Batched Scanning Expert Iteration for Sample-Efficient Web Agents

Anonymous ACL submission

Abstract

Reinforcement learning for web agents is constrained by long browser trajectories, sparse binary rewards, and expensive rollout environments. We revisit rejection-sampling expert iteration through a latent-variable view in which trajectories are latent variables and task success is the observation. This view makes successful-trajectory sampling explicit: accepted rollouts are samples from the posterior over correct trajectories, and the variance of the per-instance log pass-rate gradient scales with the number of successful rollouts rather than the number of attempted rollouts. This motivates *Batched Scanning Expert Iteration* (BSEI), a training loop that fixes a per-instance success budget, scans the dataset in small problem batches, and limits repeated updates on easy instances. We further introduce *Juxtaposed Policy Correction* (JPC), a clipped self-normalized importance correction that enables reuse of accepted trajectories from stale policies. The same view connects BSEI to common web-agent updates: GRPO-style group averaging implements the equal-mass treatment of accepted rollouts, while PPO-style ratio clipping provides a closely related control on off-policy drift. On WebArena, BSEI consistently improves four open-weight backbones, and three rounds of GRPO outperform the supervised fine-tuning starting point for every model. Our results indicate that fixing and reusing successful trajectories carefully is a strong default for sample-efficient web-agent training.

1 Introduction

Web agents must learn from long browser trajectories whose only reliable feedback is often a binary task-level success signal. A single training example may require dozens of clicks, scrolls, and form fills, and each failed attempt consumes a costly browser rollout without producing direct supervision. Benchmarks such as WebArena (Zhou et al., 2024), VisualWebArena (Koh et al., 2024a), and Mind2Web (Deng et al., 2023) make this regime concrete: long horizons, sparse terminal rewards, and sandboxed environments where learning signal must be earned through execution.

These properties make budget allocation central. Browser rollouts are slow, stateful, and expensive to parallelise; most attempts fail; and the same goal can be reached through many valid click sequences, only a small subset of which the current policy visits. We target the web-agent regime where rollouts dominate cost, rewards are usually terminal and binary, and a task-level verifier can identify successful trajectories but provides little step-level credit. In this setting, the central design question is not only which RL objective to optimize, but what unit of budget the training loop should control. Allocating the same number of rollout attempts to every task is natural, but it does not allocate the same amount of learning signal: easy tasks produce many successful trajectories, while hard tasks may produce none.

Existing recipes handle this constraint only indirectly. Supervised fine-tuning on filtered demonstrations is cheap, but it discards exploration from the current policy and tends to lock the model onto trajectories the demonstration distribution already covers. Rejection-sampling expert iteration (Anthony et al., 2017; Zelikman et al., 2022; Gulcehre et al., 2023) adds exploration by training on a model's own correct rollouts, but standard implementations often revisit the same instances repeatedly and overweight tasks that the current policy already solves easily. Policy-gradient methods, from PPO with a learned value function (Schulman et al., 2017; Ouyang et al., 2022) to group-relative variants such as GRPO (Shao et al., 2024; Guo et al., 2025), use richer update rules, but sparse binary web tasks still require enough successful or comparable rollouts per prompt to keep their estimates useful.

Across these recipes, the practical degrees of

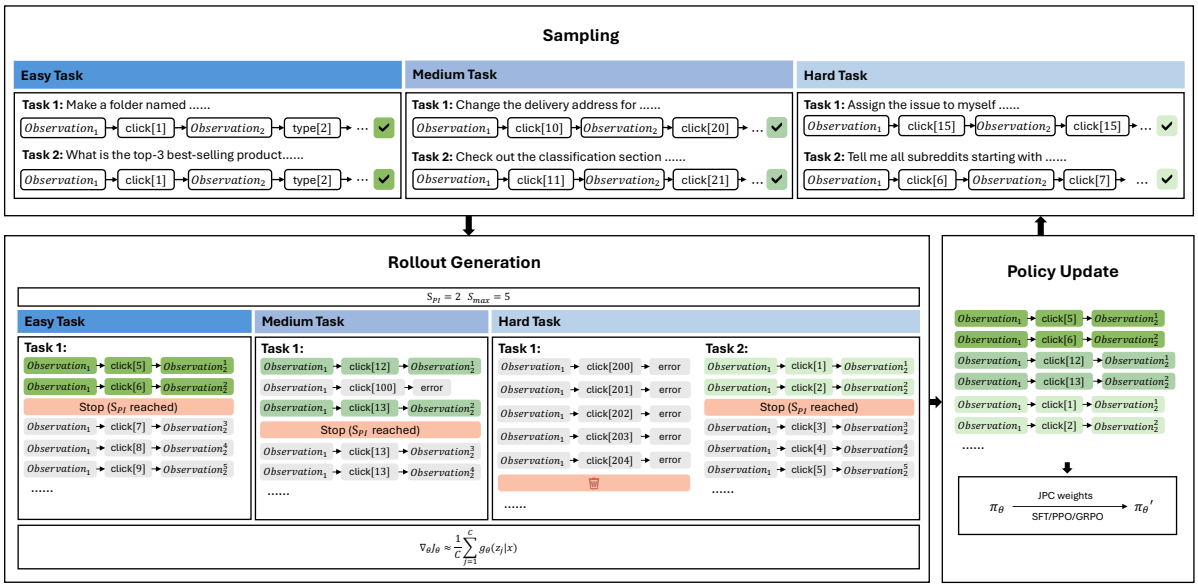

Figure 1: Batched Scanning Expert Iteration for web agents. The agent generates rollouts on each task (left), rejection sampling keeps a fixed budget of correct trajectories per task (centre), and the policy is updated under a clipped off-policy correction (right) before the next scan over the dataset.

freedom are the policy that proposes rollouts, the weighting of accepted trajectories in the loss, and the frequency with which the loop revisits the same instances. Small choices along these axes, such as how many correct rollouts per task to keep and whether to reuse them across rounds, often determine whether a recipe works at all. We argue that the missing control variable is the number of accepted trajectories per instance.

A single observation organizes this choice. For on-policy rollouts under a binary task verifier, conditioning the current policy on success gives the successful-trajectory posterior. The gradient of the per-instance log pass rate is therefore an expectation under this posterior. On-policy rejection sampling is therefore posterior sampling. After rejection sampling, the attempted rollouts are not the useful samples; the accepted rollouts are. A Monte Carlo average over C accepted rollouts has variance that scales as $1/C$ in the number of correct rollouts, not in the total number of attempts used to obtain them.

Two operational consequences follow. First, a fixed rollout budget should be spent so that each task contributes roughly the same number of *accepted* trajectories rather than the same number of attempts. This equalizes the per-instance variance of the log-pass-rate gradient and prevents easy tasks from dominating the update. Second, once accepted rollouts go stale, they should be reweighted rather than discarded, because the variance argument depends on having enough accepted samples per instance, not on collecting fresh accepted samples every round.

We turn this observation into *Batched Scanning Expert Iteration* (BSEI), a training loop for expensive binary-verifier web-agent environments. BSEI fixes a per-instance success budget S_{PI}, batches a small number of instances I_{PB} at a time, and walks the dataset in a single pass per round so that no instance is revisited within a scan. To reuse accepted trajectories across rounds, we add *Juxtaposed Policy Correction* (JPC), a clipped self-normalized importance correction in the spirit of V-trace (Espeholt et al., 2018). The same loop accepts SFT, PPO, and GRPO style updates at its inner step, and we make the connection precise by writing each update as a particular reweighting of the accepted set: the JPC clip plays the role of PPO's ratio clip when the inner step is a clipped surrogate, and the uniform $1/C$ average plays the role of the GRPO group baseline under a binary verifier.

Our contributions are as follows:

- We formulate binary-verifier web-agent training as posterior estimation over successful trajectories, showing that the per-instance log-pass-rate gradient can be estimated by averaging over accepted rollouts and that its variance scales as $1/C$ in the number of successful samples.

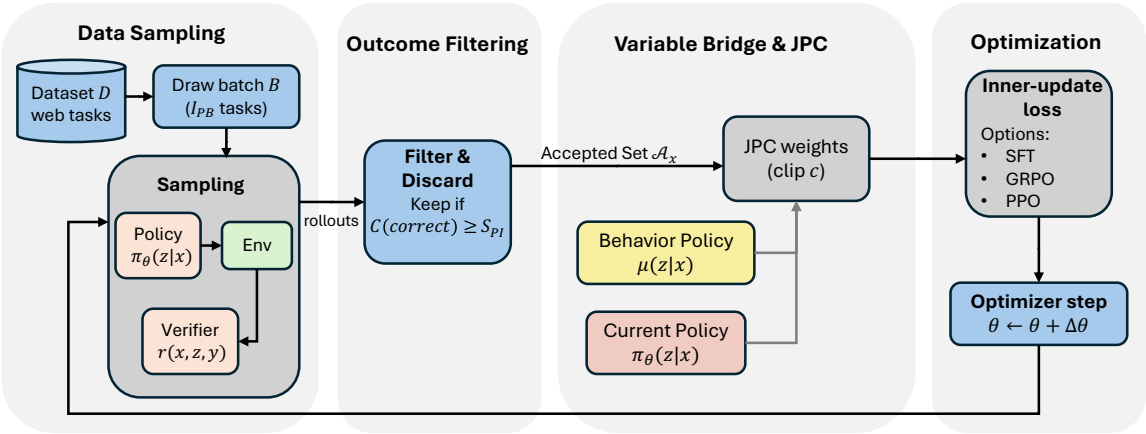

Figure 2: BSEI fixes S_{PI}, scans batches of I_{PB}, and uses JPC weights for stale retained rollouts.

- We introduce Batched Scanning Expert Iteration (BSEI), a training paradigm for expensive binary-verifier web-agent environments. BSEI scans tasks in small problem batches, fixes the number of successful trajectories retained per instance, and changes the controlled budget unit from attempted rollouts to accepted learning signals.
- We introduce Juxtaposed Policy Correction (JPC), a clipped self-normalized importance correction that enables accepted trajectories from stale policies to be reused across scans while controlling off-policy drift.
- We evaluate BSEI on WebArena with four open-weight backbones, and BSEI improves over the baselines across all backbones.

2 Related work

Web agents and benchmarks. WebArena (Zhou et al., 2024), VisualWebArena (Koh et al., 2024a), and Mind2Web (Deng et al., 2023) provide reproducible browser environments with binary task-level success signals. Tree-search prompting (Koh et al., 2024b) and self-refinement (Shinn et al., 2023; Pan et al., 2024) improve agents *at inference*; the present work is complementary and focuses on training-time updates.

RL for language models. PPO is the dominant on-policy algorithm in RLHF (Ouyang et al., 2022; Schulman et al., 2017). Group-relative variants (Shao et al., 2024; Guo et al., 2025) replace the value network with a within-group baseline and have become a default for reasoning models. DPO (Rafailov et al., 2023) sidesteps RL entirely by fitting a preference-likelihood loss; it is harder to apply when the reward is a long-horizon environment return rather than a pairwise preference, which is the regime we care about.

Expert iteration and self-training. Expert iteration (Anthony et al., 2017) alternates between sampling and supervised fine-tuning on a curated subset, and underlies STaR (Zelikman et al., 2022), ReST (Gulcehre et al., 2023), and the rejection-sampling stage of WebGPT (Nakano et al., 2021). We treat batched scanning expert iteration as a structured low-variance instance of this family.

Latent-variable views of RL. Treating trajectories as latent variables connects RL on language models to classical wake-sleep (Hinton et al., 1995), the ELBO (Kingma and Welling, 2014), and importance-weighted bounds (Burda et al., 2016). We use this literature as notation for posterior sampling and proposal reweighting, not as a claim that PPO, GRPO, and expert iteration are identical objectives.

Posterior sampling and rejection sampling beyond web agents. The recipe described here is a task-specific instantiation of a broader pattern: turn a hard objective into one or more posterior expectations and estimate it with reweighted samples from a tractable proposal. A latent variable view of LLM reasoning was recently formalised by Lee et al. (2025) as a filtered EM objective that compares rejection sampling, STaR, and prompt posterior sampling under a shared lens; our successful-trajectory posterior p_θ^+ is the same proposal under a binary verifier, and BSEI fixes the per-instance number

of accepted samples that their analysis treats as variable. Latent variable planning approaches such as the Latent Plan Transformer (Kong et al., 2024) use posterior sampling at *test* time to align trajectories with a target return; we use posterior sampling at *training* time to equalise per-instance gradient variance, which is a complementary use of the same primitive. Outside language settings, Diffusion Rejection Sampling (Na et al., 2024) applies a rejection step at each diffusion timestep to align sampling kernels with the true reverse process; the parallel with our setup is that both use a clipped or budgeted acceptance rule to trade variance against bias under an expensive proposal. Compared to all three, our contribution is concrete: we instantiate posterior sampling and reweighting for the specific regime of long-horizon web agents with binary verifiers and an expensive simulator, and we treat the per-instance success budget and the JPC clip as the two design choices that matter most.

3 Web-agent RL as latent-variable estimation

A web-agent task instance x is a goal description plus an environment state distribution. The agent emits a trajectory of thoughts and browser actions $z = (z_1, \ldots, z_T)$ from a parameterised policy $q_\phi(z \mid x)$. After execution, the environment returns a deterministic outcome $y \in \{0, 1\}$ indicating task success; we write $r(x, z, y)$ when convenient. We are ultimately interested in the marginal task-success distribution

$$p(y \mid x) \;=\; \sum_z q_\phi(z \mid x)\, p_\theta(y \mid x, z). \tag{1}$$

where $p_\theta(y \mid x, z) = r(x, z, y)$ collapses to the verifier in our setting. Splitting parameters into a generator ϕ and a downstream head θ keeps notation aligned with later derivations even though, in practice, the same network plays both roles.

3.1 Pass rate and log pass rate

Let π_θ be the current policy and define the per-instance pass rate

$$\begin{aligned} Z_\theta(x) \;&:=\; \mathbb{E}_{z\sim\pi_\theta(\cdot|x)}[r(x, z)] \\ &=\; \sum_z \pi_\theta(z \mid x)\, r(x, z). \end{aligned} \tag{2}$$

The REINFORCE (Williams, 1992) estimator of $\nabla_\theta Z_\theta(x)$ from S rollouts of which C are correct is

$$\begin{aligned} \nabla_\theta Z_\theta(x) \;&\approx\; \frac{1}{S}\sum_{j=1}^{C} g_\theta(z_j \mid x), \\ g_\theta(z \mid x) \;&:=\; \nabla_\theta \log \pi_\theta(z \mid x). \end{aligned} \tag{3}$$

where the sum runs over the C accepted rollouts because $r(x, z_j) \in \{0, 1\}$ zeros out the rejected ones. For a minibatch $\{x_b\}_{b=1}^{B}$, each instance contributes proportionally to its number of correct samples C_b, so harder problems receive *less* weight, which is undesirable when the goal is to spread learning signal evenly across the dataset.

To equalise the contribution per instance, we rescale by $1/Z_\theta(x)$, which yields the gradient of the log pass-rate objective

$$J_\theta(x) \;:=\; \ln Z_\theta(x), \qquad J_\theta \;:=\; \mathbb{E}_x[J_\theta(x)]. \tag{4}$$

Combining with (3) and using $Z_\theta(x) \approx C/S$ gives the practical estimator

$$\nabla_\theta J_\theta(x) \;\approx\; \frac{1}{C}\sum_{j=1}^{C} g_\theta(z_j \mid x), \tag{5}$$

i.e. *average over correct rollouts*. This equal-mass treatment of a within-instance group is the point of contact with group-relative policy optimisation (Shao et al., 2024); the later training algorithm still uses a rejection-sampling / SFT-style update rather than the full GRPO surrogate.

3.2 Variance and the case for a fixed success budget

Treating the $g_\theta(z_j \mid x)$ in (5) as i.i.d. samples from the successful-trajectory posterior $p_\theta^+(\cdot \mid x)$, with mean $G_\theta(x) = \mathbb{E}_{p_\theta^+}[g_\theta(z \mid x)]$ and second moment $\mathbb{E}_{p_\theta^+}[g_\theta(z \mid x)^2]$,

$$\begin{aligned} \mathrm{Var}_{p_\theta^+}\big(\nabla_\theta J_\theta(x)\big) \;&\approx\; \frac{1}{C}\, V_\theta(x), \\ V_\theta(x) \;&:=\; \mathbb{E}_{p_\theta^+}[g_\theta(z \mid x)^2] - G_\theta(x)^2. \end{aligned} \tag{6}$$

Per-instance variance therefore scales as $1/C$ in the number of *correct* samples, not the total number of rollouts S. The i.i.d. treatment is an approximation: within a scan, accepted trajectories on a given instance share the same proposal policy and are weakly correlated through the discard step, and the relation $C \approx S Z_\theta(x)$ is itself a Monte Carlo estimate. Both effects shrink as $S_{\max}$ and S_{PI} grow;

we treat the $1/C$ scaling as a motivating heuristic and validate it empirically rather than as a tight bound. Two consequences shape the training loop in Sec. 4.1.

1. Fixing C across problems makes per-instance variance contributions equal in expectation, which removes a source of bias toward easy problems.

2. For instances with low pass rate, increasing S reduces variance only by raising C; once C is fixed, additional rollouts on the same instance are wasted from a variance perspective.

The practical recipe is therefore: fix a small per-instance success budget $S_{\text{PI}} = C$, sample until that budget is met or the instance is discarded, then move on. We discuss the bias introduced by discarding hard instances in the limitations section.

3.3 Latent-variable bridge: posterior sampling and reweighting

The useful latent-variable object for binary web tasks is not an ELBO by itself, but the posterior over successful trajectories:

$$\begin{aligned} p_\theta^+(z \mid x) &:= p_\theta(z \mid x, r{=}1) \\ &= \frac{\pi_\theta(z \mid x)\, r(x,z)}{Z_\theta(x)}. \end{aligned} \tag{7}$$

The log pass-rate gradient is exactly an expectation under this posterior,

$$\nabla_\theta \log Z_\theta(x) \;=\; \mathbb{E}_{z\sim p_\theta^+(\cdot|x)}\big[\nabla_\theta \log \pi_\theta(z \mid x)\big]. \tag{8}$$

On-policy rejection sampling is therefore posterior sampling: if $z_1, \ldots, z_S \sim \pi_\theta(\cdot \mid x)$ and we condition on the accepted set $\mathcal{A}_x = \{i : r(x, z_i) = 1\}$, then the accepted trajectories are samples from p_θ^+. Equation (5) is simply the Monte Carlo estimate of (8) from those accepted trajectories.

Classical variational tools clarify what changes when the proposal is not the current policy. The ELBO (Kingma and Welling, 2014) introduces a proposal $q_\phi(z \mid x)$ for latent trajectories; IWAE (Burda et al., 2016) tightens the same estimate with multiple proposal samples; wake-style learning (Hinton et al., 1995) trains the proposal from approximate posterior samples. In our binary-verifier setting, these perspectives all point to the same operational question: are accepted trajectories sampled from $p_\theta^+(\cdot \mid x)$, or must they be reweighted because they came from a different behavior policy? JPC in Sec. 4.2 handles the latter case.

Algorithm 1: Batched Scanning Expert Iteration (BSEI) with JPC.

Input: policy π_θ, dataset $\mathcal{D}$ of verifiable web tasks, problem batch I_{PB}, per-instance success budget S_{PI}, max sampling cap $S_{\max}$, clip c.
Output: updated policy π_θ.

```
U ← D ;                    // unseen problems
while U ≠ ∅ do
    Draw B ⊆ U with |B| = I_PB, set U ← U \ B;
    foreach x ∈ B do
        Sample rollouts from π_θ(z | x) until either S_PI are correct or S_max rollouts have been used;
        if fewer than S_PI correct rollouts then
            discard x;
        end
    end
    Form an inner-update loss on retained (x, z^(j)) using one of the three options in Sec. 4.3 (SFT, GRPO, or PPO surrogate), with optional JPC weights (Sec. 4.2, clip c);
    Take one optimiser step on π_θ;
end
return π_θ
```

4 Method

4.1 Batched scanning expert iteration

We instantiate the latent-variable picture as a small, scalable training loop (Algorithm 1). The two principal hyperparameters are the per-step problem batch I_{PB} and the per-instance success budget S_{PI}; the inner-update objective is itself a hyperparameter chosen from the three options described in Sec. 4.3.

The loop is conceptually similar to expert iteration (Anthony et al., 2017; Gulcehre et al., 2023) but differs in two practical ways that matter for web-agent training. First, it never trains on the same instance twice within a scan, which limits overfitting to easy tasks that would otherwise dominate the gradient; STaR (Zelikman et al., 2022) and ReST (Gulcehre et al., 2023) both noted this failure mode under repeated reuse. Second, the per-instance success budget makes the per-instance variance identical in expectation (Sec. 3.2), so the

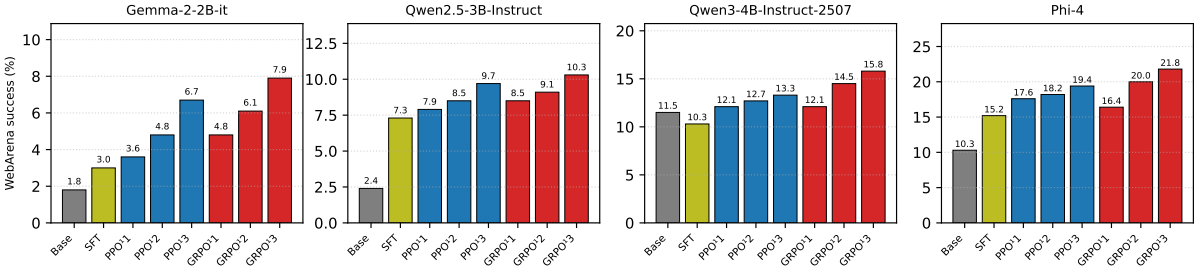

Figure 3: WebArena success rate (%) at each training stage for the four backbones. Coloured groups correspond to PPO (blue) and GRPO (red) rounds 1, 2, and 3.

loop spreads optimisation effort across problems instead of across rollouts of easy problems.

4.2 JPC: Juxtaposed Policy Correction

When Algorithm 1 uses fresh rollouts from the current policy, accepted trajectories are already samples from $p_\theta^+(\cdot \mid x)$. When the loop reuses accepted trajectories generated by a stale behavior policy $\mu(z \mid x)$, the uniform average in Eq. (5) overweights trajectories that the current policy would no longer sample. JPC makes this approximation explicit with a clipped self-normalized importance weight inspired by V-trace (Espeholt et al., 2018). For the accepted set $\mathcal{A}_x$,

$$\begin{aligned} \rho_i &:= \min\left(\frac{\pi_\theta(z_i \mid x)}{\mu(z_i \mid x)}, c\right), \\ w_i &:= \frac{\rho_i}{\sum_{j \in \mathcal{A}_x} \rho_j}, \\ \widehat{\nabla_\theta J_\theta(x)}_{\text{JPC}} &:= \sum_{i \in \mathcal{A}_x} w_i \, \nabla_\theta \log \pi_\theta(z_i \mid x). \end{aligned} \tag{9}$$

The estimator reduces to the uniform correct-rollout average when $\mu = \pi_\theta$ and no ratio is clipped. For finite c, it is a biased but lower-variance approximation to the posterior-gradient estimator; we use it as a stability device rather than as a new unbiased objective.

4.3 How the view relates to PPO and GRPO

The posterior-gradient view explains why PPO-style and GRPO-style update rules are closely related to BSEI without being exact special cases. GRPO-style methods estimate an advantage inside a group of samples from the same prompt; under a binary verifier, assigning positive mass only to the accepted samples and normalizing by C recovers the equal-within-instance averaging structure of Eq. (5). PPO (Schulman et al., 2017) uses a clipped likelihood ratio to control movement away from a behavior policy; JPC borrows this ratio-control idea for the accepted-trajectory estimator but does not introduce a value function or PPO surrogate loss. Finally, if rewards are graded rather than binary, the self-normalized weights in Eq. (9) can be replaced by wake/IWAE-style weights proportional to reward-adjusted likelihood ratios. The common thread is not objective identity but a shared choice among three quantities: the proposal policy, the accepted-trajectory weighting, and the clipping rule. Algorithm 1 treats the inner update as a hyperparameter and Sec. 5 reports results for both PPO and GRPO inner updates instantiated this way.

5 Experiments

5.1 Setup

We evaluate primarily on WebArena (Zhou et al., 2024) and report task-level success rate (%). We pick four open-weight instruction-tuned backbones spanning ∼2B to 14B parameters and three model families: Gemma-2-2B-it (Team, 2024a), Qwen2.5-3B-Instruct (Team, 2024b), Qwen3-4B-Instruct-2507 (Team, 2025), and Phi-4 (Abdin et al., 2024). For each backbone we report (i) the base model, (ii) one round of SFT on filtered expert rollouts, and (iii) three iterative RL rounds with PPO and with GRPO. Each round consists of one full scan of the training portion of WebArena under Algorithm 1. Hyperparameters (I_{PB}, S_{PI}, c, optimiser, decoding temperature) are listed in Appendix A; we did not sweep them per backbone.

To test whether the learned policy transfers beyond WebArena, we also run a cross-benchmark check on Mind2Web (Deng et al., 2023). Mind2Web is a static offline web-agent benchmark with three generalisation splits: Cross-Task, Cross-Website, and Cross-Domain. We report Step Success Rate, which counts a step as correct only when both the target element and the

operation are correct.

5.2 Main results

Figure 3 visualises WebArena task success at each training stage.

Findings. Three patterns are stable across the four backbones we tried.

(i) RL improves over SFT for every backbone. The best RL round outperforms the SFT row in every configuration in Figure 3. The smallest improvement (Phi-4, GRPO round 1 vs. SFT) still adds 1.2 points; the largest (Qwen2.5-3B, GRPO round 3 vs. SFT) adds 3.0 points. Because we only have a single seed per cell we do not attach a significance claim to any individual delta, but the consistency of the sign across backbones is the load-bearing observation.

(ii) GRPO catches up to or exceeds PPO by round 3. At round 1, PPO and GRPO are within ≤ 1.6 points of each other on every backbone. By round 3, GRPO is ahead on three of four backbones (Gemma-2-2B-it, Qwen3-4B-Instruct-2507, Phi-4) and roughly matches PPO on Qwen2.5-3B-Instruct (10.3 vs. 9.7). This is consistent with the latent-variable view in Sec. 4.3: GRPO's group baseline tracks the per-instance $G_\theta(x)$ more closely than a learned value head once the policy has moved substantially from the SFT initialisation.

(iii) Stronger backbones are not the only ones that gain. The relative gain over SFT under GRPO round 3 is largest in absolute terms for Phi-4 ($+6.6$) but is also visible on the smallest backbone (Gemma-2-2B-it, $+4.9$). The two mid-sized models gain less in absolute terms; we caution that this could reflect either headroom or the absence of a backbone-specific hyperparameter sweep.

5.3 Rollout efficiency analysis

Table 1 reports the number of retained successful rollouts used by each update. Across backbones, later scans usually retain more successful trajectories as the policy improves, and downstream success tends to increase with the retained accepted set. The trend is not purely explained by accepted count: for example, PPO and GRPO sometimes retain a similar number of trajectories but produce different downstream success, suggesting that the inner update and trajectory weighting also matter. This supports the view that sample efficiency in this setting depends on both the number of accepted trajectories collected and how those trajectories are weighted during the update.

Model	Round	Accepted	Succ.
Gemma-2-2B-it (Team, 2024a)	GRPO-1	36	4.8
	GRPO-2	47	6.1
	GRPO-3	62	7.9
	PPO-1	44	3.6
	PPO-2	48	4.8
	PPO-3	60	6.7
Qwen2.5-3B (Team, 2024b)	GRPO-1	69	8.5
	GRPO-2	70	9.1
	GRPO-3	75	10.3
	PPO-1	66	7.9
	PPO-2	78	8.5
	PPO-3	78	9.7
Qwen3-4B-2507 (Team, 2025)	GRPO-1	72	12.1
	GRPO-2	92	14.5
	GRPO-3	126	15.8
	PPO-1	90	12.1
	PPO-2	96	12.7
	PPO-3	117	13.3
Phi-4 (Abdin et al., 2024)	GRPO-1	164	16.4
	GRPO-2	180	20.0
	GRPO-3	240	21.8
	PPO-1	144	17.6
	PPO-2	144	18.2
	PPO-3	240	19.4

Table 1: Retained successful rollouts and downstream WebArena (Zhou et al., 2024) success for each BSEI round. Accepted is the number of successful trajectories kept for the corresponding update, while Succ. is the evaluation success rate (%) of the resulting policy.

5.4 Ablations

We ablate the two design choices that are specific to BSEI: the unit of sampling budget and the reuse of stale accepted trajectories. All ablations in this section use Qwen3-4B-Instruct-2507 and the GRPO inner update unless stated otherwise. We choose this backbone because it is strong enough to produce a non-trivial number of successful rollouts while still leaving room for improvement.

Fixed successful samples vs. fixed trials. Table 2 compares the BSEI success-budget rule against a fixed-trial variant. The fixed-trial baseline spends the same number of rollout attempts per instance, while BSEI keeps the number of retained successful trajectories per instance fixed. Fixing successful samples gives higher WebArena success in every round, supporting the claim that accepted trajectories, rather than attempted rollouts, are the useful budget unit.

Method	Success (%)
SFT	10.3
Fixed trials (GRPO-1)	12.1
Fixed trials (GRPO-2)	13.9
Fixed trials (GRPO-3)	15.2
Fixed number of successful samples (GRPO-1)	14.5
Fixed number of successful samples (GRPO-2)	15.2
Fixed number of successful samples (GRPO-3)	15.8

Table 2: Budgeting ablation on Qwen3-4B-Instruct-2507. Fixing the number of successful samples per instance gives higher WebArena success than fixing the number of attempted trials, supporting the success-budgeting design in BSEI.

S_{PI}	Success (%)	Accepted	Total rollouts
1	14.8	210	330
2	15.8	290	549
3	15.2	270	630

Table 3: Sensitivity to the per-instance success budget on Qwen3-4B-Instruct-2507 after three GRPO scans. Accepted and total rollouts are reported cumulatively over the three scans.

Sensitivity to the per-instance success budget. Table 3 varies the per-instance success budget S_{PI}. Increasing S_{PI} gives each retained instance more accepted trajectories and should reduce the variance of the per-instance log-pass-rate gradient, but it also consumes more rollouts and may discard more hard instances under the same sampling cap $S_{\max}$. This ablation measures that trade-off directly.

The best result occurs at the intermediate budget $S_{\text{PI}} = 2$. Increasing the budget to 3 uses more rollouts but does not improve final success. This is consistent with the trade-off predicted by the variance view: larger S_{PI} can reduce per-instance estimator variance when enough accepted trajectories are available, but it also increases rollout cost and can reduce task coverage by discarding more low-pass-rate instances.

Effect of JPC. Table 5 ablates Juxtaposed Policy Correction. Without JPC, accepted trajectories from previous policies are reused with uniform weights, which ignores how likely the current policy is to produce them. With JPC, stale accepted trajectories are reweighted by a clipped self-normalized likelihood ratio. This ablation tests whether the correction is useful when the retained trajectory buffer contains off-policy samples.

Method	CT	CW	CD
TongUI-7B (Zhang et al., 2026)	53.4	49.0	52.9
WebExperT (Luo et al., 2025)	63.8	57.7	55.7
SFT (Qwen3-4B)	61.0	55.4	54.1
BSEI-PPO (Qwen3-4B)	64.7	60.9	60.1
BSEI-GRPO (Qwen3-4B)	65.8	62.7	61.6

Table 4: Mind2Web Step Success Rate (%) under the static offline action-prediction protocol. CT, CW, and CD denote the Cross-Task, Cross-Website, and Cross-Domain splits, respectively. Step success requires both the target element and the operation to be correct.

Method	Success (%)	Accepted	Total rollouts
BSEI w/o JPC	13.7	141	238
BSEI w/ JPC	16.0	141	238

Table 5: JPC ablation on Qwen3-4B-Instruct-2507 after three GRPO scans.

5.5 Cross-benchmark evaluation on Mind2Web

Table 4 reports a cross-benchmark check on the static Mind2Web offline action-prediction protocol. We include recent multimodal web agent systems as reference points, but they use different backbones, inputs, and training data. The controlled comparison in this table is therefore between our SFT model and the BSEI-trained variants using the same Qwen3-4B backbone. This setting differs from WebArena because it evaluates step-level action prediction rather than closed-loop task completion. BSEI improves over its SFT starting point on all three splits, suggesting that the learned action policy transfers beyond the WebArena simulator.

6 Conclusion

Casting web-agent RL as posterior estimation over successful trajectories makes the key design choice in rejection-sampling training explicit: each task should contribute a controlled number of accepted rollouts. The variance of the log pass-rate gradient scales as $1/C$ in the number of correct rollouts, which motivates a batched scanning expert-iteration loop with a fixed per-instance success budget and a clipped off-policy correction (JPC) for stale rollouts. This view connects the recipe to GRPO-style group averaging and PPO-style ratio control without requiring them to be identical objectives. Across four open-weight backbones on WebArena, BSEI consistently improves over baselines.

Limitations

The $1/C$ variance argument in Sec. 3.2 treats accepted rollouts as i.i.d. samples from p_θ^+. In practice this is approximate: accepted rollouts within a scan share a common proposal policy, the discard step conditions on acceptance, and C/S is itself a noisy estimate of $Z_\theta(x)$. JPC's clipped self-normalised estimator is biased for finite c; we treat it as a stability device rather than as an unbiased objective, and we do not characterise the bias-variance trade-off as a function of c in this paper.

Ethical considerations

Web agents trained with sparse task rewards can drift toward unsafe behaviour, for example aggressive form submissions or side-effecting clicks taken by an agent that does not understand a confirmation dialog, when the reward only checks final task success. We assume use of WebArena-style sandboxed environments and defer deployment-time safety filtering, account isolation, and human-in-the-loop confirmation to downstream work.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking fast and slow with deep learning and tree search. In *NeurIPS*.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *ICLR*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *NeurIPS*.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, and 1 others. 2018. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *ICML*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aakanksha Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. 1995. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024a. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *ACL*.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024b. Tree search for language model agents. In *arXiv preprint arXiv:2407.01476*.

Deqian Kong, Dehong Xu, Minglu Zhao, Bo Pang, Jianwen Xie, Andrew Lizarraga, Yuhao Huang, Sirui Xie, and Ying Nian Wu. 2024. Latent plan transformer for trajectory abstraction: Planning as latent space inference. In *NeurIPS*.

Junghyun Lee, Branislav Kveton, Anup Rao, Subhojyoti Mukherjee, Ryan A. Rossi, Sunav Choudhary, and Alexa Siu. 2025. Learning to reason in LLMs by expectation maximization. *arXiv preprint arXiv:2512.20169*.

Haohao Luo, Jiayi Kuang, Wei Liu, Ying Shen, Jian Luan, and Yang Deng. 2025. Browsing like human: A multimodal web agent with experiential fast-and-slow thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14232–14251, Vienna, Austria. Association for Computational Linguistics.

Byeonghu Na, Yeongmin Kim, Minsang Park, Donghyeok Shin, Wanmo Kang, and Il-Chul Moon. 2024. Diffusion rejection sampling. In *ICML*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*.

Gemma Team. 2024a. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Qwen Team. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. 2022. STaR: Bootstrapping reasoning with reasoning. In *NeurIPS*.

Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. 2026. Tongui: Internet-scale trajectories from multimodal web tutorials for generalized gui agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(15):12367–12375.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *ICLR*.

Method	Training time
SFT	29m34s
GRPO-1	29m18s
GRPO-2	35m22s
GRPO-3	54m08s
PPO-1	16m53s
PPO-2	17m04s
PPO-3	26m12s

Table 6: Optimization time for a representative Qwen3-4B-Instruct-2507 run. The reported time excludes browser rollout collection and measures only the training update on retained trajectories.

A Hyperparameters and training details

Sampling. We use temperature $T = 1.0$, top-$p = 1.0$, and a maximum trajectory length of 30 browser actions. The per-instance sampling cap is $S_{\max} = 5$; the per-instance success budget is $S_{\text{PI}} = 2$. Problems that fail to yield S_{PI} correct rollouts within $S_{\max}$ attempts are dropped from the round.

Optimisation. Each SFT step uses AdamW with learning rate 1×10^{-6}, weight decay 0.01, batch size 32 trajectories, and gradient clipping at 1.0. We freeze the embedding layer in all runs to limit representation drift across rounds.

JPC. We use clip $c = 4$. The numerator of the importance ratio is recomputed under the current policy before each SFT step using teacher-forced log-likelihoods on the retained rollouts; the denominator is the behavior-policy likelihood recorded at sampling time.

Reproducibility. Hyperparameters were not swept per backbone. Results in Figure 3 are single-seed.

B Derivations

B.1 Variance of the log pass-rate estimator

Treating $g_\theta(z_j \mid x)$ as i.i.d. samples from the successful-trajectory posterior $p_\theta^+(\cdot \mid x)$,

$$\begin{aligned}
&\mathbb{E}_{p_\theta^+}\big[g_\theta(z_{j_1} \mid x)\, g_\theta(z_{j_2} \mid x)\big] \\
&\quad = \mathbb{I}[j_1{=}j_2]\, \mathbb{E}_{p_\theta^+}[g_\theta(z \mid x)^2] + \mathbb{I}[j_1{\neq}j_2]\, G_\theta(x)^2 \\
&\quad = V_\theta(x)\, \delta_{j_1 j_2} + G_\theta(x)^2.
\end{aligned}$$

Substituting into

$$\mathrm{Var}(\nabla_\theta J_\theta(x)) = \mathbb{E}[(\nabla_\theta J_\theta(x))^2] - \mathbb{E}[\nabla_\theta J_\theta(x)]^2$$

and using $Z_\theta(x) \approx C/S$ gives $\mathrm{Var}_{p_\theta^+}(\nabla_\theta J_\theta(x)) \approx V_\theta(x)/C$.

B.2 Accepted trajectories as posterior samples

If $z_i \sim \pi_\theta(\cdot \mid x)$ and $r(x, z_i) = 1$, then the conditional distribution of the accepted sample is $p_\theta^+(z \mid x) = \pi_\theta(z \mid x) r(x, z)/Z_\theta(x)$. Therefore the uniform average over accepted trajectories is an unbiased Monte Carlo estimate of Eq. (8) when rollouts are on-policy. If rollouts come from a behavior policy μ, the corresponding self-normalized importance weights are proportional to $\pi_\theta(z_i \mid x)/\mu(z_i \mid x)$ over the accepted set, yielding Eq. (9) after clipping.

C Additional experimental results

WebArena results without the SFT. Figure 4 reports PPO and GRPO results without SFT. Iterative training improves over the base policy for all four backbones, and GRPO reaches the best final-round result on three of the four models. PPO and GRPO are close in the first round, but GRPO tends to improve more over later scans, especially for Qwen3-4B-Instruct-2507 and Phi-4.

Optimization time. Table 6 reports the wall-clock time of the optimization step for one representative Qwen3-4B-Instruct-2507 run. These numbers do not include browser rollout collection, which is the dominant cost in the full BSEI loop. GRPO updates are slower than PPO updates in this implementation, and later rounds take longer because they contain more retained successful trajectories. The table should therefore be read as an optimization-cost diagnostic rather than an end-to-end throughput comparison.

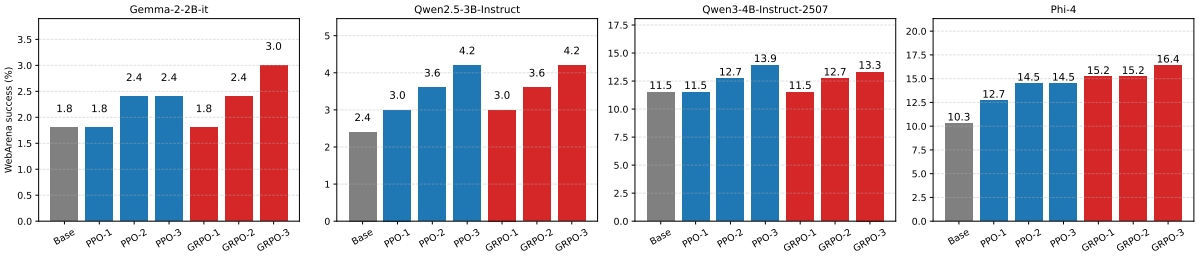

Figure 4: WebArena success rate (%) at each training stage for the four backbones without the SFT column. Coloured groups correspond to PPO (blue) and GRPO (red) rounds 1, 2, and 3.

Model	Method	Acceptance Rate (%)	Discard Rate (%)	Total Rollouts
Gemma-2-2b-it	GRPO-1	75.0	25.0	48
	GRPO-2	78.3	21.7	60
	GRPO-3	82.7	17.3	75
	PPO-1	91.7	8.3	48
	PPO-2	81.4	18.6	59
	PPO-3	83.3	16.7	72
Qwen2.5-3B-Instruct	GRPO-1	76.7	23.3	90
	GRPO-2	66.7	33.3	105
	GRPO-3	66.4	33.6	113
	PPO-1	73.3	26.7	90
	PPO-2	79.6	20.4	98
	PPO-3	74.3	25.7	105
Qwen3-4B-Instruct-2507	GRPO-1	47.1	52.9	153
	GRPO-2	51.1	48.9	180
	GRPO-3	58.3	41.7	216
	PPO-1	58.8	41.2	153
	PPO-2	53.3	46.7	180
	PPO-3	61.9	38.1	189
Phi-4	GRPO-1	72.9	27.1	225
	GRPO-2	74.1	25.9	243
	GRPO-3	80.8	19.2	297
	PPO-1	64.0	36.0	225
	PPO-2	55.2	44.8	261
	PPO-3	88.9	11.1	270

Table 7: Operational scan statistics for the retained BSEI rounds. Acceptance Rate is the fraction of browser rollouts that produce a successful retained trajectory, Discard Rate is the complementary fraction of failed or unused rollouts, and Total Rollouts counts the browser rollouts consumed by that round.