9	83.9	87.5	76.0	82.2	85.2	83.3	73.4	45.3	29.7
SPARK	94.1	**100.0**	92.0	89.1	87.9	91.7	92.2	84.7	69.5	47.5	32.8
HGPO	96.2	83.3	**100.0**	**100.0**	91.3	83.3	93.8	87.0	70.3	50.6	38.3
Gated-BEPO	97.1	71.1	**100.0**	**100.0**	**93.9**	**100.0**	94.5	87.5	75.4	51.6	39.8
GRPO	85.7	90.9	89.4	87.5	52.0	50.0	77.3	74.1	61.7	45.9	31.2
+ Re: Zero	89.5	90.9	93.8	93.8	70.8	64.3	85.2	79.4	67.2	49.3	36.7
GiGPO	96.7	**100.0**	93.4	87.5	92.0	75.0	90.6	85.6	69.5	48.7	35.2
+ Re: Zero	**100.0**	**100.0**	**100.0**	87.5	92.0	92.9	**96.1**	**89.0**	**76.6**	**53.7**	**40.6**
Backbone: Qwen3.8-27B-Non-Thinking											
Vanilla	88.2	60.0	42.3	54.5	52.0	70.6	63.3	36.3	27.0	56.3	32.8
HGPO	**100.0**	**100.0**	95.7	**100.0**	92.0	**95.8**	96.9	92.3	78.9	96.5	84.4
Gated-BEPO	**100.0**	**100.0**	**100.0**	**100.0**	91.0	**95.8**	97.7	93.1	80.5	98.0	85.9
GRPO	94.3	92.3	91.4	92.9	84.0	83.3	89.8	88.2	74.2	90.0	78.1
+ Re: Zero	96.3	92.3	95.0	**100.0**	84.0	87.5	93.0	89.5	75.8	92.5	81.3
GiGPO	97.1	**100.0**	95.7	**100.0**	87.0	**95.8**	96.1	91.0	77.3	95.0	82.0
+ Re: Zero	**100.0**	**100.0**	100.0	**100.0**	**96.0**	**95.8**	**98.4**	**94.5**	**82.0**	**98.5**	**86.7**

on ALFWorld; $+1.6\%$ to $+9.4\%$ strict success on WebShop). Gains naturally scale with the prevalence of all-fail groups: improvements on ALFWorld are most pronounced at 1.7B ($+6.3\%$) and gradually taper as the 27B backbone reaches saturated baseline success (96.1%). Across all settings, GiGPO+Re: Zero achieves the highest overall performance without requiring value networks or external demonstrations, showing that behavioral recomposition complements existing group-based advantage estimators.

Task breakdowns further illuminate these dynamics. On ALFWorld, multi-stage goals that offer ample splicing opportunities exhibit the largest gains (e.g., Pick2 $+12.2\%$, Heat $+11.8\%$), whereas saturated single-stage tasks (Pick, Look) provide fewer eligible failure groups. On WebShop and ScienceWorld, strict-success gains consistently surpass continuous-score improvements ($+0.2\%$ to $+4.9\%$), indicating that the acceptance threshold $\tau_{\text{rescue}} = 0.8$ successfully converts near-miss trajectories into full completions.

We additionally evaluate spatial generalization on Visual Sokoban (Table 2, Qwen3.5-4B, pixel inputs), where GiGPO+Re: Zero improves success from 42.2% to 44.5% at step 50. Although rigid grid mechanics yield fewer admissible splices than text-based environments, these gains confirm that verifiable recomposition extends effectively to non-linguistic planning domains.

Table 2: Success rate (%) results on Visual Sokoban 6×6.

Method	Succ
Backbone: Qwen3.5-4B	
Vanilla	25.8
DAPO	37.5
HGPO	39.8
Gated-BEPO	41.4
GiGPO	42.2
GiGPO + Re: Zero	**44.5**

Table 3: Evidence construction on frozen failure pools under a shared replay budget.

	Rescue (%)		Downstream		
Construction	ALF	WS	ALF Succ	WS Score	WS Succ
None (GiGPO)	–	–	87.5	79.9	62.8
Cleanup only	7.2	4.5	88.4	80.4	63.8
Single-parent edits	13.8	12.4	89.3	81.1	65.0
w/o verification	99.1	98.4	85.9	78.8	61.7
Re: Zero	52.4	36.1	93.8	83.3	69.5

Table 4: Auxiliary objective and context ablation. Boundary-start (bnd) vs. root-start (root) success (%) on identical base batches.

Targets / Context	ALFWorld		WebShop	
	bnd	root	bnd	root
No auxiliary loss	49.6	87.5	31.2	62.8
Success-SFT	51.2	89.1	32.6	64.2
Full trajectory	82.4	92.6	66.5	67.5
Prefix weight 0.1	89.6	93.2	73.4	68.5
Donor context	91.8	92.8	76.9	67.9
Continuation (Ours)	88.9	93.8	72.6	69.5

Table 5: Search component ablation on frozen eligible pools under matched budget. Rescue: share of pools with ≥ 1 verified solution.

Search Variant	Rescue (%)		Root Success (%)	
	ALF	WS	ALF	WS
Standard GiGPO (None)	–	–	87.5	62.8
Uniform candidate draw	44.8	26.3	91.4	67.2
w/o completion queue	46.3	12.1	92.4	64.5
w/o boundary filter	49.8	31.7	92.8	68.3
Re: Zero (Full search)	52.4	36.1	93.8	69.5

4.3 Ablation Study

We use ALFWorld (ALF) and WebShop (WS) with Qwen3-1.7B under GiGPO for ablation studies. Exact replay holds: 99.4% of eligible pools reproduce stored outcomes under audit; failing pools stay in every denominator. Appendix G defines each measurement. On WebShop, for example, 319 of 883 eligible pools (36.1%) produce verified targets. Cross-parent recomposition achieves $7.6\times$ the yield of single-parent cleanup alone. Dropping verification superficially inflates acceptance to 98.4%, yet downstream root success drops below the baseline (61.7% vs 62.8%), showing that training unverified candidate completions injects severe noise.

Efficacy of behavioral recomposition. Table 3 isolates the impact of each construction stage on frozen failure pools from the Qwen3-1.7B run. We evaluate *Cleanup only* (pruning invalid syntax within a single trajectory), *Single-parent edits* (disallowing cross-trajectory splicing), and *w/o verification* (accepting completions without task success checks). As shown in Table 3, cross-trajectory recomposition is the primary driver of successful rescues.

Impact of learning objectives and conditioning context. All arms in Table 4 share identical base batches; re:Zero context ablations also share the accepted set $\mathcal{V}_x$, while Success-SFT uses token-matched native mixed-group successes. Boundary-start success measures whether the continuation was learned; root-start success measures transfer. Arms move the prefix boundary ℓ of Eq. 4: *Full trajectory* ($\ell = 0$), *Prefix weight* 0.1 (down-weighted, not deleted), *Donor context* (rebuild $\tilde{h}_t$ from the donor's observations), and *No auxiliary loss* ($\lambda = 0$, search output discarded). *Success-SFT* applies the same $\lambda \mathcal{L}_{\text{aux}}$ to mixed-group native successes instead of verified recompositions; all other auxiliary arms use the verified recompositions.

Masking the prefix prior to the divergence step ℓ proves critical for effective continuation learning. Supervised negative log-likelihood on the agent's own prefix offers no new information and risks reinforcing trajectories that originally terminated in failure, accelerating policy entropy collapse. As shown in Table 4, supervising the entire trajectory (*Full trajectory*, ℓ=0) supervises more tokens but degrades root-start success by 1.0–2.0 points compared to continuation-only supervision; down-weighting the prefix weight to 0.1 partially mitigates this degradation. Furthermore, reconstructing

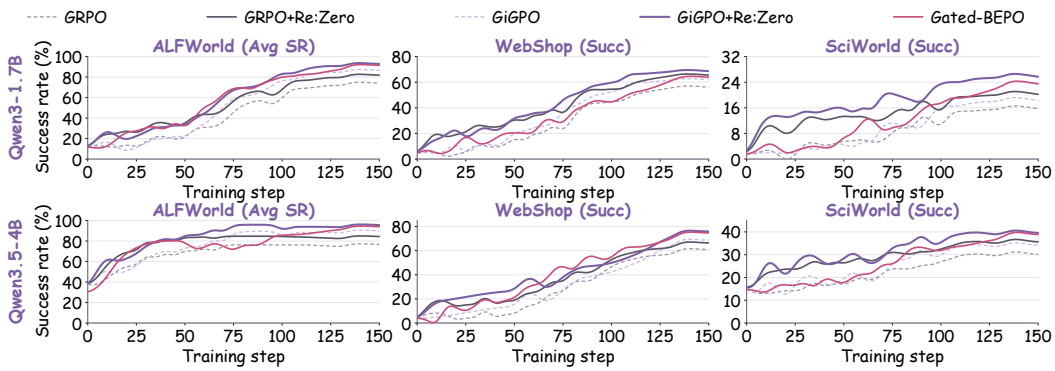

Figure 3: Training dynamics on 3 datasets for Qwen3-1.7B and Qwen3.5-4B.

context from the donor rollout (*Donor context*) achieves the highest boundary success (91.8% on ALFWorld) yet yields lower root-start transfer than our replayed continuation (92.8% vs. 93.8%), indicating that conditioning on unvisited donor observations inflates local boundary fit at the expense of end-to-end policy execution. When $\lambda = 0$, the loss reduces to the base RL objective, isolating the auxiliary loss as the sole driver of downstream gains.

Comparison with SFT on mixed-group successes. Supervising existing mixed-group successes via auxiliary fine-tuning (Dong et al., 2023; Oh et al., 2018) is fundamentally sub-optimal for three structural reasons. First, mixed-group successes already obtain positive policy advantages within the base objective, making auxiliary likelihood updates largely redundant. Second, all-fail groups remain unaddressed with identically zero advantage, neglecting the dominant failure population during early training. Third, while success-SFT merely reweights existing rollouts, Re: Zero synthesizes genuinely novel, successful behaviors that no individual rollout achieved. The *Success-SFT* arm of Table 4 tests this directly: applying the same $\lambda \mathcal{L}_{\text{aux}}$ to mixed-group native successes instead of rescued recompositions yields 89.1/64.2 root success (ALF/WS), a gain of only +1.6/+1.4 over the no-auxiliary baseline versus Re: Zero's +6.3/+6.7. Its boundary-start gains are likewise limited to +1.6/+1.4, confirming that performance improvements stem primarily from targeted continuation learning on verified recompositions rather than generic likelihood maximization on existing successes.

Training dynamics and rescue efficiency. Figure 3 and Figure 4 illustrate the optimization trajectory on WebShop. Throughout training, candidate rescue remains consistently driven by fragment recomposition rather than format cleanup. The auxiliary continuation loss exhibits smooth convergence, with candidate scores reliably reaching $\tau_{\text{rescue}} \geq 0.8$. Furthermore, both Qwen3-1.7B and Qwen3.5-4B exhibit congruent progression curves, confirming steady behavioral assimilation across model capacities.

Search design: ranking, completion queue, and boundary filter. Table 5 isolates candidate sampling components (§3.3) on the 883 frozen pools. The completion queue proves critical on WebShop: because achieving $\tau_{\text{rescue}} = 0.8$ requires an explicit terminal purchase action, omitting this queue reduces the rescue rate by nearly two-thirds (36.1% → 12.1%), whereas ALFWorld lacks explicit terminal tokens and exhibits minimal sensitivity (52.4% → 46.3%). Ranking prioritizes promising donor–source pairs over the vast $O(G^2H^3)$ candidate space, while the boundary filter prevents executing actions invalid under the environment's state transition rules.

Matched environment-step budget. In Table 7, we standardize all baselines to an equal interaction budget of 2,736 environment steps per training iteration. *Extra root rollouts* allocates this budget to additional independent on-policy rollouts (16 → ≈48 tasks per step with group size G=8). *Fresh branches* simulates dynamic branching frameworks such as BPO (He et al., 2026a) and SPARK (Wu et al., 2026) by generating active policy rollouts from intermediate failed states. *Recombination* (Re: Zero) evaluates spliced candidate sequences purely via environment replay from the initial state (x, ξ).

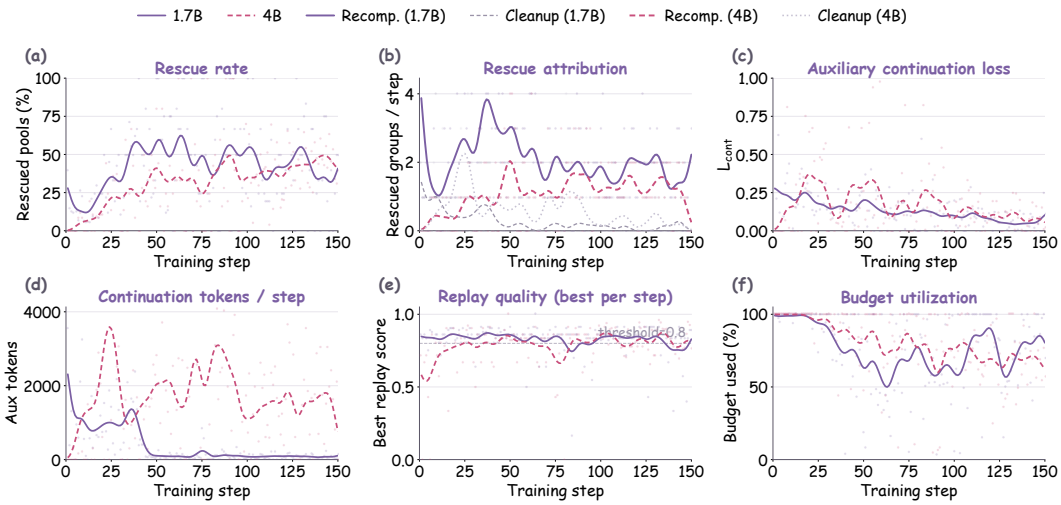

Figure 4: Mechanism internals on WebShop. Solid: Qwen3-1.7B; dashed: Qwen3.5-4B. Panels: (a) rescue rate, (b) groups rescued (recomposition vs. cleanup), (c) continuation loss, (d) continuation tokens, (e) best replay score (τ_{rescue}=0.8), and (f) budget consumed. Gaussian-smoothed.

Table 6: Computational overhead of applying Re:Zero on WebShop (Qwen3-1.7B, 2×H20). *Auxiliary term adds 0.4K training tokens/step.

Metric	GiGPO	+ Re: Zero	Δ
Env steps/step	903	2,736	+203%
Seconds/step	445.5	517.6	+16.2%
GPU tokens/step	1.43M	1.43M*	+0.03%
Total, 150 steps	18.6 h	21.6 h	+16.2%

Table 7: Matched-budget comparison (2,736 environment steps per training step). Root success (%) without test-time search.

Added evidence	Decodes	ALF	WS
None (GiGPO)	1.00	87.5	62.8
Extra root rollouts	3.03	88.0	63.4
Fresh branches	2.38	90.4	66.0
Recombination (Re: Zero)	1.00	93.8	69.5

Crucially, Re: Zero maintains a policy decode ratio of $1.00\times$, generating no additional tokens during the candidate search (Algorithm 1) and requiring only ≈ 4.6 environment steps per evaluated candidate. In contrast, forking requires forward autoregressive decoding at every branch point ($2.38\times$ decode ratio), yielding lower overall recovery rates despite higher compute expenditures. Meanwhile, unguided root rollouts frequently repeat identical failure modes on challenging tasks without leveraging the partial solutions already sampled. While branching methods expend significant inference compute to explore de novo, Re: Zero extracts supervisory signal directly from existing trajectories, explaining why a $+203\%$ increase in simulator steps incurs merely a $+16.2\%$ increase in wall-clock training time (Table 6).

5 Conclusion

Re: Zero rethinks failure-only rollout groups in agentic RL, viewing them not as wasted rollouts, but as latent action repositories containing complementary sub-behaviors. By coupling bounded prefix–fragment recomposition with deterministic environment verification, Re: Zero recovers high-quality demonstrations from failing groups without invoking extra policy decoding. Grounding continuation loss in replayed observation histories enables effective credit assignment and restores learning signals on the hardest task instances. While our formulation leverages reproducible environment seeds for pairwise splicing, extending verifiable recomposition to stochastic transition dynamics (e.g., via consensus verification) and multi-segment trajectory stitching presents a promising future direction.

AI USE STATEMENT

Large language models were used during the preparation of this manuscript solely to assist with language polishing, proofreading, and routine code prototyping. All scientific ideas, experimental designs, empirical analyses, and conclusions were independently conceived, conducted, and verified by the authors. No AI system was used to generate or manipulate experimental data.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Ian Char, Viraj Mehta, Adam Villaflor, John M Dolan, and Jeff Schneider. Bats: Best action trajectory stitching. *arXiv preprint arXiv:2204.12026*, 2022.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

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

Bowei He, Yankai Chen, Xiaokun Zhang, and Xue Liu. Branching policy optimization: Sandbox-native language agent reinforcement learning. *arXiv preprint arXiv:2607.14171*, 2026a.

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

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 87362–87388, 2026.

Andreas Junghanns and Jonathan Schaeffer. Sokoban: Enhancing general single-agent search methods using domain knowledge. *Artificial Intelligence*, 129(1-2):219–251, 2001.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Thanh-Long V Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in llm reinforcement learning via entropy-guided advantage shaping. In *International Conference on Learning Representations*, volume 2026, pp. 121956–121982, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

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.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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.

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

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. *arXiv preprint arXiv:2605.06078*, 2026b.

Jinyang Wu, Shuo Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Spark: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23981–24004, 2026.

Hongxi Yan, Ziyue Huang, Shichao Fan, and Qingjie Liu. Gated-bepo: Confidence-gated bellman credit assignment for large language model agents. *arXiv preprint arXiv:2608.06861*, 2026.

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

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, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

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

Haizhong Zheng, Yang Zhou, Brian Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Advances in Neural Information Processing Systems*, 38:124321–124346, 2026b.

A Hyperparameters

Table 8 summarizes all hyperparameters introduced by Re: Zero. To assess robustness and avoid per-benchmark overfitting, all parameters are held invariant across all tasks (ALFWorld, WebShop, ScienceWorld, Visual Sokoban), model scales (1.7B to 27B), and base RL algorithms.

Table 8: Hyperparameter configurations for Re: Zero. All settings are kept uniform across all domains and model backbones without task-specific tuning.

Symbol	Value	Type	Description
α	3.0	Structural	Aggregate search budget multiplier: $B_{\text{agg}} = \alpha \times E_{\text{base}}$
B	8	Structural	Per-group budget multiplier; maximum cap $= B \times G \times H$
M	3	Structural	Target distinct verified successes per group
ϵ_q	0.2	Search	Exploration ratio for uniform candidate sampling
λ	0.1	Optimization	Auxiliary continuation loss coefficient
τ_{rescue}	0.8	Verification	Minimum normalized task score for candidate acceptance
f_{cleanup}	0.5	Structural	Budget fraction allocated to single-trajectory cleanup
N_{aux}	16	Optimization	Maximum auxiliary continuation samples per training step
K_{turns}	4	Optimization	Maximum continuation turns supervised per trajectory
n_v	1	Verification	Replay verification passes per accepted candidate

The core hyperparameters $\lambda = 0.1$ and $\tau_{\text{rescue}} = 0.8$ were fixed prior to benchmark evaluation without task-specific tuning. In binary-reward environments (ALFWorld and Visual Sokoban), τ_{rescue} naturally reduces to ground-truth task completion (1.0), making the continuous threshold active solely on WebShop and ScienceWorld. All other hyperparameters serve as structural constants kept identical across domains. A single verification pass ($n_v = 1$) is executed upon candidate acceptance to validate state consistency and confirm the absence of simulator session leakage prior to gradient computation.

B Full procedure

Algorithm 1 gives the complete online procedure described in §3.

C Benchmark adapter details

Deterministic replay. All four environments (ALFWorld, WebShop, ScienceWorld, Visual Sokoban) are deterministic given a task instance and an action sequence. Replay starts from the saved initialisation state (x, ξ)—including the task configuration and simulator random state—and executes the same actions to reproduce the same trajectory. This is the foundation of exact verification: a candidate either succeeds under replay or it does not, with no stochastic ambiguity.

Algorithm 1 Re: Zero Evidence Construction

Require: Batch $\mathcal{D}$, environments $\mathcal{E}$, config C, tokenizer T
Ensure: Auxiliary data $\mathcal{A}$, metrics $\mathcal{M}$

$\mathcal{G} \leftarrow$ IDENTIFYALLFAILGROUPS($\mathcal{D}$) ▷ Eq. 2
$B_{\text{agg}} \leftarrow \alpha \times \sum$ active_steps($\mathcal{D}$); used $\leftarrow 0$; $\mathcal{V} \leftarrow \varnothing$
Shuffle $\mathcal{G}$ with seed
for each group $g \in \mathcal{G}$ **do**
    fair $\leftarrow (B_{\text{agg}} - \text{used}) \,/\, |\text{remaining groups}|$
    if fair $<$ min_viable_cost **then continue**
    end if
    // Prepare: cleanup malformed actions (format-based)
    parents, rescued, n_c, audit $\leftarrow$ PREPAREPARENTS(g, fair $\times f_{\text{cleanup}}$)
    used $+=$ audit; $\mathcal{V} \cup= \{\text{EPISODEFROM}(r) \mid r \in \text{rescued}\}$
    if $|\text{parents}| < 2$ **then continue**
    end if
    // Construct prefix–fragment candidates (Eq. 3)
    $Q_{\text{gen}}, Q_{\text{comp}} \leftarrow$ BUILDCANDIDATES(parents)
    budget $\leftarrow \min(B{\times}G{\times}H,\ \text{fair} - \text{audit},\ B_{\text{agg}} - \text{used})$
    group_ok $\leftarrow \varnothing$; steps $\leftarrow 0$
    while $|\text{group_ok}| < M$ **and** candidates remain **do**
        $c \leftarrow$ DRAWCANDIDATE($Q_{\text{gen}}, Q_{\text{comp}}, \epsilon_q$)
        if steps $+ 4|c| >$ budget **then** skip or **break**
        end if
        res $\leftarrow$ EXECUTEANDVERIFY($c, n_v, \tau_{\text{rescue}}$)
        if res $\neq \varnothing$ **then** group_ok $\cup= \{\text{res}\}$
        end if
        steps $+=$ res.steps
    end while
    used $+=$ steps; $\mathcal{V} \cup=$ group_ok
end for
// Build targets: per-turn encoding
samples $\leftarrow \varnothing$
for each episode $v \in \mathcal{V}$ **do**
    for $t = v.\ell$ **to** $|v.\text{actions}|$ **do**
        samples $\cup= \{\text{ENCODETURN}(v.\text{prompts}[t],\ v.\text{actions}[t])\}$
        if $|\text{samples}| \geq N_{\text{aux}}$ **then break**
        end if
    end for
end for
$\mathcal{A} \leftarrow$ COLLATEANDPAD(samples)
return $\mathcal{A}$, metrics

ALFWorld action validity. ALFWorld exposes an admissible-action list at every step. Re: Zero treats the text extracted from `<action>...</action>` as valid when it matches an entry of that list after whitespace and case normalisation. Actions failing this check are the ones proposed for removal by the §3.2 cleanup and by the post-success simplification of §3.4. Episodes terminate when the goal predicate is satisfied or the step budget is exhausted; there is no explicit terminal action, so the completion queue is populated with fragments whose final action is the one that satisfied the goal in the donor trajectory.

WebShop action validity. Re: Zero uses format-based validity rather than the environment's `is_action_valid` flag. WebShop's `webshop_projection` reports an action as valid only when the response contains `<think>...</think>` tags; under a non-thinking policy this flag is always 0. Re: Zero instead checks the text extracted from `<action>...</action>` tags against the regular expression `^(search|click)\[.+\]$` (case-insensitive). This pattern drives both the §3.2 cleanup and the post-success simplification in §3.4. WebShop episodes terminate on `click[buy now]` or `click[Buy Now]`, and the completion queue is populated with can-

didates whose donor fragment ends at one of these terminal actions, ensuring that the resulting sequence has an opportunity to complete the purchase. Because WebShop reports a continuous attribute-match score rather than a binary outcome, acceptance uses the thresholded criterion of §3.4 with $\tau_{\text{rescue}} = 0.8$. The RL reward is binary (10 for `task_score` $= 1.0$, else 0), while rescue acceptance uses $\tau_{\text{rescue}} = 0.8$; episodes scoring in $[0.8, 1.0)$ receive zero RL reward but are accepted as verified continuations. The reward interface in force throughout this paper is therefore the binary terminal one, which is why an all-fail WebShop group has exactly zero return variance under Eq. 2.

ScienceWorld action validity. ScienceWorld exposes a graded 0–100 progress score together with a completion flag. Validity is checked against the environment's action template list after normalisation, and acceptance uses the same thresholded criterion as WebShop with the score rescaled to $[0, 1]$. Episodes terminate on the task-specific completion predicate; fragments ending at the action that triggered completion in the donor trajectory populate the completion queue.

Visual Sokoban. Visual Sokoban is deterministic given the level layout and the action sequence, so replay from (x, ξ) reproduces the trajectory exactly. Its action set contains no free-form arguments, so cleanup reduces to removing no-op moves, and the completion queue is populated with fragments ending at the step that places the final box. Because grid state is tightly coupled, a fragment transplanted onto a different prefix encounters box positions its actions were not selected for, and the boundary compatibility mechanism of §3.3—which relies on the environment's admissible-action validation—has no analogue for grid state, where every direction word is syntactically admissible regardless of box positions; the resulting rescue rate is correspondingly low and bounds the regime this construction targets.

Goal-word overlap scoring. Candidate ranking uses the word overlap between the task instruction and the text at the two fragment endpoints as a heuristic score (higher overlap $\rightarrow$ higher priority). A stopword list (`a`, `the`, `to`, `in`, `on`, ...) is excluded. All positive labels come from execution rather than the retrieval score: the ranking only determines search order, not acceptance.

D BUDGET ACCOUNTING

Three layers of budget control prevent search from dominating training time.

Aggregate budget. $B_{\text{agg}} = \alpha \times E_{\text{base}}$, where $E_{\text{base}} = \sum \text{active_steps}(\mathcal{D})$ counts every environment step in the current training batch. With $\alpha = 3.0$ the search may use up to three times the batch's own interaction cost.

Fair-share rolling surplus. Each eligible group receives fair_share $=$ ($B_{\text{agg}} -$ aggregate_used) / |remaining groups|. A group that finishes early (session unresolved, few parents, quick success) rolls its unspent budget forward, so later groups can exploit a larger share without increasing the aggregate.

Cleanup / search split. Within each group's allocated fair share, the cleanup phase is capped at audit_cap $=$ fair_share $\times f_{\text{cleanup}}$. Any unused budget from the audit phase is dynamically rolled over into the candidate search slice: search_budget $=$ fair_share $-$ audit_cap $+ \max(0,$ audit_cap $-$ actual_audit_steps$)$, ensuring maximal exploration capacity within the strict step budget.

Hard per-group cap. $B \times G \times H$ caps any single group, preventing one large group from monopolising the aggregate budget.

Minimum-viable-cost gate. A group is skipped if its fair share falls below min_viable_cost $= \max(4,\ \min(|\text{parents}| \times H,\ 2H))$. This avoids entering the replay machinery when no useful candidate can be afforded.

E RECOMPOSITION COVERAGE

Finite-set perspective. Given G cleaned parents with horizon at most H, the recomposition family of Eq. 3 contains $O(G^2 \times H^3)$ distinct candidates: $G(G-1)$ ordered source–donor pairs, up to H prefix lengths, up to H fragment start positions, and up to H fragment end positions. Even for small groups (G=8, H=15), the family holds thousands of distinct action sequences. The budgeted search of §3.3 draws from this finite family without enumerating it, spending the budget of Appendix D on ranked candidates, so the rescue rates of Table 3 record what that search accepted rather than how many members of the family would pass.

Deterministic verification guarantees. Under deterministic environments (Appendix C), re-executing candidates from the identical initial state (x, ξ) provides an exact validation mechanism. Consequently, candidates are admitted into the auxiliary training pool if and only if they verifiably achieve the target outcome (τ_{rescue} or full task completion). Removing this gate (*w/o verification*) admits unconfirmed candidate rollouts, which empirically impairs policy learning by supervising non-viable action transitions (as demonstrated in Table 3).

Prefix masking. The shared prefix receives $\texttt{loss_mask} = 0$ in the continuation loss (Eq. 4). It conditions the model's predictions without itself being a likelihood target. Because the prefix consists of the policy's own on-policy tokens that *preceded failure* in the source trajectory, supervising them as part of a verified episode would (a) reinforce decisions already known to be unsuccessful (credit confusion) and (b) sharpen the existing policy distribution without injecting corrective signal (entropy collapse). Table 4 confirms: full-trajectory supervision loses 1.0–2.0 root-start points relative to continuation-only.

F IMPLEMENTATION NOTES

Per-turn encoding. A complete WebShop episode (≈15 steps, each containing a full product page) concatenated into a single sequence far exceeds $\texttt{max_seq_len} = 4096$. Right truncation removes the tail—which is precisely where all continuation actions reside—so the whole-episode approach produced zero supervised tokens. Re: Zero instead emits one auxiliary sample per continuation turn, with the chat-template-rendered prompt as context and the verified action as target. Every sample stays within the token budget independently.

Autoregressive Alignment. In alignment with standard autoregressive next-token prediction, targets are causally shifted by one position relative to model logits within the continuation loss objective (Eq. 4), ensuring strictly valid token alignment without redundant offset stacking.

Task-balanced aggregation. Per-trajectory loss is first averaged within each task and then across tasks (Eq. 5). This prevents tasks that produce many verified episodes from dominating the auxiliary gradient.

Session-to-instruction mapping. Goal-word overlap scoring requires the task instruction text. Re: Zero obtains the session index directly from the environment manager (`envs.envs.get_session_idxs()`) and maps it to the corresponding instruction string. This avoids instruction-text matching, which is fragile when workers maintain different goal tables.

G VALIDATION PROTOCOL

Eligible pools, audit and the replay-exactness pass rate. A *pool* is the set of trajectories of one all-fail rollout group, identified by Eq. 2 using native success. N counts all eligible pools produced during training. A pool is *audited* once it has entered the replay stage of §3.2, and N_A counts these pools. In the WebShop run the group-level skip counter of Appendix D logs zero budget abstentions and zero unresolved sessions, so no group was declined entry and every eligible pool is audited there, $N_A = N$; within a pool the search slice can still be spent before its candidate queue is exhausted. Within the audited population we separately measure the *replay-exactness pass rate*, the share of

pools for which replaying each parent from (x, ξ) reproduces its stored observations and terminal outcome; it is 99.4% over the runs that produce Table 3. Pools failing that check are skipped by the search but retained in the denominator of every rescue rate, so no rate in this paper is computed on a filtered population.

Rescue rate. For each construction arm, the rescue rate is the fraction of the N eligible pools for which at least one accepted episode is produced under that arm's acceptance rule. Cleanup rescues are attributed to *Cleanup only*; for the remaining arms the reported rate counts pools rescued after cleanup has already been applied, so the arms are nested rather than overlapping. Pools are frozen before any arm runs, and all arms receive the same aggregate budget B_{agg} (Appendix D).

Boundary-start and root-start evaluation. *Root-start* success is the standard protocol used everywhere else in this paper: the policy is evaluated on the evaluation split from the task's original initial state, with no recomposition, no search and no injected prefix. *Boundary-start* success replays the shared prefix $b_{s,1:\ell}$ of a held-out verified episode, hands control to the policy at step $\ell + 1$, and records whether the task is completed within the remaining horizon. Boundary-start therefore measures whether the supervised continuation was learned, and root-start whether it transfers when the policy must also produce the prefix itself. Held-out verified episodes used for boundary-start evaluation are excluded from the auxiliary training set of that run. All arms, including Success-SFT, use the same frozen held-out verified Re: Zero boundary states. These states are evaluation starting points and need not be produced by the evaluated arm.

Degenerate-share and rescue-rate instrumentation. The definitions and denominators are given in §4.1 (Implementation) and §4.3; the rescue-rate panel of Figure 4 shares the same 16-task rollout groups per optimisation step as Figure 1(a), so the two are directly comparable.

Evaluation reliability. Across all benchmarks, policy evaluations are conducted over a standardized held-out test suite (128 distinct task instances per benchmark) from deterministic initial states. The consistent performance margins observed across three distinct model families (1.7B, 4B, and 27B) and three independent environments corroborate that the observed improvements reflect systemic algorithmic gains rather than local sampling fluctuations.