A Subgoal-driven RL Framework for Improving Long-Horizon Web Agents

Taiyi Wang[1], Sian Gooding[1], Florian Hartmann[1], Oriana Riva[1] and Edward Grefenstette[1]
[1]Google DeepMind

Abstract

Large language model (LLM)-based agents have emerged as powerful autonomous controllers for digital environments, spanning mobile interfaces, operating systems, and web browsers. Web navigation, for example, demands handling dynamic content and long action sequences, making it a particularly complex task. Existing LLM-backed agents exhibit weakened long-horizon planning abilities during RL fine-tuning, where sparse and delayed rewards from terminal Outcome Reward Models (ORMs) make it difficult for agents to identify the actions that lead to success, preventing them from sustaining coherent reasoning over extended tasks. We address this with two contributions: (1) a validated subgoal generation procedure that produces reliable, monotonically calibrated progress signals between the initial state and the terminal ORM reward; and (2) MiRA (Milestoning your Reinforcement Learning Enhanced Agent), an RL training framework using dense, milestone-based reward signals via a learned potential critic. Starting from the open Gemma3-12B base (6.4%), MiRA enables the agent to reach 43.0% on the WebArena-Lite benchmark. This performance surpasses proprietary systems such as GPT-4-Turbo (17.6%) and GPT-4o (13.9%), as well as the previous open-model state of the art, WebRL (38.8%). Our findings demonstrate that milestone-based reward shaping significantly boosts an agent's long-horizon abilities, paving the way for more robust, general-purpose autonomous systems.

1 Introduction

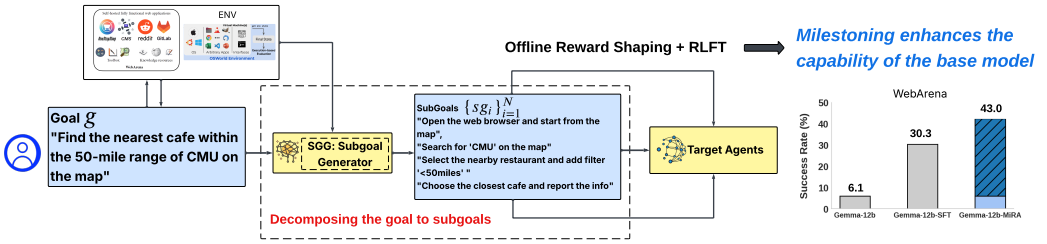


Figure 1: **Overview of MiRA.** A high-level task goal is decomposed into a sequence of subgoals by a Subgoal Generator (SGG). These subgoals drive reward shaping during RL fine-tuning of the target agent.

LLM-based agents have gained significant traction as autonomous interfaces for real-world digital environments, spanning mobile device automation [Yang et al., 2023, Christianos et al., 2024, Rawles et al., 2024], OS control [Xie et al., 2024, Zhang et al., 2025a], and web navigation [Wang et al., 2024, Qi et al., 2024, Deng et al., 2023]. Among these, **web navigation** serves as a particularly rigorous testbed: even frontier proprietary systems that perform strongly on aggregate UI-control

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

tasks suffer marked performance drops on open-ended, long-horizon web benchmarks [Zhang et al., 2024, Google DeepMind, 2025]. Smaller open-source models face an even wider gap, typically relying on supervised fine-tuning (SFT) to bridge it. However, SFT is limited by static data and poor generalization, motivating a shift toward **reinforcement learning (RL)** for long-horizon optimization [Bai et al., 2024, Wang et al., 2024]. Realistic benchmarks such as WebArena [Zhou et al., 2023] and Mind2Web [Deng et al., 2023] further underscore the challenge: current agents—proprietary and open-source alike—exhibit sharp performance degradation as task complexity increases.

Our empirical analysis further quantifies this degradation. On the WebArena-Lite benchmark [Liu et al., 2024, Koh et al., 2024], proprietary models such as Gemini-2.5-Pro exhibit "mid-task stuck" behaviors in nearly 50% of evaluation trajectories, while SFT-trained open models like Gemma-12B-SFT still fail to progress in over 30% of cases. The same deficiency dominates error distributions in prior RL-trained agents such as WebRL [Qi et al., 2024], where high-level goal instability leads to divergence from the target objective. Collectively, these observations highlight that current models—regardless of scale or fine-tuning—lack the robust internal planning and progress-awareness required to sustain reasoning over extended interactions, suggesting that a key bottleneck is *long-horizon credit assignment under sparse rewards*.

A promising direction is to introduce **explicit subgoals** as intermediate checkpoints that guide the agent toward the final objective. However, integrating such milestoning into agents raises two key challenges: **(C.1)** *Where do subgoals come from, and how reliable are they?* **(C.2)** *How can intermediate rewards be embedded in RL training to improve credit assignment while controlling the auxiliary signal's influence on the true task objective?*

To address these challenges, we propose **MiRA** (Milestoning your Reinforcement Learning Enhanced Agent), a subgoal-oriented RL fine-tuning framework. As illustrated in Figure 1, our system decomposes high-level goals into structured subgoals and uses them to provide dense, potential-based reward shaping during training. Fundamentally, our approach follows a simple principle: *"If the final goal is difficult to reach directly, increasing the probability of reaching meaningful intermediate milestones helps."* Our contributions are as follows:

1. **Solid Subgoal Generation.** We present an iterative in-context mutation strategy that generates reliable, semantically grounded subgoals, calibrated against final-task success to serve as dense reward signals as well as progress indicators for RL fine-tuning (§4.1).
2. **MiRA: Milestone-Based RL Fine-Tuning.** We introduce a dual-critic RL architecture combining a goal-conditioned value critic with a learned potential critic for dense, subgoal-driven reward shaping, trained via an iterative failure-driven curriculum. Applied to Gemma3-12B, MiRA substantially outperforms prior open-source and proprietary baselines on the verified subset of WebArena (§4.2).

2 Related Work

2.1 GUI Agents and Reinforcement Learning Fine-Tuning

LLM-based agents for GUI control can be grouped into three paradigms: *prompting-based*, *imitation-based*, and *RL-based*. Prompting-based agents steer frozen models through structured instructions and tool use, achieving strong zero-shot performance but limited adaptability. Imitation-based agents rely on SFT over demonstrations, which improves task alignment but depends on static data and fails to teach recovery from errors [Zhai et al., 2024]. This brittleness has motivated a shift toward reinforcement fine-tuning, where agents learn from active interaction. However, even with RL, adapting to complex workflows remains difficult due to the sparsity of feedback signals in multi-step environments. This challenge persists across the spectrum of GUI-based agents, including mobile controllers [Bai et al., 2024, Yang et al., 2023, Wang et al., 2024], OS assistants [Zhang et al., 2025a, Xie et al., 2024], and web agents. In the web domain, the WebRL framework [Qi et al., 2024] applies a self-evolving curriculum to address training-task scarcity, feedback sparsity, and distribution drift. Despite these advances, rewards in web navigation are often binary after many interactions, and RL-based agents still show steep performance drops as task length increases.

In hierarchical settings, an LLM planner produces intermediate sub-goals that a goal-conditioned policy executes [Wang et al., 2023, Zhang et al., 2025b], allowing semantic goal decomposition and pruning of irrelevant actions. However, even such planning architectures face critical limitations:

tasks exceeding $\sim$10 steps often fail due to impaired sub-goal coherence and drift from the original aim [Qian et al., 2025, Shlomov et al., 2025]. Recent efforts have introduced Process Reward Models (PRMs) to mitigate these failures [Xi et al., 2025, Cui et al., 2025]. Unlike Outcome Reward Models that provide sparse feedback only upon task completion, PRMs offer dense, step-by-step supervision. For instance, Web-Shepherd [Chae et al., 2025] leverages checklist-style sub-goal verification, while AgentPRM [Xi et al., 2025] introduces a dual-scoring mechanism measuring both "promise" and "progress." However, learned PRMs often suffer from high inference overhead and susceptibility to reward over-optimization. Our approach instead utilizes explicit milestones as rigid semantic checkpoints, retaining continuous progress tracking while ensuring verifiable reliability.

Prompting-based web agents such as WebVoyager [He et al., 2024] and AgentOccam [Yang et al., 2025] improve navigation through multimodal interaction or observation–action alignment, while SiRA [Deng et al., 2025] uses world-model simulation for inference-time planning. In contrast, MiRA focuses on *training-time* credit assignment: semantic milestones are converted into progress targets for RL rather than a fixed inference-time plan.

2.2 Goal-Conditioned Reinforcement Learning

Sequential web tasks naturally align with goal-conditioned reinforcement learning (GCRL) [Kaelbling, 1993], where the agent optimizes cumulative reward conditioned on a task goal. A major difficulty is *reward sparsity*, which hinders efficient credit assignment. Hindsight Experience Replay (HER) [Andrychowicz et al., 2017] addresses this by reinterpreting failed trajectories as successes for alternative goals, but assumes Markovian rewards that limit applicability to long-horizon web navigation. Recent work has pivoted toward on-policy GCRL: GCPO [Gong et al., 2024] leverages self-curriculum learning for non-Markovian rewards, while WebAgent-R1 [Wei et al., 2025] demonstrates that end-to-end on-policy RL with "thinking" steps can surpass imitation baselines.

Explicit subgoal scaffolding remains critical for exploration. VSC-RL [Wu et al., 2025] utilizes subgoal-conditioned learning to boost sample efficiency but struggles to balance intermediate completion with final-goal optimality. One Subgoal at a Time [Guo et al., 2025] decomposes formally specified temporal-logic tasks into reach–avoid objectives, whereas MiRA learns reward supervision from natural-language web milestones. Our milestones serve as *auxiliary* training signals; unlike classical PBRS, their use does not guarantee an unchanged optimal policy.

3 Preliminary and Motivation

3.1 Problem Formulation

We formulate the web navigation task as a finite-horizon Partially Observable Markov Decision Process (POMDP), defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, R, H \rangle$. Here, $\mathcal{S}$ represents the latent environment state (e.g., server-side databases and hidden DOM elements), which is inaccessible to the agent. At each timestep t, the agent receives a partial observation $o_t \in \Omega$ (comprising the rendered HTML, screenshot, and task instruction g) governed by the observation function $\mathcal{O}(o_t \mid s_t)$.

The agent selects a discrete action $a_t \in \mathcal{A}$ (e.g., clicking, typing, scrolling) according to a policy $\pi(a_t \mid o_{\leq t})$ conditioned on the history of observations. Following the action, the environment transitions to the next state s_{t+1} governed by the implicit dynamics $\mathcal{T}(s_{t+1} \mid s_t, a_t)$. The episode terminates either when the task is successfully verified or when the number of interactions reaches the horizon H, with the objective of maximizing the expected cumulative reward.

State Representation. In web navigation tasks, the instantaneous observation alone is insufficient to characterize progress, because many actions depend on prior interactions (e.g., previously opened panels, typed queries, or navigation paths). Since the latent environment state is inaccessible, we use the action–observation history as the agent's working state representation, and with slight abuse of notation we also denote it s_t: $s_t = [a_1, a_2, \ldots, a_{t-1}, o_t]$, where o_t denotes the current DOM tree, textual caption, or multimodal screenshot representation.

Reward and Objective. The environment provides a sparse binary reward signal: $r_t(s_t, a_t, g) = \mathbb{1}[s_{t+1} \in \mathcal{S}_g]$, where $\mathcal{S}_g$ denotes the set of states that satisfy the goal g. In practice, this reward is computed by an *automatic LLM-as-Judge* that evaluates goal satisfaction using the task instruction, action history, and final-state information. We adopt the open-source ORM from WebRL [Qi et al., 2024] as the final goal-level checker, and Gemini-2.5-Pro for subgoal-level checking (prompts in

Appendix A.6). The learning goal is to find a policy π that maximizes the expected discounted return $J(\pi) = \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=0}^{H}\gamma^t r_t\big]$, where $\gamma \in [0,1]$ is the discount factor.

3.2 Potential-Based Reward Shaping (PBRS)

In sparse-reward settings such as web navigation, credit assignment is difficult. Classical Potential-Based Reward Shaping (PBRS) [Ng, 2003] augments rewards by $\tilde{r}_t = r_t + \gamma\Phi(s_{t+1}) - \Phi(s_t)$. With the standard assumptions on state-based potentials and terminal handling, this construction preserves optimal policies; it is also related to potential initialization of the Q-function [Wiewiora, 2003] (proof for the classical rule in Appendix A.1). MiRA is *inspired* by potential differences, but deliberately uses $\Phi(s_{t+1}) - \Phi(s_t)$ without the next-state discount and trains its outcome critic separately. Consequently, the classical policy-invariance result does *not* apply to the implemented MiRA objective.

3.3 Motivation: Failure Analysis and Why Milestones Matter

Current web agents fail predominantly due to *mid-task stagnation*: our automated failure analysis on the full WebArena benchmark [Zhou et al., 2023] (812 tasks across 5 domains, more details in Appendix A.2) reveals that *Get Stuck Midway*—where agents enter non-productive action loops—accounts for 42–49% of all failures across base models, their SFT variants, and even proprietary systems like Gemini-2.5-Pro. Agents that fail midway are often close to the solution but lack the means to break an incorrect loop. Notably, this mid-task stagnation mode dominates uniformly across open-source backbones, their SFT variants, and the specialized WebRL agent, indicating a structural deficiency rather than a capacity gap. This suggests that explicit subgoal signals—which inject intermediate supervision precisely where agents stall—would yield the largest gains, directly motivating our milestone-based approach to long-horizon credit assignment.

4 Subgoal-Oriented Web Agent

Given the analysis above, we hypothesize that *subgoal setting serves as a critical mechanism for strengthening long-horizon reasoning and stabilizing web agent performance*. We decompose our methodology into two components: generating reliable subgoals (addressing **C.1**) and enhancing RL fine-tuning through subgoal-driven reward shaping (addressing **C.2**).

4.1 Subgoal Generation

To effectively guide web agents, we generate subgoals that bridge the gap between abstract user intents and precise step-by-step actions. Achieving this requires ensuring both reliability and proper granularity, which necessitates a fundamental understanding of website-specific constraints and features. Representative examples of generated subgoals for each website category are provided in Appendix A.6.

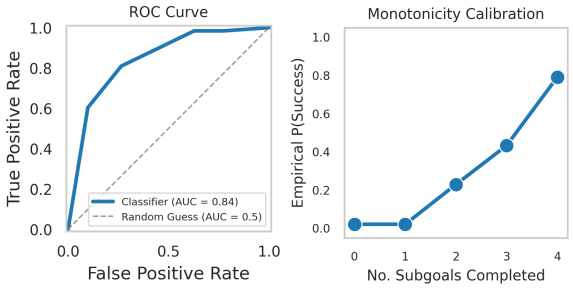


Figure 2: **Subgoal quality validation. (Left)** ROC curve using the fraction of completed subgoals as a score yields AUROC of 0.84. **(Right)** Monotonicity calibration shows strictly increasing $P(y{=}1 \mid m)$ as more subgoals m are completed (Kendall's $\tau = 0.4585$, $p < 0.001$).

Methodology. We leverage Gemini-2.5-Pro to generate subgoals given a high-level task description and the initial web state. To ensure robustness and generalization, we employ an iterative in-context learning strategy. Specifically, we curate a diverse set[1] of few-shot demonstrations mapping intents to logical intermediate milestones. To mitigate positional bias and overfitting to specific action sequences, we randomize the distribution of these few-shot examples during prompting. This

[1] 2 examples for each website category

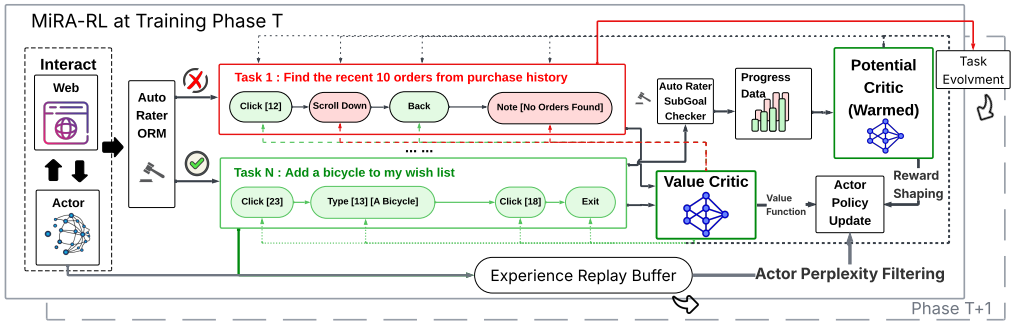


Figure 3: **The MiRA-RL training pipeline.** Trajectories are evaluated by an Auto Rater (binary success) and a SubGoal Checker (intermediate progress). Two critics are trained: a **Value Critic** V_ϕ for final success and a **Potential Critic** P_ψ for progress. The actor is updated using shaped rewards, stabilized by perplexity filtering.

shuffling or mutation ensures the model attends to the *semantics* of the task rather than memorizing rigid execution patterns. We iteratively optimize the generation context until convergence.

Evaluation of subgoal quality. To validate utility, we analyze the correlation between the generated subgoal completion vectors $\mathbf{z}_i \in \{0,1\}^K$ and the ground-truth final success $y_i \in \{0,1\}$ across a validation set of agent traces[2]. Under the *Exact Equivalence* regime, we observe an F1 of 0.6847 (Precision 0.7917, Recall 0.6032), indicating that completing all subgoals is a strong but not necessary condition for success. Under the *Graded Agreement* regime, defining a progress score $s_i = \frac{1}{K}\sum_{k=1}^{K} z_{i,k}$, we find an AUROC of **0.84** (Figure 2, left) and a strictly monotonic relationship between the number of completed subgoals and the empirical success probability, confirmed by Kendall's $\tau = 0.4585$ ($p < 0.001$; Figure 2, right). These results support using graded milestone completion as an informative, though imperfect, auxiliary signal. Exact agreement is not required: a successful agent may take an alternative valid path or miss a checker-specific milestone, while completing the milestones alone may not ensure a correct final response. Executor and granularity analyses are given in Appendix A.8.

4.2 MiRA: Agent RL Fine-Tuning with Subgoals

To incorporate subgoal-level feedback and provide dense reinforcement signals, MiRA extends the value estimation framework of prior work [Qi et al., 2024, Bai et al., 2024, Wang et al., 2024]. As illustrated in Figure 3, the training pipeline introduces a dual-critic architecture: a standard *goal-conditioned value critic* $V_\phi(s, g)$ trained on binary outcomes (final success/failure), and a novel *potential critic* $P_\psi(s, g)$ that acts as a dense **progress model** driven by a subgoal checker.

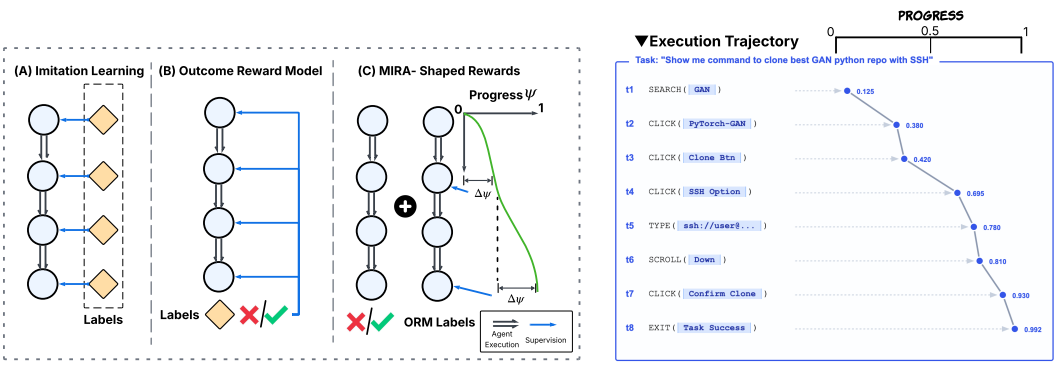
Progress labeling via subgoal completion. For each trajectory and task goal g, we obtain a binary subgoal-completion vector at each step, $\mathbf{z}_t = (z_{t,1}, \ldots, z_{t,K}) \in \{0,1\}^K$, where $z_{t,k} = 1$ indicates the k-th subgoal has been achieved[3]. This allows the agent to maintain an explicit belief state of its progress, represented by the completion vector $\mathbf{z}_t$—a signal that can also be leveraged at inference time for dynamic re-planning. Key steps are defined where the cumulative count c_t increases ($c_t > c_{t-1}$). To obtain smooth supervision, we assign normalized progress labels $p_t^* \in [0,1]$ through linear interpolation between key steps (Figure 4, right). Let t_j and t_{j+1} be the steps where the j-th and $(j{+}1)$-th subgoals complete. For any $t \in [t_j, t_{j+1}]$:

$$\alpha_t = \frac{t - t_j}{t_{j+1} - t_j}, \quad p_t^* = (1 - \alpha_t)\,\frac{j}{K} + \alpha_t\,\frac{j+1}{K}. \tag{1}$$

Example. Consider a trajectory with $K = 3$ subgoals completing at $t_1 = 2$, $t_2 = 4$. The final subgoal's logic check passes at $t = 6$, but the agent performs concluding actions until termination at $T = 9$. *Segment 1* ($t \in [0,2]$): progress ramps $0 \to 0.33$. *Segment 2* ($t \in [2,4]$): $0.33 \to 0.66$.

[2]Agent is trained with MiRA; base model is Gemma3-12B.

[3]Only positive traces are sampled.



(a) (b)

Figure 4: **Potential-based reward shaping. (Left)** Comparison of supervision signals: (A) Imitation learning matches exact steps; (B) ORM provides sparse terminal feedback; (C) MiRA uses shaped rewards via a learned potential function. **(Right)** Predicted progress score ψ increases monotonically as the agent completes subgoals, generating dense $\Delta\psi$ rewards at every timestep.

Segment 3 ($t \in [4, 9]$): we anchor the last subgoal to the trajectory end $T = 9$ rather than $t = 6$, so progress ramps $0.66 \rightarrow 1.0$ over $[4, 9]$, ensuring the potential critic learns a monotonically increasing value through the final verification steps.

Potential critic for dense shaping. We train the potential critic $P_\psi(s, g) \in [0, 1]$ to regress onto these progress labels. To construct a practical progress model, we augment a pretrained LLM with an MLP head and sigmoid activation. We use Llama3-8b (WebRL) and a vanilla-RL agent to perform exploratory rollouts across 1,237 tasks, post-process the trajectories using a subgoal checker to derive dense progress labels, and fine-tune Gemma-12B in a single supervised stage. Each state $s_t = [a_1, \ldots, a_{t-1}, o_t]$ captures the full action–observation history, enabling the critic to learn from the stable longest-common subsequences of actions that characterize successful trajectories. We then use this pre-trained network as a warm-up model and continue fine-tuning during RL. The critic is optimized via:

$$\mathcal{L}_P(\psi) = \mathbb{E}_{(s_t, g, p_t^*)} \Big[\| P_\psi(s_t, g) - p_t^* \|^2 \Big]. \tag{2}$$

Auxiliary reward shaping. During RL, the potential critic provides an auxiliary reward $r'_t = r_t + \alpha \big[P_\psi(s_{t+1}, g) - P_\psi(s_t, g) \big]$, corresponding to the $\Delta\psi$ arrows in Figure 4. The learned potential is supervised on verified successful traces, whereas the outcome critic V_ϕ is trained exclusively on the terminal ORM label. The resulting actor target combines outcome-based TD estimation with a shaped Monte-Carlo return (Eq. 5). This is a *success-anchored surrogate*, not a policy-invariant PBRS advantage: the missing γ in the potential difference and the terminal-only critic rule out applying the classical guarantee [Ng, 2003, Wiewiora, 2003]. The terminal ORM still defines task success, but auxiliary shaping can influence the learned policy. We therefore select a moderate shaping weight α empirically; larger weights degrade success rate (Appendix A.8).

Goal-conditioned value critic. The primary value critic $V_\phi(s, I)$ models the probability that the trajectory passing through state s ultimately succeeds on instruction I. We train it via binary cross-entropy against the terminal outcome label $r \in \{0, 1\}$ assigned to each trajectory by the ORM (i.e., all transitions in a successful trajectory share label r=1, and all transitions in a failed trajectory share r=0):

$$\mathcal{L}_V(\phi) = -\mathbb{E}_{(s,I,r)\sim\mathcal{D}} [r \log V_\phi(s, I) + (1-r) \log(1-V_\phi(s, I))] \,. \tag{3}$$

Following Farebrother et al. [2024], Qi et al. [2024], we adopt this classification-based formulation to handle the sparse, binary nature of terminal rewards.

Actor optimization. We use a KL-regularized, log-ratio regression update motivated by Policy Mirror Descent [Abbasi-Yadkori et al., 2019, Schulman et al., 2017]. For a conventional regularized policy-improvement step, the target takes the exponential-tilting form $\pi^*(a|s, I) \propto$

$\pi_{\text{ref}}(a|s, I) \exp(A^*(s, a, I)/\beta)$ (Appendix A.10). In MiRA, however, the transition-level target is the surrogate defined below, not an exact advantage for a single shaped MDP. The actor is fitted by MSE regression on log-probability ratios:

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_\nu \left[\left(\beta \log \frac{\pi_\theta(a|s,I)}{\pi_{\text{ref}}(a|s,I)} - A_t^{\text{shaped}} \right)^2 \right], \tag{4}$$

where A_t^{shaped} is a mixed TD–Monte-Carlo *surrogate target* rather than an unbiased advantage estimator:

$$A_t^{\text{shaped}} = \lambda\big(r'_t + \gamma V_\phi(s', g) - V_\phi(s, g)\big) + (1-\lambda)\big(G_t - V_\phi(s, g)\big), \quad G_t = \textstyle\sum_{u=t}^{T} \gamma^{u-t} r'_u. \tag{5}$$

The objective permits regression on replay samples ν and adjusts log-probability ratios in both directions. The reference-policy term regularizes these updates, but it does not itself establish monotonic policy improvement or policy invariance. An exact decomposition of this success-anchored surrogate is given in Appendix A.1. The full procedure is in Algorithm 2 (Appendix A.5).

Iterative curriculum refinement. A key limitation of static off-policy RL is vulnerability to local optima: once an agent saturates on a fixed task set, it ceases to generalize [Qi et al., 2024]. To address this, we wrap the MiRA inner loop (Algorithm 2) in an *iterative curriculum* that alternates between environment interaction and offline policy updates. In each phase, the current policy collects rollouts, which are annotated with subgoal progress labels via a Gemini-2.5-Pro subgoal checker and final-outcome labels via the WebRL-trained ORM, and filtered by perplexity to retain only informative trajectories. Failures on the current *training* tasks are analyzed to resample related tasks from a pool of 1,573 LLM-generated instructions, with Gemini-2.5-Flash helping assess similarity; the pool is filtered separately for functional overlap with the 165 evaluation tasks (Appendix A.4); for sparse categories with limited pool coverage (e.g., *Map*), new instructions are synthesized by perturbing failure samples. Over successive phases, this failure-driven loop gradually expands the task distribution while the policy internalizes successful behaviors through potential-shaped offline RL. The full algorithm and implementation details are provided in Appendix A.4.

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate primarily on WebArena-Lite [Liu et al., 2024], a curated and verified subset of WebArena [Zhou et al., 2023] comprising 165 tasks across five domains: *Shopping Admin (CMS)*, *Map*, *Shopping (OSS)*, *Reddit*, and *Gitlab*. To assess sensitivity to the curated subset, we additionally evaluate selected Gemma3-12B methods on the broader 812-task WebArena suite. This is a broader evaluation within the same environment family, *not* an independent benchmark or evidence of cross-environment generalization.

Baselines. For open-source agents, we employ Llama-3.1 (8B) and Gemma-3 (12B) as backbones, comparing against SFT, AWR [Peng et al., 2019], DigiRL [Bai et al., 2024], and WebRL [Qi et al., 2024]. For reference, we also report proprietary models GPT-4-Turbo, GPT-4o, Gemini-2.5-Flash, and Gemini-2.5-Pro.[4] All learning-based methods are initialized from the same SFT checkpoints (Llama-3.1-SFT and Gemma-3-SFT). Critic networks share the actor backbone with randomly initialized value heads. For MiRA, the potential critic is pre-trained on an offline dataset before the curriculum commences. We reproduce DigiRL within WebArena-Lite, retaining its AWR optimization and experience replay buffer with 6 rounds of online interaction. For WebRL, we use the official Llama-3.1 checkpoint and train Gemma-3 from scratch following the same curriculum. Static baselines (SFT, AWR) appear as dashed horizontal lines in SR evolution curves. The original WebRL release reports 42.4% for Llama3.1-8B in its own setup, while the 38.8% in our table is our unified-harness evaluation of the released checkpoint (Appendix A.6). Our most direct comparison is the budget- and backbone-matched Gemma3 WebRL reproduction (35.1%) versus MiRA (43.0%).

[4]Although newer Gemini and OpenAI models were released while this study was underway, we could not re-run all experiments with them. We chose to base our evaluation on these models to ensure a fair and comprehensive assessment of all method variants.

Table 1: **Task Success Rate (SR) comparison on WebArena-Lite.** Comparison of MiRA against proprietary and open-sourced baselines. We report Pass@1 to emphasize immediate task completion capability.

Models	#Params	Reddit	Gitlab	CMS	Map	OSS	Avg. SR
Proprietary LLMs							
GPT-4-Turbo	N/A	10.5	16.7	14.3	36.7	13.3	17.6
GPT-4o	N/A	10.5	10.0	20.0	20.0	11.1	13.9
Gemini-2.5-flash	N/A	21.1	33.3	14.3	11.5	24.4	20.6
Gemini-2.5-pro	N/A	31.6	30.0	34.3	15.4	13.3	23.0
Small Open-sourced LLMs							
AutoWebGLM [Lai et al., 2024]	6B	9.4	15.0	28.6	24.8	17.1	18.2
GLM-4-Chat [Glm et al., 2024]	9B	5.3	10.0	6.7	3.3	6.7	6.1
GLM-4 + SFT (BC)	9B	47.4	13.3	31.4	23.3	13.3	22.4
Llama3.1 + SFT (BC)	8B	36.8	6.7	20.0	33.3	17.8	20.6
Llama3.1 + AWR [Peng et al., 2019]	8B	57.9	26.7	31.4	26.7	17.8	28.5
Llama3.1 + DigiRL [Bai et al., 2024]	8B	57.9	26.7	37.1	33.3	17.8	30.3
Llama3.1 + WebRL [Qi et al., 2024]	8B	68.4	36.7	51.4	15.4	**40.0**	38.8
Gemma3 + SFT (BC)	12B	52.6	40.0	42.9	23.1	17.8	30.9
Gemma3 + DigiRL	12B	52.6	46.7	37.1	**34.6**	20.0	33.3
Gemma3 + WebRL	12B	68.4	43.3	40.0	30.8	20.0	35.1
Gemma3 + MiRA (ours)	**12B**	**73.7**	**56.7**	**54.3**	30.8	28.9	**43.0**

5.2 Results: Multi-Phase Training, Domain Trends, and Ablations

Table 1 presents the main end-to-end results through the whole fine-tuning effort. Gemma3 + MiRA (12B) achieves the highest average success rate of **43.0%**, substantially outperforming the WebRL baseline (35.1%) and DigiRL (33.3%) under the same Gemma3-SFT initialization. The margin is largest in complex domains such as *Gitlab* (56.7%) and *Shopping Admin* (54.3%), where strict procedural dependencies make sparse-reward methods particularly fragile. MiRA also surpasses all proprietary baselines by a wide margin (e.g., +20.0% absolute over Gemini-2.5-Pro), demonstrating that milestone-driven shaping closes—and reverses—the gap between open and closed models. To understand how these gains emerge, we next examine the training dynamics across curriculum phases, the per-domain breakdown, and a systematic ablation of each MiRA component.

Additional evaluations. On the full 812-task WebArena suite, Gemma3-12B SFT, WebRL, and MiRA obtain 18.8%, 24.0%, and **30.4%** Pass@1 SR, respectively: a 6.4-point margin over matched WebRL. On Llama3.1-8B in the Lite evaluation, MiRA reaches **41.2%** ($\pm$1.2 run-level SE), versus WebRL's 38.8% ($\pm$1.1). These results support robustness within the evaluated WebArena setting and across these two backbones, not transfer to independent environments.

Training dynamics and subgoal chaining. Across all training rounds, MiRA steadily improves from an initial SR of $\sim$31% to **43%**, while the baseline RL model without subgoal shaping saturates near 35% (Figure 5a). Per-domain analysis (Figure 5b) shows consistent upward trends, particularly on Gitlab and Shopping Admin, where subgoal-based shaping provides denser intermediate rewards for complex multi-step interactions.

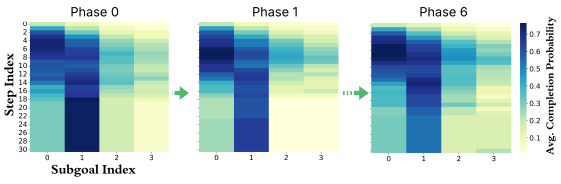
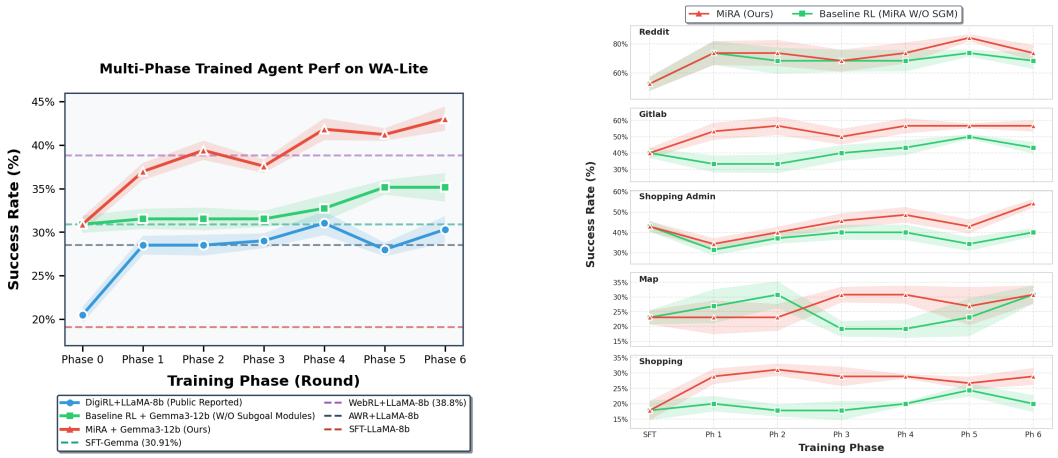


Figure 6: **Subgoal completion patterns.** Phase 0 stagnates at Subgoal 1; Phase 6 exhibits a diagonal gradient indicating sequential chaining of subgoals over the episode horizon.

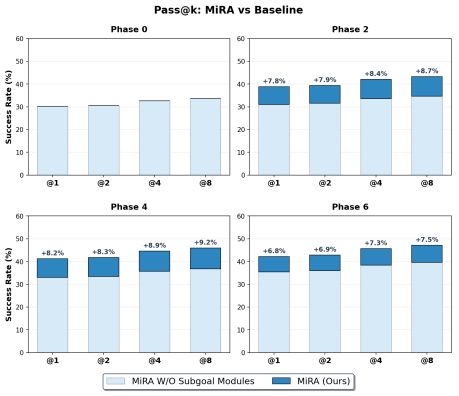
Figure 6 visualizes the average subgoal completion probability over the episode horizon. In Phase 0, probability mass concentrates as a vertical band over the first two subgoals—the agent initiates tasks but fails to transition to intermediate objectives. By Phase 6, the distribution shifts to a structured *diagonal gradient*, demonstrating that MiRA has enabled *sequential fluency*: the agent efficiently chains subgoals in lockstep with the episode timeline, transforming fragmented attempts into coherent, long-horizon trajectories.



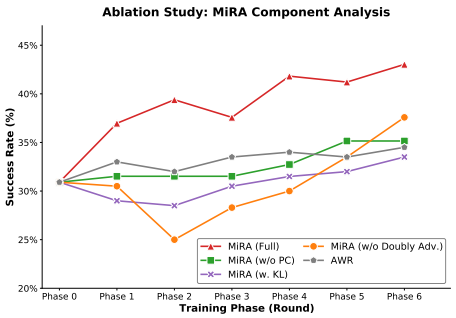
(a) Overall SR across phases.

(b) Per-site SR trends.

Figure 5: **Multi-phase and domain-level success rate trends. (a)** MiRA (red) continually improves with each refinement phase, outperforming both standard RL (green) and previous methods such as WebRL and SFT. **(b)** Per-domain SR gains of MiRA vs. baseline RL, consistent across Reddit, Gitlab, CMS, Map, and Shopping (OSS). Results averaged over 5 runs.



(a) **Pass@k scaling.** Performance across varying sample budgets ($k \in \{1, 2, 4, 8\}$) at different training stages.



(b) **Component ablation.** Impact of removing the potential critic (PC), mixed TD–Monte-Carlo estimation, and MSE updates. Each curve averages 3 trials from the same SFT init (SR≈31%).

Figure 7: **Robustness and component analysis. (a)** MiRA consistently outperforms the baseline across all sample budgets (+7.9% at pass@2 in Phase 2; +7.5% at pass@8 in Phase 6). **(b)** Removing any key component degrades SR toward the SFT baseline, validating the full framework.

Pass@k consistency and ablated variants. For a task with c successes out of n rollouts, we compute pass@k for $k \in \{1, 2, 4, 8\}$ using the unbiased estimator $\text{pass@}k = 1 - \binom{n-c}{k} / \binom{n}{k}$. As shown in Figure 7a, MiRA consistently outperforms the baseline across all sample budgets; the gap widens in Phase 2 (+7.9% at pass@2) and MiRA maintains a strong lead through Phase 6 (+7.5% at pass@8). To probe whether simpler alternatives suffice, we compare five configurations: (1) **MiRA (Full)**: the complete framework with MSE regression (Eq. 4), potential critic, and mixed TD–Monte-Carlo target (Eq. 5); (2) **MiRA w/o PC**: no potential critic, sparse outcome rewards only; (3) **MiRA w. KL**: replaces MSE with direct KL minimization; (4) **MiRA w/o Doubly Adv.**: only 1-step TD error (λ=1); (5) **AWR** [Peng et al., 2019]: a simpler off-policy baseline.

Component analysis. Each ablated variant isolates a distinct failure mode visible in Figure 7b. **(i) Mixed TD–Monte-Carlo target:** Removing MC mixing (orange) causes performance to *collapse* to ≈25% in early phases because the under-calibrated V_ϕ propagates bias directly into advantage estimates via the 1-step TD error. By Phase 6 the variant recovers to ≈37%, but the early damage delays convergence; the MC return G_t reduces dependence on the early critic, although replay-based

estimates are not necessarily unbiased. **(ii) Potential critic:** Without P_ψ (green), learning plateaus at $\approx 35\%$—the agent cannot credit intermediate actions for eventual success, confirming subgoal-based potential shaping is essential for long-horizon credit assignment. The KL variant (purple) also underperforms ($\approx 33\%$ by Phase 6); a full justification for MSE over KL divergence is provided in Appendix A.11. Together, these components address complementary failure modes, and removing any single one results in substantial degradation. Additional parameter-sensitivity ablations are in Appendix A.8.

6 Conclusion

We have presented MiRA, a framework that enhances long-horizon planning in web agents through milestone-based potential shaping. By combining an automated failure analysis with validated subgoal generation and a dual-critic RL architecture, MiRA boosts Gemma3-12B to 43.0% on WebArena-Lite—surpassing both proprietary systems and prior open-source state of the art. Our evidence identifies sparse-reward credit assignment as an important bottleneck and shows that semantically grounded milestones provide useful auxiliary training feedback in the evaluated WebArena settings. This is an empirical result rather than a policy-invariance or general cross-environment guarantee. Extended discussion connecting MiRA to process reward models, the subgoal grounding trade-off, and the path toward self-evolving autonomous agents is provided in Appendix A.12.

Acknowledgments and Disclosures

Acknowledgments. This work was conducted while Taiyi Wang was a member of the Frontier AI organization at Google DeepMind. We gratefully acknowledge our colleagues in the Frontier AI team for their support, insightful discussions, and valuable feedback throughout this research.

Funding and support. This work was supported by Google DeepMind through institutional research resources.

Competing interests. The authors declare no competing interests.

References

Y. Abbasi-Yadkori, P. Bartlett, K. Bhatia, N. Lazic, C. Szepesvari, and G. Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pages 3692–3702. PMLR, 2019.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

H. Bai, Y. Zhou, J. Pan, M. Cemri, A. Suhr, S. Levine, and A. Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.

H. Chae, S. Kim, J. Cho, S. Kim, S. Moon, G. Hwangbo, D. Lim, M. Kim, Y. Hwang, M. Gwak, et al. Web-shepherd: Advancing prms for reinforcing web agents. *arXiv preprint arXiv:2505.15277*, 2025.

F. Christianos, G. Papoudakis, T. Coste, J. Hao, J. Wang, and K. Shao. Lightweight neural app control. *arXiv preprint arXiv:2410.17883*, 2024.

G. Cui, L. Yuan, Z. Wang, H. Wang, Y. Zhang, J. Chen, W. Li, B. He, Y. Fan, T. Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

M. Deng, J. Hou, Z. Hu, and E. P. Xing. General agentic planning through simulative reasoning with world models, 2025. URL https://arxiv.org/abs/2507.23773.

X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

J. Farebrother, J. Orbay, Q. Vuong, A. A. Taïga, Y. Chebotar, T. Xiao, A. Irpan, S. Levine, P. S. Castro, A. Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

T. Glm, A. Zeng, B. Xu, B. Wang, C. Zhang, D. Yin, D. Zhang, D. Rojas, G. Feng, H. Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

X. Gong, D. Feng, K. Xu, B. Ding, and H. Wang. Goal-conditioned on-policy reinforcement learning. *Advances in neural information processing systems*, 37:45975–46001, 2024.

Google DeepMind. Technical blog post: Introducing the Gemini 2.5 Computer Use model. https://blog.google/technology/google-deepmind/gemini-computer-use-model/, October 2025. Accessed: 2025-11-28.

Z. Guo, İ. Işık, H. M. S. Ahmad, and W. Li. One subgoal at a time: Zero-shot generalization to arbitrary linear temporal logic requirements in multi-task reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/6fcb1afcc1e9c2c82c8ddddf03bcf0f6-Abstract-Conference.html.

H. He, W. Yao, K. Ma, W. Yu, Y. Dai, H. Zhang, Z. Lan, and D. Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, 2024. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8, 1993.

J. Y. Koh, R. Lo, L. Jang, V. Duvvur, M. Lim, P.-Y. Huang, G. Neubig, S. Zhou, R. Salakhutdinov, and D. Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905, 2024.

H. Lai, X. Liu, I. L. Iong, S. Yao, Y. Chen, P. Shen, H. Yu, H. Zhang, X. Zhang, Y. Dong, et al. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306, 2024.

H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

X. Liu, T. Zhang, Y. Gu, I. L. Iong, Y. Xu, X. Song, S. Zhang, H. Lai, X. Liu, H. Zhao, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024.

A. Y. Ng. *Shaping and policy search in reinforcement learning*. University of California, Berkeley, 2003.

X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Z. Qi, X. Liu, I. L. Iong, H. Lai, X. Sun, W. Zhao, Y. Yang, X. Yang, J. Sun, S. Yao, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Y. Qian, Y. Wang, J. Zhang, Y. Zong, M. Chen, H. Zhou, J. Huang, Y. Zeng, X. Hu, C. H. Song, et al. Webgrapheval: Multi-turn trajectory evaluation for web agents using graph representation. *arXiv preprint arXiv:2510.19205*, 2025.

C. Rawles, S. Clinckemaillie, Y. Chang, J. Waltz, G. Lau, M. Fair, A. Li, W. Bishop, W. Li, F. Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

J. Schulman, X. Chen, and P. Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

S. Shlomov, A. Oved, S. Marreed, I. Levy, O. Akrabi, A. Yaeli, Ł. Strąk, E. Koumpan, Y. Goldshtein, E. Shapira, et al. From benchmarks to business impact: Deploying ibm generalist agent in enterprise production. *arXiv preprint arXiv:2510.23856*, 2025.

T. Wang, Z. Wu, J. Liu, J. Hao, J. Wang, and K. Shao. Distrl: An asynchronous distributed reinforcement learning framework for on-device control agents. *arXiv preprint arXiv:2410.14803*, 2024.

Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023.

Z. Wei, W. Yao, Y. Liu, W. Zhang, Q. Lu, L. Qiu, C. Yu, P. Xu, C. Zhang, B. Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025.

E. Wiewiora. Potential-based shaping and q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

Q. Wu, J. Liu, J. Hao, J. Wang, and K. Shao. Vsc-rl: Advancing autonomous vision-language agents with variational subgoal-conditioned reinforcement learning. *arXiv e-prints*, pages arXiv–2502, 2025.

Z. Xi, C. Liao, G. Li, Y. Yang, W. Chen, Z. Zhang, B. Wang, S. Jin, Y. Zhou, J. Guan, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. *arXiv preprint arXiv:2511.08325*, 2025.

T. Xie, D. Zhang, J. Chen, X. Li, S. Zhao, R. Cao, T. J. Hua, Z. Cheng, D. Shin, F. Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

K. Yang, Y. Liu, S. Chaudhary, R. Fakoor, P. Chaudhari, G. Karypis, and H. Rangwala. AgentOccam: A simple yet strong baseline for LLM-based web agents. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oWdzUpOlkX.

Z. Yang, J. Liu, Y. Han, X. Chen, Z. Huang, B. Fu, and G. Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.

S. Zhai, H. Bai, Z. Lin, J. Pan, P. Tong, Y. Zhou, A. Suhr, S. Xie, Y. LeCun, Y. Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971, 2024.

C. Zhang, S. He, J. Qian, B. Li, L. Li, S. Qin, Y. Kang, M. Ma, G. Liu, Q. Lin, et al. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*, 2024.

C. Zhang, L. Li, S. He, X. Zhang, B. Qiao, S. Qin, M. Ma, Y. Kang, Q. Lin, S. Rajmohan, et al. Ufo: A ui-focused agent for windows os interaction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 597–622, 2025a.

K. Zhang, C. K. Lee, and Y. P. Tsang. Goal-conditioned resource allocation with hierarchical offloading and reliable consensus for blockchain-based industrial digital twins. *IEEE Transactions on Network Science and Engineering*, 2025b.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Appendix

A.1 Proof of PBRS Equivalence for Goal-Conditioned Tasks

We provide the mathematical proof that Potential-Based Reward Shaping (PBRS) is equivalent to training with the original reward function but initializing the Q-function to the potential function Φ. This proof holds for the goal-conditioned setting under the standard PBRS assumptions, including compatible discounting and terminal potentials, where the reward, value, and potential functions depend on a goal $g \in \mathcal{G}$. **It does not prove policy invariance for MiRA's implemented shaping or actor target.**

A.1.1 Equivalence of Bellman Updates

Let's define two learning scenarios for a goal-conditioned Bellman optimality update.

- **Scenario A: PBRS (Shaped Reward)**
  - Reward: $\tilde{r}(s, a, s', g) = r(s, a, s', g) + \gamma\Phi(s', g) - \Phi(s, g)$
  - Q-function: $\tilde{Q}_k(s, a, g)$
  - Initialization: $\tilde{Q}_0(s, a, g) = 0$
  - Bellman Update: $\tilde{Q}_{k+1}(s, a, g) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\tilde{r}(s, a, s', g) + \gamma \max_{a'} \tilde{Q}_k(s', a', g)\right]$
- **Scenario B: Φ-Initialization (Original Reward)**
  - Reward: $r(s, a, s', g)$
  - Q-function: $Q_k(s, a, g)$
  - Initialization: $Q_0(s, a, g) = \Phi(s, g)$
  - Bellman Update: $Q_{k+1}(s, a, g) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s, a, s', g) + \gamma \max_{a'} Q_k(s', a', g)\right]$

Claim 1 (Wiewiora, 2003, adapted for goal-conditioning)**.** *For any goal $g \in \mathcal{G}$ and all $k \geq 0$, the Q-functions from the two scenarios are related by:*

$$\tilde{Q}_k(s, a, g) = Q_k(s, a, g) - \Phi(s, g)$$

Proof by Induction. We prove this claim by induction on the iteration step k.

Base Case (k=0): We check the relationship for the initial Q-values:

$$\begin{aligned} \tilde{Q}_0(s, a, g) &= Q_0(s, a, g) - \Phi(s, g) \\ 0 &= \Phi(s, g) - \Phi(s, g) \\ 0 &= 0 \end{aligned}$$

The base case holds.

Inductive Step: Assume the relationship holds for an arbitrary step k. This is our inductive hypothesis:

$$\tilde{Q}_k(s, a, g) = Q_k(s, a, g) - \Phi(s, g)$$

We must now prove that it also holds for $k + 1$. We start with the Bellman update for Scenario A:

$$\tilde{Q}_{k+1}(s, a, g) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\tilde{r}(s, a, s', g) + \gamma \max_{a'} \tilde{Q}_k(s', a', g)\right]$$

Substitute the PBRS reward definition $\tilde{r}$:

$$\tilde{Q}_{k+1}(s, a, g) = \mathbb{E}_{s'} \left[(r(s, a, s', g) + \gamma\Phi(s', g) - \Phi(s, g)) + \gamma \max_{a'} \tilde{Q}_k(s', a', g)\right]$$

Now, substitute the inductive hypothesis $\tilde{Q}_k(s', a', g) = Q_k(s', a', g) - \Phi(s', g)$ into the $\max$ term:

$$\tilde{Q}_{k+1}(s, a, g) = \mathbb{E}_{s'} \left[(r + \gamma\Phi(s', g) - \Phi(s, g)) + \gamma \max_{a'} (Q_k(s', a', g) - \Phi(s', g))\right]$$

The potential $\Phi(s', g)$ is a function of state s' and goal g, but not action a'. Therefore, it is a constant with respect to the $\arg\max$ operation over a' and can be pulled out:

$$\max_{a'} (Q_k(s', a', g) - \Phi(s', g)) = \left(\max_{a'} Q_k(s', a', g)\right) - \Phi(s', g)$$

Substitute this back into our main equation:

$$\tilde{Q}_{k+1}(s,a,g) = \mathbb{E}_{s'}\left[(r + \gamma\Phi(s',g) - \Phi(s,g)) + \gamma\left(\left(\max_{a'} Q_k(s',a',g)\right) - \Phi(s',g)\right)\right]$$
$$= \mathbb{E}_{s'}\left[r + \gamma\Phi(s',g) - \Phi(s,g) + \gamma\max_{a'} Q_k(s',a',g) - \gamma\Phi(s',g)\right]$$

The $\gamma\Phi(s',g)$ terms cancel out:

$$\tilde{Q}_{k+1}(s,a,g) = \mathbb{E}_{s'}\left[r(s,a,s',g) + \gamma\max_{a'} Q_k(s',a',g) - \Phi(s,g)\right]$$

Since $\Phi(s,g)$ depends on the current state s and goal g, but not on the next state s', we can pull it out of the expectation over s':

$$\tilde{Q}_{k+1}(s,a,g) = \left(\mathbb{E}_{s'}\left[r(s,a,s',g) + \gamma\max_{a'} Q_k(s',a',g)\right]\right) - \Phi(s,g)$$

The term in the parentheses is exactly the Bellman update for $Q_{k+1}(s,a,g)$ from Scenario B.

$$\tilde{Q}_{k+1}(s,a,g) = Q_{k+1}(s,a,g) - \Phi(s,g)$$

This completes the inductive step. □

A.1.2 Implication: Goal-Conditioned Policy Invariance

The result $\tilde{Q}_k(s,a,g) = Q_k(s,a,g) - \Phi(s,g)$ proves that the goal-conditioned Q-function learned with PBRS is simply the "true" goal-conditioned Q-function offset by the potential $\Phi(s,g)$. This directly proves policy invariance for any given goal g.

A greedy, goal-conditioned policy π_k at step k is derived by:

$$\pi_k(s,g) = \arg\max_a Q_k(s,a,g)$$

If we derive the policy $\tilde{\pi}_k$ from the PBRS Q-function $\tilde{Q}_k$:

$$\tilde{\pi}_k(s,g) = \arg\max_a \tilde{Q}_k(s,a,g)$$

Substitute our proven relationship:

$$\tilde{\pi}_k(s,g) = \arg\max_a \left(Q_k(s,a,g) - \Phi(s,g)\right)$$

Since $\Phi(s,g)$ is a constant with respect to the $\arg\max$ over a (as it does not depend on the action a), it does not change which action yields the maximum value.

$$\arg\max_a \left(Q_k(s,a,g) - \Phi(s,g)\right) = \arg\max_a Q_k(s,a,g)$$

Therefore, for any state s and goal g:

$$\tilde{\pi}_k(s,g) = \pi_k(s,g)$$

The policy derived from the shaped Q-function is identical to the policy derived from the original Q-function at every step k. Under the conditions of the *classical* PBRS rule above, the optimal action selection is unchanged. This statement is not a guarantee for MiRA.

The shaping-weight selection and its limitations are reported in Appendix A.8.

Interpretation of MiRA's actor target. Let $\Delta P_t = P_\psi(s_{t+1},g) - P_\psi(s_t,g)$ and $G_t^{\text{out}} = \sum_{u=t}^{T}\gamma^{u-t}r_u$. Expanding Eq. 5 gives the algebraic decomposition

$$A_t^{\text{shaped}} = A_t^{\text{out}} + \alpha\left[\lambda\Delta P_t + (1-\lambda)\sum_{u=t}^{T}\gamma^{u-t}\Delta P_u\right], \tag{6}$$

$$A_t^{\text{out}} = \lambda\big(r_t + \gamma V_\phi(s_{t+1},g) - V_\phi(s_t,g)\big) + (1-\lambda)\big(G_t^{\text{out}} - V_\phi(s_t,g)\big). \tag{7}$$

Thus the actor regresses onto an outcome-anchored quantity plus a controlled progress term. Because V_ϕ is trained only on terminal outcomes while G_t contains shaped rewards, this quantity is not an unbiased advantage for either the original MDP or a conventional PBRS-shaped MDP. Its use is an empirically evaluated surrogate, not a proof of preserved optimal policies.

A.2 Detailed Failure Analysis

This appendix provides the full failure analysis content that motivated our framework, building on the condensed motivation in the main text. We developed a comprehensive automated analyzer that performs large-scale trajectory evaluation and categorization. Leveraging the reasoning and perception capabilities of the Gemini-2.5-Flash model, our analyzer systematically inspects and summarizes tons of recorded trajectories to uncover behavioral patterns. In short, our analyzer performs three core functions: **(1)** it first acts as an objective judge, providing outcome-based summarization; **(2)** it then performs failure categorization into a specific error mode; and **(3)** it conducts a root-cause analysis to pinpoint the exact decision step where the agent deviated.

A.2.1 Function 1: Trajectory Evaluation and Summarization

As its first function, the analyzer provides an objective **Failure Summarization**. This process determines *if* a failure occurred based on hardcoded behavioral rules before passing the trajectory to the next stage of diagnosis. This procedure gives out the failure summary: A trajectory is marked as a failure solely based on the outcome of the final benchmark checking rules (i.e., *"Only judge, never guess"*). This ensures that classification is objective and free of bizarre hallucinations, focusing mostly on the key state validation (e.g., Achieved XX, not achieved XX). A typical objective summary, adhering to this rule, would be:

Failure Summary

```
''Agent navigated to Hollidaysburg (film) page.  Agent extracted
information from this page.  Agent failed to move forward and
terminated.  Final benchmark check:  FAILED.''
```

A.2.2 Function 2: Categorization via Prioritized Hardcoded Rules

Once a trajectory is confirmed as a failure by the first function, the analyzer's second function is **Categorization via Prioritized Hardcoded Rules**. Failed trajectories are systematically classified into one of four mutually exclusive failure modes using the hard-coded rules detailed in Table 2, which are applied in a strict, descending order of priority.

Table 2: **Hardcoded Rules for Failure Mode Classification**

Category	Hardcoded Rules
Stop at Wrong Page (*Wrong Termination*)	The agent called the exit() function or terminated, but the message was judged *incorrect* by the final benchmark rules.
Get Stuck Midway	The agent failed and exhibited repetition, characterized by identical last N actions or a short action-state sequence repeating M times.
Fail to Make Reasonable Attempt	The agent already reached the right page, but didn't attempt to exit() **OR** The agent failed very early, either by an immediate, short exit() call or by clicking a nonsensical, distracting element as its first action.
Others (*System error, etc.*)	The agent failed, but did not trigger Rules A, B, or C, indicating a general deviation from the success path.

A.2.3 Function 3: Identifying the Decision Step

The third and final function of the analyzer is to pinpoint the exact *Key Decision Step* where the agent deviated from an optimal or successful trajectory. This capability moves beyond simple error categorization to actionable root-cause analysis. To automate this pinpointing process with high fidelity, we employ a differential analysis strategy. The analyzer aligns the agent's failed trajectory τ_{fail} against a set of *Reference Success Traces* derived from two distinct sources: **(1) Teacher Demonstrations:** Successful trajectories generated by superior, proprietary reasoning models (e.g.,

IBM-CUGA) which serve as an upper-bound oracle for reasoning quality. **(2) Golden Paths from Peers:** Validated ground-truth sequences from the training dataset or historical successful runs by other trained agents (e.g., WebRL agent). Then the analyzer scans for the **First Point of Significant Divergence**—the earliest timestep t where the agent's action a_t semantically contradicts the reference strategy. This method isolates the root cause by filtering out downstream errors that are merely symptoms of the initial mistake. Specifically, the system targets two primary divergence patterns:

- **Semantic Deviation:** Identifying the step where the agent chooses a link or search query that conceptually drifts from the cluster of actions found in successful traces (as seen in Step 3 of Figure 8).
- **Stagnation and Loop Onset:** For "Get Stuck" failures, the analyzer utilizes state-hashing to detect cyclic sub-graphs in the interaction history. It retroactively marks the entry point into the loop—rather than the loop itself—as the key decision step, highlighting the moment the agent failed to recognize the lack of state progression.

As illustrated in Figure 8, the analyzer parses the agent's complete trajectory against the task intent. In this typical example (Task ID 429), the goal is to find a specific filming location for "The Chair" in Pennsylvania. The optimal path involves two main milestones (subgoals): (1) finding the correct Wikipedia page to identify the college, and (2) searching for that college on the map.

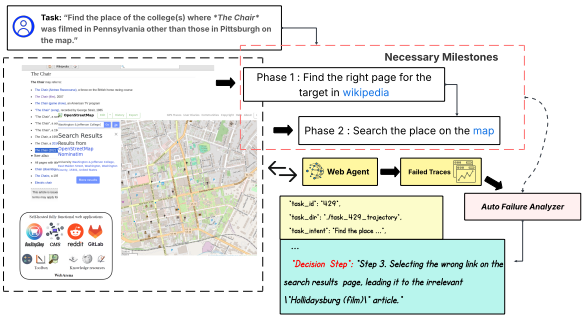


Figure 8: **Automated Identification of Key Decision Steps.** The analyzer compares the agent's trajectory against task intent and known successful patterns to pinpoint the exact step (e.g., Step 3) where a critical deviation occurred.

The auto-analyzer successfully identifies that the critical failure occurred at **Step 3**, where the agent selected an incorrect link ("The Chair (2007 film)") instead of identifying the correct TV show or other relevant entries. This single decision cascaded into a failure, as it led the agent down an irrelevant path of information.

The reliability of this identification stems directly from the robust classification rules defined previously (Function 2). Because the analyzer first rigorously classifies the *type* of failure (e.g., "Get Stuck Midway" vs. "Fail to Make Reasonable Attempt"), it can narrow down the search space for the decisive error. For instance, in a "Get Stuck Midway" scenario, the decision step is often the entry point into the repetitive loop. In a "Wrong Termination" scenario, it is the final action before incorrectly claiming success. By grounding the search for the key decision step in these validated failure modes, the analyzer achieves high precision in identifying the exact moment the agent's reasoning faltered.

A.2.4 Soundness of the Classifier

To ensure the reliability of our automated failure categorization, we implemented a rigorous prompting and validation framework.

Few-Shot Prompts: The analysis system was primed using four distinct failure examples—one for each defined category—to ground the analyzer's reasoning. This few-shot context provides the Large Language Model with concrete reference points for distinguishing between subtle failure modes (e.g., distinguishing a "stuck" agent from one that simply fails to attempt the task).

Table 3: Agreement between automated analyzer and human labels ($N = 40$).

Failure Type	Score
Stuck in Midway	10/10
Wrong Termination	10/10
Fail Attempt	8/10
Others	9/10

Validation: We conducted sanity checks on the classification accuracy using a manually labeled dataset of 40 examples (10 instances per failure type). As shown in Table 3, the analyzer demonstrates high agreement with human annotators, particularly in detecting procedural failures like "Stuck in Midway" and "Wrong Termination" (100% accuracy). Slight deviations occurred in

the "Fail to Attempt" category (8/10), where the boundary between a very short unsuccessful attempt and zero attempt can be semantically ambiguous. Validation also included an analysis of action distributions for each failure type to confirm behavioral consistency across the classified subsets.

A.2.5 Failure Number Analysis: The Mid-Stuck Challenge

Quantitative analysis of the failure distribution for various agents (Gemini-2.5-pro, Gemma, and Gemma-SFT) demonstrates that **Get Stuck Midway** is the most significant challenge across all tested models.

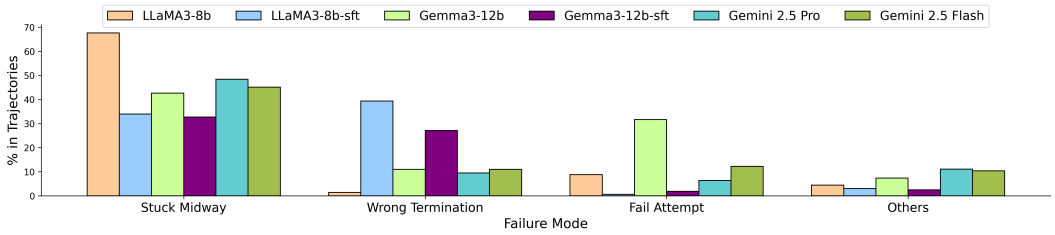


Figure 9: Overview of Failure mode distribution of existing out-of-box models

- **Midway Dominance:** The majority of failures across all baseline models (Gemini, Gemma) stem from the *Get Stuck Midway* mode, with failure ratios ranging from approximately 42% to 49%. This mode is indicative of policy oscillation and poor long-term planning, particularly when the agent encounters unexpected states or is required to execute a long, complex sequence of actions.
- **Planning Deficiencies:** This high frequency suggests that existing policies, largely based on Supervised Fine-Tuning (SFT) or similar techniques, lack the robust planning and self-correction mechanisms necessary for sequential, multi-step web tasks. An agent stuck in a loop is fundamentally failing to update its internal state or subgoal, thereby repeating failed actions.
- **The Value of Subgoals and Planning:** The other primary failure mode, *Fail to Make Reasonable Attempt*, while significant for Gemma (approximately 32%), is less dominant than the Midway failure. Agents that fail midway are often closer to the solution but lack the means to complete the final steps or break an incorrect loop. Therefore, policies integrated with explicit *subgoal completion* or *trajectory planning* signals are expected to yield the largest performance gains by directly targeting the pervasive *Get Stuck Midway* category.

A.3 Resolving the Planning Bottleneck: Post-Training Failure Distribution

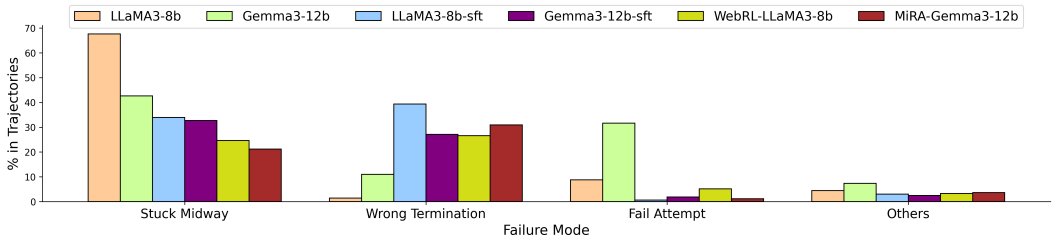


Figure 10: **Shift in Failure Distribution Post-Training.** Comparison of failure modes across Base, SFT, and RL-tuned models. MiRA (dark red) significantly reduces the "Stuck Midway" error rate compared to SFT and standard WebRL baselines, indicating that the potential-based curriculum successfully helps the agent break out of local optima and navigation loops.

To quantify the impact of MiRA on the agent's fundamental planning capabilities, we applied our automated failure analyzer to the post-training trajectories. Figure 10 presents the comparative distribution of failure modes across base LLMs, SFT baselines, the WEBRL baseline, and our MiRA agent.

The most significant finding is the drastic reduction in *Get Stuck Midway* errors. As shown in Figure 10, base models (e.g., LLaMA3-8b) and even their SFT counterparts suffer heavily from navigation loops, with Gemma3-12b-SFT exhibiting a stuck rate of $\approx 33\%$. In contrast, MiRA (Gemma3-12B) reduces this category to $\approx 21\%$, a 12-point reduction over its own Gemma3-SFT initialization. We note that the WEBRL-LLaMA3-8B reference ($\approx 25\%$ stuck rate) uses a different backbone and is included as the prior open-source state of the art rather than a matched-backbone control; the SFT-vs-MiRA comparison on the same Gemma3-12B backbone provides the direct evidence that potential-based shaping resolves the planning bottleneck. This reduction serves as strong empirical evidence that our potential-based reward shaping and curriculum strategy effectively smoothen the optimization landscape. By providing dense subgoal signals, MiRA enables the agent to recover from repetitive states where standard policies typically stall, effectively "pushing" the agent through long-horizon dependencies.

Interestingly, while MiRA minimizes navigation stalls, we observe a relative increase in *Wrong Termination* errors (rising to $\approx 31\%$). We note that this phenomenon is consistent across all trained agents: since successful termination sequences are highly rewarded during training, optimized policies naturally develop a bias toward attempting termination actions even without full confidence. Furthermore, the rate of wrong termination is compounded by the AutoRater (LLM-as-Judge), which judges success based on reaching the correct terminal page, often failing to verify the semantic details of the final answer. This shift, however, represents a clear progression in capability. Rather than failing to navigate (execution failure), the agent now traverses the full horizon to reach a terminal state, indicating that MiRA has successfully solved the lower-level planning bottleneck. This effectively exposes the higher-level semantic reasoning limitations of the underlying LLM—a preferable trade-off for scalable web automation compared to navigation stagnation.

A.4 Iterative Policy Refinement and Curriculum Generation

This section provides additional implementation details for the iterative refinement framework described in Section 5.2. The full training pipeline alternates between environment interaction, offline RL optimization, and curriculum generation to progressively improve the agent.

Task-generation quality control and leakage audit. Candidates originate from website/domain seeds and controlled task perturbations. The pipeline applies structural and semantic deduplication, feasibility and task-specific grading, teacher best-of-N execution checks, pilot-RL/replay filtering, and human review of representative tasks and failures. Curriculum evolution is driven by failures in the *training* distribution, not by evaluating the held-out benchmark. For the 1,573 generated-task pool against the 165 WebArena-Lite evaluation tasks, a hierarchical audit compares normalized instructions and metadata, intent templates and instantiated arguments, embedding-based near-duplicate candidates, and finally functional equivalence (initial state, intended operations, and terminal condition). After filtering and adjudication, no retained normalized exact duplicates or task pairs above the configured semantic threshold remain. This reduces known overlap, but cannot rule out every possible functional similarity.

Matched six-phase budget and curriculum novelty. We compare all RL methods under six matched interaction phases and optimization schedules; six phases are not claimed to prove convergence. In an audit of 200 selected tasks per phase, novelty relative to earlier phases falls from 74% in Phase 1 to 8% in Phase 6, with near-duplicate rates rising from 8% to 68%. Novelty here uses maximum cross-phase embedding similarity below 0.85, and near-duplicates exceed 0.92. The number of newly identified error clusters falls from 31 to 2. These diagnostics motivate stopping due to diminishing curriculum coverage, while additional phases could still improve the policy. The original WebRL phase-8 result uses a larger interaction budget and is not evidence that the baseline converged.

Phase K: Environment Interaction and Data Collection. At the start of phase K, the current model RL-agent K is deployed on the task distribution Task Set for Phase K within the environment Env. This stage gathers diverse trajectories under the current policy's capabilities.

1. **Rollout Execution.** The agent performs interactive rollouts within Env, producing sequences of states, actions, and observations that reflect both successful behaviors and failure cases.

2. **Trace Processing and Buffer Integration.** Newly generated rollouts are passed to the *Trace Processing* module for annotation and filtering.
    - *Subgoal Annotation:* We utilize LLM-based **subgoal checkers**[5] to verify the completion status of intermediate objectives at every step. These binary checks are interpolated to generate dense **subgoal progress scores** (p_t^*), which serve as regression targets for the Potential Critic.
    - *Perplexity Filtering:* To ensure the actor learns from high-quality data, we evaluate the **perplexity** of the current actor policy on the collected traces. Trajectories exhibiting perplexity above a dynamic threshold are discarded as out-of-distribution or irrational behavior.
    - *Training Target Assignment:* The filtered data is processed to support the hierarchical optimization in Algorithm 2. First, the **Potential Critic** is trained on the progress labels p_t^*. Second, the **Main Value Critic** is updated using the sparse terminal outcomes (ORM labels). Finally, using the updated potential landscape, we compute shaped rewards and the resulting **Monte-Carlo (MC) returns** (G_t) to optimize the **Actor**. For each phase, we perform one epoch of RL training over the collected data.

    This produces an updated and temporally coherent **Training Data** corpus that captures the agent's evolving experience distribution while filtering out noise.

Offline RL Training: Model Patching via Batched Updates. The aggregated Training Data is then used to refine the model parameters through *offline batched RL*, effectively "patching" the agent between deployment rounds. The training phase minimizes the regression-style policy loss $\mathcal{L}_\pi(\theta)$ from Eq. (4), along with critic updates from Eqs. (2) and (3). The result is a newly optimized policy RL-agent $K+1$, which inherits prior knowledge while incorporating corrections from the latest rollout data. Offline RL offers two practical benefits: (1) it stabilizes learning by decoupling environment noise from gradient updates, and (2) it enables fine-grained policy improvements at scale by batching large numbers of stored trajectories, including previously failed ones, into a consistent optimization phase.

Online Curriculum Generation: Failure-Driven Task Evolution. Once the offline phase concludes, the system performs targeted task selection to maintain progressive learning difficulty:

1. **Task Generation (Task Gen.).** We utilize a pre-generated pool of 1,573 feasible tasks, authored by an LLM generator (Gemini-2.5-pro) seeded with domain descriptions and filtered for feasibility before use; generated tasks are further screened against the WebArena-Lite evaluation set to eliminate overlap. The Task Gen. module analyzes the Failure traces from Phase K and resamples tasks from this pool that exhibit high similarity to the failed instances. In practice, we use larger models such as Gemini-2.5-flash to identify semantic similarity, limit the top-K similar tasks per failure trace, and permit training-task reuse across phases. For rare or unbalanced task categories where the pool coverage is sparse (e.g., "Map" navigation), we synthesize new instructions by perturbing failure samples to ensure balanced distribution.
2. **Adaptive Phase Transition.** The resulting Task Set for Phase K+1 becomes the curriculum for the next phase. The refined agent RL-agent $K+1$ is then redeployed, closing the loop between environment feedback and policy improvement.

Iterative Curriculum Refinement. This iterative process forms a continual adaptation loop:

$$\text{Phase } K \xrightarrow[\text{Failures}]{\text{Task Gen.}} \text{Phase } K+1,$$

where each cycle explicitly leverages failure cases as learning signals. Over successive phases, the curriculum automatically evolves from simple to complex tasks, while the policy progressively internalizes successful behaviors through potential-shaped, subgoal-oriented offline RL. This cyclical design ensures that MiRA-RL maintains long-horizon learning efficiency while continually repairing and extending the model's competence boundary.

[5]We adopt few-shot based Gemini-2.5-pro and engineered prompts

Algorithm 1 MiRA Outer Loop: Curriculum Generation & Experience Replay

Input: Initial set I_0, task pool I_{pool}, SFT model $\mathcal{M}_{\text{SFT}}$, Envs, Reward Model $\mathcal{M}_{\text{ORM}}$
Initialize: $\pi_1 \leftarrow \mathcal{M}_{\text{SFT}}$, Critics V_ϕ, P_ψ, Buffer $\mathcal{B} \leftarrow \emptyset$, Failure Set $\mathcal{F} \leftarrow \emptyset$, $I_{\text{train}} \leftarrow I_0$
for Phase $k = 1$ to N **do**
    Collect trajectories D_{roll} via π_k on I_{train}; label successes with $\mathcal{M}_{\text{ORM}}$
    Split $D_{\text{roll}} \rightarrow D_{\text{success}} \cup D_{\text{fail}}$
    Update $\mathcal{B} \leftarrow \mathcal{B} \cup D_{\text{success}}$ and $\mathcal{F} \leftarrow \mathcal{F} \cup \{\text{instructions of } D_{\text{fail}}\}$
    Calculate ppl(τ) for $\tau \in \mathcal{B}$ using current π_k
    Filter $D_{\text{exp}} \leftarrow \left\{\tau \in \mathcal{B} \mid 1/0.9 \leq \text{ppl}(\tau) \leq 1/0.5\right\}$
    Combine data: $D_{\text{train}} \leftarrow D_{\text{roll}} \cup D_{\text{exp}}$
    $(\pi_{k+1}, V_\phi, P_\psi) \leftarrow$ MiRA-RL$(D_{\text{train}}, V_\phi, P_\psi, \pi_k, \dots)$
    $I_{\text{sim}} \leftarrow$ Resample(I_{pool}) based on semantic similarity to $\mathcal{F}$
    $I_{\text{syn}} \leftarrow$ Generate$(\mathcal{F})$ for rare/unbalanced failure types
    $I_{\text{train}} \leftarrow$ Select$(I_{\text{sim}} \cup I_{\text{syn}})$
end for

A.5 MiRA-RL Inner-Loop Algorithm

Algorithm 2 summarizes the inner-loop actor optimization with potential-based reward shaping. Three phases alternate per iteration: critic updates, reward shaping & return computation, and the MSE-regression actor update.

Algorithm 2 MiRA-RL: Actor Optimization with Potential-Based Reward Shaping (Inner Loop)

Require: Training batch $\mathcal{D}_{\text{train}}$; value critic V_ϕ; potential critic P_ψ; current policy π_θ; reference policy π_{ref}; discount γ; temperature β; mixing λ; learning rates η.

Initialize local optimization step counter $t \leftarrow 0$
while optimization not converged **do**
    Phase A: Critic updates
    for $i = 1$ to N_c **do**
        Sample minibatch $\mathcal{B}_c \subset \mathcal{D}_{\text{train}}$
        Update Value Critic $\mathcal{L}_V(\phi)$ via Eq. (3)
        Update Potential Critic $\mathcal{L}_P(\psi)$ via Eq. (2)
        $\phi \leftarrow \phi - \eta_V \nabla \mathcal{L}_V, \quad \psi \leftarrow \psi - \eta_P \nabla \mathcal{L}_P$
    end for
    Phase B: Reward Shaping & Returns
    Sample minibatch $\mathcal{B}_a \subset \mathcal{D}_{\text{train}}$
    for each trajectory $\tau \in \mathcal{B}_a$ **do**
        Calculate potential diffs: $r'_t \leftarrow r_t + \alpha(P_\psi(s_{t+1}, g) - P_\psi(s_t, g))$
        Compute returns G_t using shaped rewards r'_t
    end for
    Phase C: Actor Update
    for each transition in $\mathcal{B}_a$ **do**
        Compute Advantage A_t^{shaped} (Eq. 5) mixing Monte-Carlo and Critic
        Compute log-ratio ρ_t against π_{ref}
    end for
    Update $\theta \leftarrow \theta - \eta_\pi \nabla_\theta \mathcal{L}_\pi(\theta)$ (minimizing Eq. 4)
end while
return Updated parameters $(\pi_\theta, V_\phi, P_\psi)$

A.6 Experimental Details

In this section, we provide a detailed breakdown of the experimental setup, including the compute infrastructure, the specific prompts used for subgoal decomposition, the hyperparameters for all baseline comparisons, and the system instructions that govern agent behavior and evaluation. While our motivation study (Section 3.3) analyzes the full 812-task WebArena benchmark to characterize failure modes at scale, our primary evaluation uses the 165-task WebArena-Lite subset. SFT initialization

relies on separate cold-start tasks, and RL training uses generated tasks screened for overlap with the evaluation set.

Compute Infrastructure. All experiments were conducted on a single node equipped with 8 high-memory NVIDIA A100 GPUs (80GB each). A complete multi-phase training run of MiRA on Gemma3-12B—covering the potential critic warmup, the six-phase curriculum, and per-phase WebArena-Lite evaluation—takes approximately three days of wall-clock time on this setup ($\sim$580 GPU-hours per full run). Per-phase evaluation over the 165 WebArena-Lite tasks completes in roughly 40 minutes. Including the five main-result seeds, the four ablation variants (w/o PC, w. KL, w/o Doubly Adv., AWR), and hyperparameter sensitivity sweeps (α grid, perplexity filtering), the full research project consumed approximately 6,000 GPU-hours.

Additional evaluation results and WebRL reproducibility. On the full 812-task WebArena benchmark, Gemma3-12B SFT, WebRL, and MiRA achieve 18.8%, 24.0%, and 30.4% Pass@1 SR, respectively. The full suite contains difficult, unstable, and potentially infeasible cases; it is not a fully independent environment. On WebArena-Lite, Llama3.1-8B SFT, DigiRL, WebRL, and MiRA obtain 20.6$\pm$1.1, 30.3$\pm$1.4, 38.8$\pm$1.1, and 41.2$\pm$1.2% Pass@1 SR (run-level SE). Our 38.8% WebRL score evaluates the released checkpoint in a unified harness, while the original work reports 42.4% in its configuration. Repeated frozen-checkpoint evaluations in our harness give 37.6%, 38.8%, and 40.0%; repeated ORM labeling changes the aggregate SR by roughly 1–2 points. The matched Gemma3 comparison uses the same actor, SFT initialization, reset/evaluation setup, and six-phase training budget for WebRL and MiRA.

Inference cost. The subgoal generator, subgoal checker, and both critics are training-only: at deployment, the actor uses the same instruction/history/HTML action interface as matched baselines. Median episode-level input/output tokens and action counts (with interquartile ranges) are:

Table 4: Inference usage per episode: median [P25, P75].

Policy	Input tokens	Output tokens	Actions
Gemma3-SFT	74k [42k, 118k]	0.82k [0.51k, 1.31k]	10 [6, 16]
Gemma3-WebRL	86k [49k, 134k]	0.95k [0.60k, 1.45k]	12 [7, 18]
Gemma3-MiRA	82k [47k, 128k]	0.91k [0.58k, 1.38k]	11 [7, 17]

Subgoal Generation Examples and Decomposition. To bridge the gap between high-level user intents and low-level browser actions, MiRA employs a multimodal decomposition strategy. We utilize **Gemini-2.5-pro** as the teacher model due to its strong visual reasoning capabilities. As shown in Figure 15, the model is prompted with the task instruction and the initial webpage screenshot, and is strictly instructed to decompose the task into a fixed number of milestones (e.g., exactly 4 steps). This standardization ensures consistent temporal granularity across diverse tasks. Table 5 presents qualitative examples of these generated subgoals across various WebArena environments (Reddit, GitLab, CMS, etc.), demonstrating the model's ability to identify logical checkpoints such as "Maps to subreddit," "Search for user," and "Submit post."

Table 5: Examples of generated subgoals across WebArena domains.

Website	**User Intent**	**Generated Subgoals**
Reddit	Ask for product recommendations for running shoes within a budget of $100 in r/sports	1. Navigate to the "r/sports" subreddit. 2. Initiate the creation of a new post. 3. Write a title and body asking for shoes under $100. 4. Submit the post.

Continued on next page

Table 5 – continued from previous page

Website	User Intent	Generated Subgoals
	Like all submissions created by CameronKelsey in subreddit earthporn	1. Navigate to the subreddit 'earthporn'. 2. Search for posts by user 'CameronKelsey'. 3. Click the 'like' button on the first post. 4. Repeat for all other visible posts.
Gitlab	Create a new public project "awesome-llms" and add primer, convexegg, abishek as members	1. Navigate to new project creation page. 2. Enter "awesome-llms", set visibility to public. 3. Finalize project creation. 4. Add primer, convexegg, and abishek in members page.
	Display issues in `kkroening and ffmpeg-python` with labels related to questions	1. Navigate to `kkroening/ffmpeg-python` repo. 2. Go to the "Issues" tab. 3. Open "Labels" filter menu. 4. Select label related to "questions".
Shopping Admin (CMS)	Tell me the grand total of invoice 000000002	1. Navigate to sales/invoices section. 2. Search for invoice 000000002. 3. Open details for the invoice. 4. Identify grand total amount.
	Tell me the status of my latest order and when will it arrive	1. Navigate to "My Account" or "Profile". 2. Go to "Orders" or "Order History". 3. Identify most recent order. 4. Extract status and estimated arrival date.
Map	Min travel time by car from Animal Rescue League of Pittsburgh to Schenley park?	1. Navigate to map website, access directions. 2. Enter "Animal Rescue League" as start. 3. Enter "Schenley park" as destination. 4. Select car mode, find min travel time route.
	Tell me the coordinates of Apple Store near Pitt in DD format	1. Search for "Pitt" on map website. 2. Search for "Apple Store" nearby. 3. Select the correct Apple Store. 4. Obtain coordinates in DD format.
Shopping (OSS)	Look up most recent models of XBox controllers released 2020-2021?	1. Search 'Xbox controller' on Wikipedia. 2. Navigate to main page. 3. Locate 'Models' or 'History' section. 4. Identify controllers released 2020-2021.
	Show least expensive switch card holder with capacity of 15+ cards.	1. Search for "switch card holder". 2. Filter for capacity of 15 or more. 3. Sort results by price (low to high). 4. Select first item.

Hyperparameter Configuration To ensure a fair comparison, we align our hyperparameters with established baselines wherever possible. Table 6 lists the training configurations for SFT, AWR, DigiRL, WebRL, and MiRA. For the actor and critic networks, we maintain a consistent learning rate of 1×10^{-6} and a batch size of 128 across all RL methods. A key distinction for MiRA is the

Table 6: Hyperparameters used in baseline methods.

Method	Hyperparameter	Value
SFT	learning rate	1×10^{-5}
	lr scheduler type	cosine
	warmup ratio	0.1
	batch size	128
	training epoch	1
	cutoff length	16384
AWR	actor learning rate	1×10^{-6}
	actor lr scheduler type	constant
	critic learning rate	1×10^{-6}
	critic lr scheduler type	constant
	batch size	128
	discount factor	0.9
	actor training epoch	1
	critic training epoch	1
DigiRL	actor learning rate	1×10^{-6}
	actor lr scheduler type	constant
	critic learning rate	1×10^{-6}
	critic lr scheduler type	constant
	instruction value function lr	1×10^{-6}
	instr. value function scheduler	constant
	batch size	128
	discount factor	0.9
	actor training epoch	1
	critic training epoch	1
	instruction value function epoch	1
	rollout temperature	1
	replay buffer size	100,000
WebRL	actor learning rate	1×10^{-6}
	actor lr scheduler type	constant
	critic learning rate	1×10^{-6}
	critic lr scheduler type	constant
	batch size	128
	discount factor	0.9
	actor training epoch	1
	critic training epoch	1
	rollout temperature	1
MiRA	actor learning rate	1×10^{-6}
	actor lr scheduler type	constant
	critic learning rate	1×10^{-6}
	critic lr scheduler type	constant
	potential critic learning rate	2×10^{-5}
	potential critic lr scheduler type	constant
	potential critic training epoch	3
	batch size	128
	discount factor	0.9
	actor training epoch	1
	critic training epoch	1
	rollout temperature	1

Potential Critic, which uses a higher learning rate of 2×10^{-5} and is trained for 3 epochs. This adjustment is necessary because the Potential Critic performs a regression task (fitting dense progress scores) rather than standard value estimation, requiring stronger gradient updates to capture the subtle shape of the progress landscape effectively.

Agent Input

Task Instruction: Like all submissions created by CameronKelsey in subreddit earthporn.

Round 0

user:
Like all submissions created by CameronKelsey in subreddit earthporn.

assistant:
# Element: the 'Submissions' tab on the user profile navigation
do(action="Click", element="12")

Round 1

user:
* Simplified html **

assistant:
# Element: Scrolling down to find the specific post in the feed
do(action="Scroll", direction="down")

Round 2

user:

<html data-bbox="0,0,1280,720"><body data-bbox="0,0,1280,720"> <main
↪ data-bbox="90,66,806,399"><ul data-bbox="104,80,778,35">Overview<a id="12" data-text="
↪ Submissions">Submissions <header><strong id="27" data-text="
↪ CameronKelsey">CameronKelseyEarthPorn</header><form><button title="
↪ Upvote" value="1" type="submit"><use id="30" data-bbox="122,360,12,13"></use></button>12678</form></
↪ div></main> </body></html>

assistant:

Agent Output

# Element: the upvote button next to the number 12678
do(action="Click", element="30")

Figure 11: The input and output format of MiRA and baselines, where the input is composed of task instruction (in green), action history (in blue), and HTML of the current webpage (in orange). The output (in red) is the action taken on the current webpage.

System Prompts for Execution and Evaluation The performance of web agents is heavily influenced by the precise formatting of their inputs and allowed outputs. Figure 11 visualizes the data flow, where the agent receives the task instruction, action history, and current HTML. To constrain the agent's output to valid executable actions, we employ the system prompt detailed in Figure 12 for small open-weight models (e.g., Llama-3, Gemma). This prompt restricts responses to a predefined Python-style DSL (e.g., `do(...)`, `click(...)`) and explicitly forbids the hallucination of non-visible elements.

In contrast, when evaluating large proprietary models (e.g., GPT-4o, Gemini-2.5-Pro), we adopt the standard text-based accessibility tree prompt shown in Figure 14. This configuration aligns with the default WebArena benchmark settings and prior works [Chae et al., 2025, Qi et al., 2024, Wei et al., 2025], utilizing the stronger instruction-following capabilities of frontier models to parse structured element lists rather than simplified HTML. Finally, for reward modeling, we utilize the Outcome Reward Model (ORM) prompt shown in Figure 13. This prompt instructs the judge to evaluate success based on the alignment between the user intent and the final page state, serving as the ground-truth signal for filtering successful trajectories during the warmup phase.

A.7 MiRA Agent Design and Rationale

A.7.1 Observation and Action Space

The MiRA agent operates within a structured environment designed to bridge the gap between high-level user instructions and low-level DOM interactions. The agent's observation space consists of three key components:

- **User Instruction:** The natural language goal provided by the user.
- **Action History:** A chronological record of past actions, providing essential context for tracking subgoal progress.

System Prompt Used in small model baselines incl. MiRA

 # Setup
You are a professional web browsing agent assistant that can fulfill user's high-level instructions. Given simplified HTML of the browsed webpage at each step, you plan operations in Python-style pseudo code using provided functions, or customize functions if necessary.

 # Predefined Functions

```
def do(action, argument, element):
    """A single browsing operation on the webpage."""
def exit(message):
    """End the browsing process."""
def go_backward():
    """Go back to the previous page."""
def go_forward():
    """Go forward to the next page."""
```

Allowed actions: Click, Right Click, Type, Search, Hover, Scroll Up, Scroll Down, Press Enter, Switch Tab, Select Dropdown Option, Wait.

Examples:

```
# Element: the 'REPORTS' section on the left sidebar
do(action="Click", element="7")

# Element: the 'Period' dropdown
do(action="Select Dropdown Option", argument="Month", element="20")
```

Rules:
- Only **one line of code** at a time.
- Do not invent non-visible elements.
- Use "# Element" comments to indicate the chosen element.
- Avoid loops; use if-else only when necessary.
- If the page is loading, prefer "Wait".
- Never use the browser address bar to navigate.

User Prompt Template:

```
Task Instruction: {instruction}
Action History: {action history}
The Current HTML: {html of last state}
```

Figure 12: The simplified system prompt employed in baseline agents.

System Prompt for Outcome Reward Model (ORM)

 # Role
You are an expert in evaluating the performance of a website navigation agent. The agent is designed to help a human user navigate the website to complete a task. Given the user's intent, the agent's action history, and the final state of the screen, your goal is to decide whether the agent's execution is successful or not. You must respond with YES or NO.

 # User Prompt Template

```
The User Intent:
{task instruction}

Action History:
{history of agent actions}

The Current Screenshot or Page:
{simplified html/screenshots of final state}
```

Figure 13: The system prompt used for the Outcome Reward Model (ORM) to verify trajectory success.

System Prompt for Proprietary Models (GPT-4 / Gemini)

 # Role
You are an autonomous intelligent agent tasked with navigating a web browser. You will be given web-based tasks and must accomplish them by issuing specific actions.

 # Input Information
- **Objective:** The user's task description.
- **Observation:** A simplified accessibility tree representation of the webpage.
- **Current URL:** The page you are currently navigating.
- **Previous Action:** The action you just performed.

 # Available Actions

```
click [id]                  # Click on element with specific ID
type [id] [content] [0|1]   # Type content (0=no enter, 1=enter)
hover [id]                  # Hover over element
press [key_comb]            # Simulate key combination (e.g., "Ctrl+v")
scroll [up|down]            # Scroll the page
new_tab, close_tab          # Manage tabs
tab_focus [index]           # Switch to tab by index
goto [url]                  # Navigate to URL
go_back, go_forward         # History navigation
stop [answer]               # Complete task (with optional answer)
```

 # Response Format Your response must always start with the phrase "In summary, the next action I will perform is" followed by the action inside triple backticks.

 # Examples

```
OBSERVATION:
[1744] link 'HP Inkjet Fax Machine'
[1757] button 'Add to Cart'
URL: http://onestopmarket.com
OBJECTIVE: What is the price of the Fax Machine?

RESPONSE:
In summary, the next action I will perform is ```stop [$279.49]```
```

```
OBSERVATION:
[164] textbox 'Search'
[171] button 'Go'
URL: http://openstreetmap.org
OBJECTIVE: Show me hotels in Pittsburgh

RESPONSE:
In summary, the next action I will perform is
```type [164] [hotels in Pittsburgh] [1]```
```

Figure 14: The standard system prompt (Set-of-Marks Style) used for large proprietary models (e.g., GPT-4, Gemini) in WebArena, utilizing a text-based accessibility tree observation space.

- **Webpage HTML:** A simplified representation of the current webpage's HTML. To facilitate precise manipulation, we prune the DOM structure and assign unique element IDs to all interactive candidates.

The action space is designed to be granular yet comprehensive, enabling the execution of complex subgoals. The defined actions include:

- **Interaction:** *Click*, *Hover*, *Type*, *Search* (type and enter), *Select dropdown option*.
- **Navigation:** *Goto*, *Go back*, *Go forward*, *Scroll*.
- **Tab Management:** *New tab*, *Tab focus*, *Close tab*.
- **System:** *Press* (key combinations), *Exit* (terminate and respond).

To enhance interpretability and grounding, the agent appends metadata to its actions: a `# Element:` tag describing the target DOM element, and a `# Note:` tag citing supporting information from the webpage content.
```

Prompt for Subgoal Generation

 # Role

You are an assistant inside the WebArena benchmark environment. WebArena simulates websites such as GitLab, Wikipedia, shopping portals, and admin dashboards. Tasks are solved only through browser interactions (clicks, typing, navigation)—never by recalling outside world knowledge. Your job is to decompose each task intent into EXACTLY 4 concrete, high-level subtasks. Always provide exactly 4 numbered steps, no more and no less.

 # Example 1: GitLab

```
Task: Create a private JEKYLL repository called 'awesome_project'.
Screenshots: [Image1], [Image2], [Image3]

Answer:
1. Access GitLab and locate 'Create New'
2. Select the JEKYLL template from the template menu
3. Enter repo name 'awesome_project' and set it private
4. Create repository and confirm landing on its homepage
```

 # Example 2: Map/Wiki

```
Task: Find the page of the college(s) where *The Chair* was filmed in
Pennsylvania other than those in Pittsburgh on the map.
Screenshots: [Image1], [Image2], [Image3]

Answer:
1. Identify correct Wikipedia page for *The Chair* (2021 TV series) and
   locate filming colleges in Pennsylvania (excluding Pittsburgh)
2. Copy the college name from the Wikipedia page
3. Open the map website and search the copied name
4. Navigate to the college's location on the map
```

 # User Prompt Template

```
=====Your Turn=====
Q: TASK: {task_intent}
Screenshot: {task_initial_screenshot}
Decompose into exactly 4 milestones.
A: <SUBGOALS>
```

Figure 15: The multimodal prompt used to decompose tasks into four structured subgoals (Taking fixed 4 subgoals in total as example) via Gemini-2.5-pro.

A.7.2 Training Framework and Rationale

The design of the MiRA training pipeline addresses the challenge of learning robust policies from sparse rewards. The process follows a self-evolving curriculum:

Initialization via Supervised Fine-Tuning (SFT): We establish a baseline web-navigation capability by initializing the actor from an SFT checkpoint trained on a separate set of cold-start tasks, strictly disjoint from the WebArena-Lite evaluation tasks. This checkpoint also initializes the backbone of the critics (with newly initialized value heads), and the policy populates the initial training replay and failure sets. For each backbone, all matched RL methods use the same SFT initialization.

Self-Evolving Curriculum Reinforcement Learning: To progressively enhance performance, we employ a curriculum-based reinforcement learning approach. In each phase, we sample up to 400 new instructions from the LLM-generated task pool, with Gemini-2.5-flash filtering for complexity criteria consistent with the agent's current capability frontier. To prevent train-evaluation overlap, we explicitly exclude any generated instruction whose semantic similarity to any WebArena-Lite evaluation task exceeds a pre-defined threshold, computed via sentence-level embedding similarity. The actor and critic are updated using a hybrid data mixture:

1. Newly generated interaction trajectories from the current curriculum phase.
2. Historical high-quality data from the replay buffer, specifically filtering for trajectories with perplexity between $1/0.9$ and $1/0.5$.

This dual-source training strategy ensures the agent retains previously learned behaviors while adapting to increasingly complex subgoals.

A.8 Extra Ablation Studies

Table 7: Impact of replay buffer filtering based on perplexity (PPL) and rank score ($1/\mathrm{PPL}$).

Rank Score ($1/\mathrm{PPL}$)	$(0.9, 1.0]$	$[\mathbf{0.5}, \mathbf{0.9}]$	$[0.0, 0.5)$	$[0.0, 1.0]$
PPL Range	$[1, \frac{1}{0.9})$	$[\frac{\mathbf{1}}{\mathbf{0.9}}, \frac{\mathbf{1}}{\mathbf{0.5}}]$	$(\frac{1}{0.5}, \infty)$	Full Range
SR (%)	27.9	**36.4**	23.6	29.1

The impact of reward shaping (α). We perform a grid search over $\alpha \in [0.0, 0.8]$ using a held-out validation set of WebArena-Lite tasks. As shown in Table 8, very small shaping factors (0.0–0.1) behave similarly to the sparse-reward baseline, yielding only 30.9–31.5% success. In this regime, the auxiliary signal is too weak to overcome reward sparsity and thus behaves largely as noise.

Performance peaks at $\alpha = 0.3$, achieving **36.4%**, confirming that moderate shaping effectively accelerates credit assignment without drowning out the terminal reward. However, as α increases (0.5–0.8), performance drops sharply to 28.5–25.5%. Large shaping factors distort the reward landscape, causing the agent to overfit to auxiliary subgoal progress rather than true task completion. These findings highlight the necessity of calibrated shaping magnitudes.

The impact of perplexity filtering. Table 7 evaluates how different perplexity bands affect data usefulness. Training exclusively on low-perplexity (highly predictable) samples yields only 27.9% success, suggesting limited learning value from "too easy" trajectories. Conversely, using only the high-perplexity tail produces the worst performance (23.6%), indicating that noisy or out-of-distribution states destabilize the value critic.

Table 8: Impact of reward shaping factor α.

α	0.0	0.1	**0.3**	0.5	0.8
SR (%)	30.9	31.5	**36.4**	28.5	25.5

The best performance arises from filtering to the moderate-perplexity range $[1/0.9, 1/0.5]$ (i.e., rank score $[0.5, 0.9]$), achieving **36.4%**. These "borderline-difficult" transitions provide the most informative gradients—neither trivial nor erratic—leading to more robust and sample-efficient learning.

Critic-target decoupling. The following controlled study compares outcome-only, entangled shaped-return, and MiRA's decoupled outcome/potential critic targets. The AUROC measures *terminal-outcome prediction by the value critic*, not the AUROC of graded subgoal completion reported in Section 4.1.

Table 9: Outcome-critic target ablation (reported controlled-study SR).

Target design	Outcome-value AUROC	SR (%)
Outcome only, no progress	0.78	30.9
Entangled shaped-return critic	0.70	32.8
Decoupled MiRA	**0.81**	**36.4**

Subgoal reliability versus executor and granularity. Holding the subgoal definition and checker fixed, stronger executors more reliably reach intended milestones; varying the number of milestones illustrates the precision–recall trade-off. These analyses evaluate milestone–success agreement, not a head-to-head test of alternate subgoal generators.

Interpretation of shaping sensitivity. The 36.4% values in these controlled sensitivity studies are not the six-phase main-result figure (43.0%); the main-result and sensitivity settings should not be conflated. The α sweep already reported above shows a peak at $\alpha = 0.3$ and performance degradation at larger weights, as expected for an imperfect auxiliary objective.

Table 10: Executor comparison: same milestone definition and checker.

Executor	Exact F1	Precision	Recall	Graded AUROC
Llama3.1-8B	0.61	0.74	0.52	0.79
Gemma3-12B	0.68	0.79	0.60	0.84
Gemini-2.5-Pro	0.82	0.90	0.76	0.91

Table 11: Number of milestones on Gemma3-12B: calibration trade-off.

Milestones	Exact F1	Precision	Recall	Graded AUROC
3	0.66	0.73	0.61	0.80
4	0.68	0.79	0.60	0.84
5	0.69	0.82	0.60	0.86
6	0.67	0.85	0.56	0.87

A.9 Potential Critic Training via Supervised Progress Regression

Algorithm 3 Training the Potential Critic P_ψ

Require: Dataset of trajectories $\mathcal{D}$; goal set $\mathcal{G} = \{g_1, \ldots, g_K, g_{\text{final}}\}$
Ensure: Trained potential critic $P_\psi(s, g)$

$\mathcal{S} \leftarrow \emptyset$ ▷ Supervision set for regression
for each trajectory $\tau \in \mathcal{D}$ **do**
  if trajectory does *not* reach the final goal **then**
    continue ▷ Use only positive traces to ensure correlation with success
  end if
  Extract subgoal-completion vector $\mathbf{z}_t$ for all t
  Identify key steps $t_1, \ldots, t_K$ where cumulative completion increases
  Compute continuous progress labels p_t^* by linear interpolation:

$$p_t^* = (1 - \alpha_t)\tfrac{j}{K} + \alpha_t \tfrac{j+1}{K}, \quad \alpha_t = \frac{t - t_j}{t_{j+1} - t_j}$$

  with the final segment anchored to the true trajectory end T.
  for each timestep t **do**
    Construct progress-aware state $s_t = (h_{1:t-1}, o_t, g)$
    Add (s_t, g, p_t^*) to $\mathcal{S}$
  end for
end for

/* Supervised learning */
Train the potential critic via MSE regression:

$$\mathcal{L}_P(\psi) = \frac{1}{|\mathcal{S}|} \sum_{(s_t, g, p_t^*) \in \mathcal{S}} \left(P_\psi(s_t, g) - p_t^*\right)^2 .$$

Return trained P_ψ.

A central challenge in long-horizon web tasks is that the environment provides only a sparse terminal reward—typically a single binary success or failure at the end of a multi-step trajectory. Such extreme sparsity makes it difficult for an RL agent to learn meaningful credit assignment, particularly when intermediate actions influence success in subtle, delayed ways. To mitigate this, MiRA introduces a *potential critic* $P_\psi(s, g)$ that predicts a dense, continuous notion of task progress. Instead of relying on handcrafted shaping potentials, which are notoriously brittle in heterogeneous environments like WebArena, the potential critic is learned directly from data via supervised regression, yielding a shaping landscape that reflects the procedural structure of successful trajectories.

The construction of the regression targets is grounded in the SubGoal Checker, which emits a binary subgoal-completion vector $\mathbf{z}_t$ at each timestep. From this signal, we detect the key timesteps $t_1, t_2, \ldots, t_K$ at which cumulative progress increases. Between any pair of these key steps, we interpolate the progress value linearly so that the label p_t^* increases smoothly from j/K to $(j+1)/K$. This interpolation converts the discrete, staircase-like profile of subgoal completions into a continuous supervision signal defined over the entire trajectory. For the final segment, we intentionally anchor the interpolation interval to the true end of the trajectory rather than the moment the final subgoal is detected; this ensures that the learned potential continues to rise during late-stage "cleanup" actions such as verification, scrolling, and form submission, which contribute meaningfully to task resolution despite not triggering additional subgoal boundaries.

An important modeling decision is to fit the potential critic exclusively on *positive* trajectories—those that ultimately reach the final goal. This ensures that the progress targets are always aligned with successful procedural structure. Using failed trajectories as supervision would dilute this relationship, as these episodes often terminate prematurely or contain exploratory detours that do not correspond to genuine forward progress. Restricting the regression dataset to successful episodes produces a potential function that implicitly captures the common semantic subsequences shared across successful attempts, creating a shaping landscape that rises precisely along paths known to lead to final success.

Once progress labels p_t^* have been computed, the potential critic is trained to regress these values using a standard MSE objective:

$$\mathcal{L}_P(\psi) = \mathbb{E}_{(s_t, g, p_t^*)}\big[\|P_\psi(s_t, g) - p_t^*\|^2\big].$$

The critic is implemented by augmenting a pretrained LLM with a regression head, enabling it to leverage both the semantic structure of the goal instruction and the action–observation history of the trajectory. Because the target labels are smooth and monotonic, optimization is stable, and the learned potential landscape provides a faithful surrogate of distance-to-go long before RL finetuning begins.

During reinforcement learning, the potential critic contributes an auxiliary shaping reward of the form

$$r_t^{(\text{aux})} = \alpha\big[P_\psi(s_{t+1}, g) - P_\psi(s_t, g)\big],$$

which is added to the environmental reward when computing advantage targets. This shaping term effectively supplies a local gradient that encourages the agent to continue moving "forward" in the learned progress space. The primary value critic V_ϕ remains trained on terminal outcomes, so it measures a distinct quantity from the potential critic. This separation anchors the actor surrogate to task success but does not ensure policy invariance: the auxiliary term can change the learned policy. Its usefulness and limitations are assessed in the shaping-weight and critic-target ablations in Appendix A.8.

Trajectory Selection and Quality Filtering. To ensure the potential critic learns from robust and efficient behaviors, we apply strict quality controls to the gathered rollouts. First, we utilize an Outcome Reward Model (ORM) trained via WebRL [Qi et al., 2024] to verify the success of each trajectory collected from both the WebRL (Llama3-8b) and baseline RL agents (initialized from the same Gemma SFT checkpoint as MiRA). The task set remains identical to that used in the multi-phase running on the baseline methods. We then filter the verified successes to eliminate signs of suboptimal agency: we remove trajectories containing repetitive loops (defined as identical actions repeated more than 5 times) and exclude inefficient rollouts by retaining only those with a total length of less than 15 steps. This curation process yields a dataset of positive execution traces that represent more direct and logical paths to task completion, ensuring the learned potential landscape encourages efficiency rather than circuitous wandering. Then the potential critic continues to be trained jointly with the main critic and actor.

In summary, supervised progress regression allows MiRA to construct a dense potential landscape grounded in the structure of successful trajectories. This landscape provides smooth intermediate rewards that fill the gaps left by sparse feedback, enabling efficient learning of long-horizon procedural behaviors that would otherwise be inaccessible to standard reinforcement learning.

A.10 Derivation of the Optimal Policy in KL-Regularized RL

We start with the objective function for KL-regularized Reinforcement Learning (commonly used in RLHF). The goal is to maximize the expected reward while minimizing the KL-divergence from a reference policy π_{ref}:

$$\max_{\pi} \mathbb{E}_{\pi} \left[r(s,a) - \beta \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)} \right] \tag{8}$$

The closed-form solution for the optimal policy π^* is given by the Boltzmann distribution:

$$\pi^*(a|s) = \frac{1}{Z(s)} \pi_{\text{ref}}(a|s) \exp\left(\frac{Q^*(s,a)}{\beta} \right) \tag{9}$$

where $Z(s)$ is the partition function (normalization constant). In the maximum entropy framework, the optimal value function $V^*(s)$ relates to the partition function as follows:

$$V^*(s) = \beta \log Z(s) \implies Z(s) = \exp\left(\frac{V^*(s)}{\beta} \right) \tag{10}$$

Recall the definition of the Advantage function: $A^*(s,a) = Q^*(s,a) - V^*(s)$. We can substitute $Q^*(s,a) = A^*(s,a) + V^*(s)$ back into the policy equation:

$$\begin{aligned} \pi^*(a|s) &= \frac{1}{Z(s)} \pi_{\text{ref}}(a|s) \exp\left(\frac{A^*(s,a) + V^*(s)}{\beta} \right) \\ &= \frac{1}{Z(s)} \pi_{\text{ref}}(a|s) \exp\left(\frac{A^*(s,a)}{\beta} \right) \exp\left(\frac{V^*(s)}{\beta} \right) \end{aligned}$$

Substituting $Z(s) = \exp\left(\frac{V^*(s)}{\beta}\right)$, the normalization terms cancel out:

$$\begin{aligned} \pi^*(a|s) &= \frac{1}{\exp\left(\frac{V^*(s)}{\beta}\right)} \pi_{\text{ref}}(a|s) \exp\left(\frac{A^*(s,a)}{\beta} \right) \exp\left(\frac{V^*(s)}{\beta} \right) \\ \pi^*(a|s) &= \pi_{\text{ref}}(a|s) \exp\left(\frac{1}{\beta} A^*(s,a) \right) \end{aligned}$$

A.11 KL Divergence Formulation for Policy Update

Many prior studies use KL divergence to measure the distance between two policies. When KL divergence is used to optimize π_θ toward π^*, the optimization goal becomes:

$$\begin{aligned} &\arg\min_{\theta} \mathbb{E}_{s\sim d(s)} \left[D_{\text{KL}} \left(\pi^*(\cdot|s,I) \,\|\, \pi_\theta(\cdot|s,I) \right) \right] \\ &= \arg\max_{\theta} \mathbb{E}_{s\sim d(s)} \mathbb{E}_{a\sim\pi^*(a|s,I)} \left[\log \pi_\theta(a|s,I) \right] &(11) \\ &= \arg\max_{\theta} \mathbb{E}_{s\sim d(s)} \int_a \pi_{\text{ref}}(a|s,I) \exp\left(\frac{1}{\beta} A^*(s,a,I) \right) \log \pi_\theta(a|s,I) \, \mathrm{d}a &(12) \\ &= \arg\max_{\theta} \mathbb{E}_{s\sim d(s)} \mathbb{E}_{a\sim\pi_{\text{ref}}(a|s,I)} \left[\log \pi_\theta(a|s,I) \cdot \exp\left(\frac{1}{\beta} A^*(s,a,I) \right) \right], &(13) \end{aligned}$$

where $d(s)$ is a distribution over states, and we have substituted the closed-form optimal policy $\pi^*(a|s,I) = \pi_{\text{ref}}(a|s,I) \exp\left(\frac{1}{\beta} A^*(s,a,I)\right)$ from Appendix A.10.

Direct optimization of the forward KL objective requires expectations under the target policy or appropriate importance weighting. This can be challenging when learning from heterogeneous off-policy replay data. In contrast, our MSE log-ratio objective (Eq. 4) directly regresses signed surrogate targets on replay samples, allowing both positive and negative adjustments to sampled action log-probabilities. This provides a practical alternative for our off-policy training setting, although it does not guarantee exact policy improvement.

A.12 Extended Discussion

In this work, we addressed the critical planning bottleneck in autonomous web agents by introducing MiRA, a unified framework that synergizes explicit milestone-based reasoning with potential-based reinforcement learning. By transforming high-level intent into verifiable subgoals, we demonstrated that open-source RL policies can overcome the horizon truncation failures pervasive in current benchmarks. Beyond the empirical gains on WebArena-Lite, our findings open several avenues for discussion regarding the nature of reward shaping, the reliability of intermediate supervision, and the future of self-evolving autonomy.

Connection to Process Reward Models (PRM). Conceptually, our training methodology can be viewed as a form of semi-supervised Process Reward Model (PRM) learning. Unlike traditional PRMs that rely on expensive human annotations for every step of a chain-of-thought, MiRA synthesizes its own "process supervision" by generating semantic subgoals via an advanced teacher model and grounding them in execution traces. The potential critic P_ψ effectively acts as a learned value function over these synthesized milestones, densifying the sparse outcome signal. This aligns with recent findings in mathematical reasoning [Lightman et al., 2023], suggesting that verifying intermediate steps (subgoals) is often easier and more sample-efficient than verifying final answers alone. Our work extends this principle to the noisy, non-stationary domain of web navigation, proving that automated process supervision is a viable path to scaling RL for complex agents.

The "Solid Grounding" Trade-off. The efficacy of our potential-based shaping hinges entirely on the validity of the generated subgoals. As shown in our ablation studies, rewarding **solid** subgoals—those the agent can robustly perceive and achieve—creates a reliable curriculum, forming a regression curve of progress (e.g., in Gitlab tasks) that pulls the agent deeper into the episode. However, this reliance introduces a subtle *amplification bias*. Because the auxiliary reward is triggered only upon successful grounding of a milestone, the system heavily favors trajectories that master the early stages of a task. If an agent fails to ground the initial subgoals due to extreme exploration difficulty or perception errors, the shaping signal remains silent, effectively reverting the optimization to a sparse-reward regime. Thus, while MiRA excels at extending the "competence boundary" of agents that can start a task, it implies that future work must still address the cold start exploration problem in environments where even the first milestone is hard to reach.

Towards Self-Evolving Autonomous Agents. Perhaps the most compelling implication of our results is the feasibility of a closed-loop, self-improving training cycle. In our experiments, a small ensemble of capable models collectively played complementary roles: a teacher model (Gemini-2.5-pro) generated and verified subgoals, a lightweight model (Gemini-2.5-flash) handled curriculum filtering and similarity matching, an ORM judged terminal success, and the trained agent (Gemma3-12B) executed and improved across phases. Looking forward, consolidating these roles into a single self-improving model is a natural next step toward truly **self-evolving agents** capable of recursive improvement. By integrating the planning, diagnosing, and training processes into a singular, cohesive loop, future systems could autonomously continually expand their capabilities without human intervention, transforming the static "train-then-deploy" paradigm into a lifelong learning process.

Limitations and broader impacts. Our experiments cover both the curated WebArena-Lite tasks and the broader WebArena suite, with MiRA applied to Gemma3-12B and Llama3.1-8B. These results do not establish transfer to independent websites, task distributions, or model families beyond those tested. The limited exact-equivalence F1 shows that milestones can miss successful alternative paths or introduce noisy supervision, and high shaping weights can harm final success. The subgoal generation pipeline depends on a capable teacher model, which concentrates methodological trust

in proprietary systems and may introduce distributional bias from that teacher's training data. Web-navigation agents in general carry dual-use potential: while they can reduce repetitive labor and improve accessibility, the same capabilities could enable scraping, spam generation, or circumvention of access controls. We run experiments in sandboxed WebArena environments without interaction with live websites.

Future Directions. While our framework establishes a strong baseline, several dimensions warrant further exploration to generalize this approach. First, regarding milestone synthesis: we aim to transition from heuristic prompts to *learnable or hierarchical subgoal generators*. Such models could dynamically tailor the granularity of milestones for knowledge-sparse domains where static linear ramps are insufficient. Second, we see promise in refining the shaping mechanism itself; future work should explore *non-linear progress estimation* that accounts for the varying difficulty of distinct subgoals rather than treating them uniformly. Finally, we propose investigating *signal annealing strategies*. In this regime, subgoals would serve as temporary "warm-up" scaffolding that is gradually withdrawn as training progresses, ensuring that the final policy relies on the true task objective rather than over-optimizing for auxiliary rewards.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 explicitly state the two contributions: (1) a validated subgoal generation procedure with AUROC 0.84 and Kendall's $\tau = 0.4585$ (Section 4.1), and (2) the MiRA RL fine-tuning framework (Section 4.2). The reported success rate gains (Gemma3-12B: $6.4\% \rightarrow 43.0\%$) match the experimental results in Table 1, and all comparisons are scoped to the WebArena-Lite benchmark.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Extended Discussion (Appendix A.12), including the "Solid Grounding" trade-off where shaping signals require reachable initial subgoals (leading to cold-start issues in high-difficulty settings), the reliance on a capable teacher model for subgoal generation, and the current use of linear progress interpolation as opposed to non-linear difficulty-aware estimation.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper's main theoretical claim—the equivalence between Potential-Based Reward Shaping and Φ-initialization in the goal-conditioned setting (Section 3)—is accompanied by a full induction proof in Appendix A.1, adapted from Wiewiora [2003]. The derivation of the optimal policy form π^* used in the MSE objective is given in Appendix A.10, and the MSE-vs-KL comparison derivation is in Appendix A.11. All assumptions (goal-conditioned MDP, state-only potential function, $\arg\max$-invariance of constant shifts) are stated explicitly.

4. **Experimental result reproducibility**

   Answer: [Yes]

   Justification: The training pipeline (curriculum outer loop in Algorithm 1, inner MiRA-RL loop in Algorithm 2, potential critic training in Algorithm 3), the base model (Gemma3-12B), the benchmark (WebArena-Lite, 165 tasks), the system prompts (Appendix A.6, Figures 12, 13, 14, 15), and all hyperparameters (Table 6) are fully documented. The subgoal generation procedure uses publicly described prompting with Gemini-2.5-pro. All baselines (DigiRL, WebRL, AWR) are reproduced on the same SFT checkpoint for fair comparison.

5. **Open access to data and code**

   Answer: [No]

   Justification: The training code and model checkpoints are not released at submission time due to ongoing internal review. The benchmark is publicly available at https://github.com/THUDM/VisualAgentBench,https://github.com/web-arena-x/webarena, and the base models (Gemma3-12B, Llama3.1-8B) are publicly released by their respective providers. We provide full algorithmic descriptions, hyperparameters, and prompts in the appendix to enable independent reimplementation.

6. **Experimental setting/details**

   Answer: [Yes]

   Justification: Full experimental details are provided: the training protocol (Section 5.1) describes initialization from shared SFT checkpoints, the 6-round curriculum for DigiRL/WebRL/MiRA, and the evaluation setup; Appendix A.6 contains the complete hyperparameter table (Table 6) covering learning rates, schedulers, batch sizes, discount factors,

and training epochs for all methods (SFT, AWR, DigiRL, WebRL, MiRA); observation and action spaces are specified in Appendix A.7; filtering thresholds (perplexity range $[1/0.9, 1/0.5]$, $\alpha = 0.3$ for shaping) are reported alongside their ablations in Appendix A.8.

7. **Experiments compute resources**

   Answer: [Yes]

   Justification: Compute infrastructure details are provided in Appendix A.6: all experiments were conducted on a single node of 8 high-memory NVIDIA A100 GPUs (80GB each). A complete multi-phase MiRA training run on Gemma3-12B takes approximately three days of wall-clock time, and full WebArena-Lite evaluation takes $\sim$40 minutes per run. Ablation and multi-seed experiments were scheduled on the same hardware.

8. **Code of ethics**

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. The work uses publicly released models (Gemma3-12B, Llama3.1-8B) and a public benchmark (WebArena-Lite) built on self-hosted sandboxed web environments. No human subjects, no crowd-sourced annotations beyond the 40-example validation performed by the authors, and no deployed system interacting with real users or external websites.

9. **Broader impacts**

   Answer: [Yes]

   Justification: Potential positive impacts include improved accessibility tooling for users with disabilities and reduced manual overhead in repetitive web tasks. Potential negative impacts include misuse of automated web agents for scraping, spam generation, or circumvention of rate-limiting mechanisms. We discuss these considerations alongside limitations in Appendix A.12.

10. **Safeguards**

    Answer: [N/A]

    Justification: The paper does not release new high-risk pre-trained models, image generators, or scraped datasets. The base models used (Gemma3-12B, Llama3.1-8B) were publicly released by their respective providers with their own safeguards. No new dataset beyond synthetic subgoal annotations (generated over the public WebArena-Lite benchmark) is released.

11. **Licenses for existing assets**

    Answer: [Yes]

    Justification: WebArena [Zhou et al., 2023] and WebArena-Lite [Liu et al., 2024] are cited and used under their respective open-source licenses. Base models Gemma3-12B and Llama3.1-8B are used under their published license terms. Baselines DigiRL [Bai et al., 2024], WebRL [Qi et al., 2024], and AWR [Peng et al., 2019] are cited and reproduced from their original implementations. Gemini-2.5-pro and other proprietary models are accessed through their official APIs in accordance with published terms of service.

12. **New assets**

    Answer: [N/A]

    Justification: The paper does not release new assets (models, datasets, or benchmarks) at submission. Subgoal annotations are generated from the existing benchmark via prompting; the full prompts used are documented in Appendix A.6 (Figure 15) to enable regeneration.

13. **Crowdsourcing and research with human subjects**

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. The validation of the failure classifier (Appendix A.2, Table 3) used examples labeled internally by the authors, not by external crowd workers.

14. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Answer: [N/A]

Justification: The paper does not involve research with human subjects, so no IRB approval is required.

15. **Declaration of LLM usage**

Answer: [Yes]

Justification: LLMs are a central component of the research methodology rather than a writing aid. Gemini-2.5-pro is used as the teacher model for subgoal generation (Section 4.1, Figure 15) and as the AutoRater/SubGoal Checker during trace annotation. Gemini-2.5-flash is used for the automated failure analyzer (Appendix A.2) and for semantic similarity resampling in the curriculum generation loop (Appendix A.4). An Outcome Reward Model trained via WebRL [Qi et al., 2024] serves as the final task-level judge. Full prompts for all LLM roles are provided in Appendix A.6.