FailAlign: Failure-Aligned Bootstrapping for Long-Horizon HTML Based Code Generation Agents

Anonymous ACL submission

Abstract

LLM-based web agents automate browser interactions by generating structured action sequences, yet long-horizon execution remains vulnerable to error propagation under noisy DOM observations. Existing training pipelines are commonly supervised by reference demonstrations and provide limited direct supervision on near-miss failures under the same state. We propose FailAlign, a failure-aligned learning framework that integrates error signals into training. The framework mines a first-error index from rollouts and uses it in a three-stage pipeline: action-grounded context compression, hard-negative DPO for local disambiguation, and trajectory-level GRPO for cascade mitigation. On Mind2Web with Qwen3-32B, FailAlign reaches 49.7% SSR and 5.6% SR, surpassing much larger models and strong commercial systems under the same evaluation protocol. We position these results as evidence of improved robustness under the replayable Mind2Web reference setting, rather than as direct evidence for live stochastic web environments.

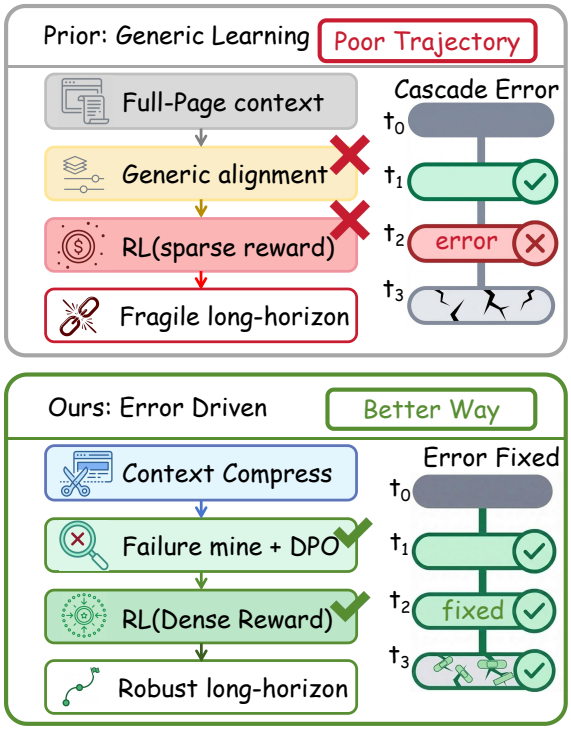


Figure 1: FailAlign helps LLMs fix error and make trajectory more robust.

1 Introduction

LLM-based web agents have emerged as a practical paradigm for autonomous browser automation, aiming to complete real-world tasks across diverse domains (Nakano et al., 2021; Yao et al., 2022; Deng et al., 2023). However, robust long-horizon execution remains difficult. Real web pages contain noisy DOM structures with many distractors, and early missteps can propagate to downstream failures. In addition, many existing training pipelines primarily optimize against reference actions and provide limited explicit supervision on near-miss failures. As a result, agents can still confuse locally plausible options that lead to different long-horizon outcomes (Figure 1).

To address these challenges, we propose FailAlign, a failure-aligned learning framework that integrates error signals into supervision. The method contains three complementary components: action-grounded context compression that distills verbose HTML into compact DOM snippets, hard-negative preference learning with DPO (Rafailov et al., 2024) for local disambiguation, and trajectory-level GRPO (Shao et al., 2024) for mitigating cascaded failures. We study this problem in the replayable, reference-based Mind2Web setting, where decisions are evaluated step by step on state-conditional DOM contexts. Under this protocol, FailAlign-RL reaches 49.7% SSR and 5.6% SR on Mind2Web (Deng et al., 2023) with Qwen3-32B (QwenTeam, 2025), outperforming larger open models and strong closed-

source models under the same evaluation protocol. We interpret these results as improved robustness within the Mind2Web reference setting and do not directly claim transfer to stronger online benchmarks such as Mind2Web2 or WebArena.

The contribution of this work lies in how failure supervision is organized across the training pipeline. FailAlign mines a single first-error anchor from each failing rollout and reuses it as a common interface for data construction across later stages. On top of this anchor, the method combines action-grounded DOM compression to reduce input noise, hard-negative DPO to sharpen local discrimination among confusable actions, and trajectory-level GRPO to reduce error cascades over longer horizons. On the Task, Domain, and Website splits of Mind2Web, this design yields consistent gains in EAcc, F1, SSR, and SR over same-scale and larger baselines, showing that failure-aligned supervision improves robustness within this benchmark.

2 Method

2.1 Problem Formulation

We study long-horizon HTML-based action code generation under a *structured multiple-choice* formulation. Given an instruction x and the current webpage state s_t, we construct a step context c_t and an option set E_t that contains candidates at step t. The model outputs a structured choice $a_t = (o_t, e_t, v_t)$, where o_t denotes the operation type (a member of { select, type, click }), e_t indexes the target element selected from the candidate set E_t, and v_t is null iff operation is click. Executing a_t updates the environment to s_{t+1}, yielding a trajectory $\tau = (s_0, a_0, \ldots, a_{T-1}, s_T)$ where T denotes the termination timestep. This formulation lets the model reason over a bounded but semantically meaningful action space, reducing ambiguity from free-form action generation while keeping the interface aligned with the benchmark protocol. It further enables deterministic step-level evaluation under the Mind2Web reference protocol by comparing the predicted choice with the recorded reference choice, which in turn defines failure traces from interaction logs and supports failure-aligned training in later sections.

2.2 Failure-Aware Supervised Warm-Up

Action-Grounded Context Compression. As shown in Figure 2, we convert each raw webpage state s_t into a compact, action-grounded DOM snippet, yielding a compressed step context c_t. Given the candidate element set E_t, we prune the DOM by taking a small local neighborhood around each candidate and union them into c_t. We preserve visible text and salient attributes with value filtering and truncation, and relabel remaining nodes with consecutive ids for compact serialization. By centering serialization on candidate neighborhoods rather than the full page, this representation retains local grounding evidence while excluding distractor-heavy boilerplate that would otherwise dominate the prompt budget.

Warm-Up Instance Construction. Following the generation of the compressed step context, we construct structured multiple-choice instances characterized by a fixed set of five distinct options. The supervision signal for this stage is the reference structured action $a_t = (o_t, e_t, v_t)$ taken directly from the Mind2Web human demonstration, where $e_t \in E_t$ identifies the grounded target element from the provided candidates. We do not relabel or rewrite these benchmark references. In scenarios where the reference target is absent from the available options, we include a designated `None` choice to represent the "not in the list" condition. This construction process results in a supervised warm-up corpus $\mathcal{D}_{\text{warm}}$ containing approximately 77K instances. The primary objective of this dataset is to train the model to accurately align the relevant step context with the predefined option space while adhering to the required output format.

Supervised Warm-Up and Error Exposure. We train a base policy π_θ on $\mathcal{D}_{\text{warm}}$ and obtain FailAlign-SFT, which equips the model with basic competence for predicting structured actions under the stepwise multiple-choice format. We then roll out FailAlign-SFT on the training task pool and collect stepwise interaction logs, recording $(x, s_t, c_t, E_t, \hat{a}_t, s_{t+1})$ at step t, where s_t is the raw html, c_t is the compressed step context, and s_{t+1} is the state after executing $\hat{a}_t$.

Step-Level Error Mining Interface. On these logs, we apply a deterministic step evaluator and assign a binary label $r_t \in \{0, 1\}$. A step is correct iff the operation type matches the Mind2Web reference, the selected option corresponds to the reference target, and the value field matches when applicable. For each rollout trajectory $\hat{\tau}$, we identify the first failing step $t^\star(\hat{\tau}) = \min\{t \mid r_t = 0\}$ when such a step exists. We keep one failure an-

Model	Size	Task				Domain				WebSite			
		EAcc	**F1**	**SSR**	**SR**	**EAcc**	**F1**	**SSR**	**SR**	**EAcc**	**F1**	**SSR**	**SR**
Closed-Source LLMs													
GPT-4o	🔒	26.1	63.1	22.9	1.6	30.1	64.2	26.8	1.4	24.9	59.8	20.9	0.6
GPT-5.1-2025-11-13	🔒	38.3	72.8	33.5	2.0	38.3	70.8	33.4	2.7	36.9	68.5	**38.7**	**2.3**
Gemini-2.5-flash	🔒	39.3	70.8	34.4	**3.8**	38.0	70.2	33.5	2.9	36.9	67.5	30.0	2.2
Gemini-2.5-pro	🔒	39.3	70.8	34.4	3.6	40.1	69.3	35.7	**3.1**	39.1	**70.1**	31.2	2.2
Claude-haiku-4-5-20251001	🔒	41.4	74.6	33.1	2.0	39.3	**71.7**	33.8	2.4	35.5	69.1	29.2	1.1
Claude-sonnet-4-5-20250929	🔒	48.5	**75.2**	**40.6**	2.8	**45.3**	71.2	**46.7**	2.3	**43.6**	68.9	35.3	0.6
Claude-opus-4-5-20251101	🔒	**50.1**	66.2	39.4	3.1	44.6	63.4	37.3	1.8	43.5	64.2	36.5	1.9
Open-Source LLMs													
Qwen3-0.6B	0.6B	7.0	3.5	0.7	0.0	8.6	6.9	1.0	0.0	6.5	6.0	0.7	0.0
Qwen3-1.7B	1.7B	13.5	44.1	8.3	0.4	16.3	42.9	10.0	0.3	13.1	40.1	6.6	0.0
Qwen3-4B	4B	21.0	53.6	15.6	0.2	23.6	54.9	17.9	0.2	19.7	50.4	12.7	0.3
Seed-Coder-8B-Instruct	8B	13.8	65.8	12.4	1.2	14.4	65.0	13.1	0.4	11.3	62.3	9.2	0.0
Qwen3-8B	8B	24.5	63.9	20.2	1.2	25.4	65.2	21.6	1.9	21.6	60.5	16.6	0.6
Qwen3-14B	14B	28.0	60.7	21.6	1.7	29.0	51.5	23.3	1.5	23.7	57.7	17.2	0.6
Qwen3-30B-A3B-Instruct-2507	3/30B	24.0	67.0	21.1	1.6	27.2	66.9	23.5	1.4	22.7	64.1	17.8	0.0
Qwen3-30B-A3B-Thinking-2507	3/30B	27.3	65.5	24.6	1.3	24.1	61.6	20.9	1.0	26.1	62.7	21.1	1.1
Qwen3-Coder-30B-A3B-Instruct	3/30B	31.3	71.4	28.3	1.4	31.7	70.6	29.0	2.4	26.5	66.5	22.7	1.7
QwQ-32B	32B	30.0	70.0	26.0	1.6	31.4	69.2	27.5	2.1	28.2	62.8	22.4	0.6
Qwen2.5-Coder-32B-Instruct	32B	24.6	64.8	21.9	2.0	25.8	64.2	22.6	1.4	21.3	60.0	18.0	0.0
Qwen3-235B-A22B-Thinking-2507	22/235B	37.4	68.4	31.7	**2.8**	36.2	68.1	31.1	3.0	33.9	64.4	27.7	1.1
Qwen3-235B-A22B-Instruct-2507	22/235B	30.6	67.2	26.7	1.5	30.9	68.3	27.6	2.8	28.7	64.6	22.7	1.1
GLM-4.7	32/358B	**41.4**	64.7	34.5	2.1	38.5	62.7	32.7	2.3	**39.7**	58.7	30.1	1.8
MiniMax-M2.1	10/230B	32.1	69.4	27.5	1.2	31.7	66.9	27.3	2.5	28.0	63.3	21.7	0.9
Qwen3-Coder-480B-A35B-Instruct	35/480B	40.9	**74.6**	**37.0**	**2.8**	37.6	**72.1**	**34.4**	3.2	36.1	**70.0**	30.6	**2.4**
DeepSeek-V3.2	37/671B	30.3	67.4	26.1	2.0	32.4	67.4	27.8	2.2	30.2	62.7	22.7	1.7
Kimi-K2-Thinking	32B/1T	40.5	72.2	34.8	2.4	**38.9**	70.7	34.3	**3.7**	36.9	67.1	30.0	1.1
Kimi-K2-Instruct-0905	32B/1T	38.3	70.6	30.1	2.1	**38.9**	64.4	31.8	1.9	37.4	65.0	**31.6**	1.8
Qwen3-32B	32B	30.7	65.4	24.9	2.4	31.7	66.7	26.8	2.2	30.2	62.6	22.7	1.1
Qwen3-32B-Raw-HTML	32B	33.0	69.1	28.1	2.4	32.3	68.6	28.4	2.3	30.9	65.0	24.6	1.6
Our Method													
FailAlign-SFT	32B	50.1	77.8	45.8	4.8	43.6	70.9	39.2	3.0	41.9	68.7	37.0	2.3
FailAlign-DPO	32B	52.5	78.9	47.5	5.1	44.2	72.0	40.2	3.4	42.3	70.0	38.1	2.8
FailAlign-RL	32B	**54.6**	**79.4**	**49.7**	**5.6**	**46.8**	**74.6**	**42.1**	**3.8**	**44.5**	**72.7**	**39.6**	**3.5**

Table 1: Results on three test sets.

chor per trajectory and form a unified first-error index

$$\mathcal{I} = \{(t^\star, x, c_{t^\star}, E_{t^\star}, \hat{a}_{t^\star}, a^*_{t^\star})\}.$$

Trajectories with no failing step are excluded from $\mathcal{I}$. In other words, $\mathcal{I}$ contains exactly one first-error step for each failing trajectory, rather than all mismatched steps along the rollout. Reusing this same exact-match criterion in evaluation, failure mining, and later reward construction keeps the supervision target consistent across the entire pipeline.

2.3 Hard-Negative Disambiguation

Objective. Given the failure index $\mathcal{I}$ in Figure 2, we focus on highly confusable step contexts in which the model makes a near-miss decision: the predicted action is locally plausible under the same options, yet deviates from the gold choice and leads to drastically different outcomes. Our goal is to construct a compact preference dataset $\mathcal{D}_{\text{pref}}$ that sharpens the decision boundary over operation type, and value fields under the same context.

Hard-Negative Construction. For each failure instance $(t, x, c_t, E_t, \hat{a}_t, a^*_t) \in \mathcal{I}$, we generate preference pairs using the compressed context c_t that was originally constructed during the supervised warm-up phase via the action-grounded DOM compression pipeline described in Section 2.2. We first extract candidate negative actions from the stepwise interaction logs collected during FailAlign-SFT rollouts, requiring each candidate to share the same context (x, c_t, E_t) as the gold action a^*_t. We then apply operation-level validity constraints induced by the structured action schema and keep only candidates whose action form is compatible with the current option space. To ensure physical validity, we further require successful environment execution: an action is retained only if it produces a valid state transition $s_t \rightarrow s'_{t+1}$ with-

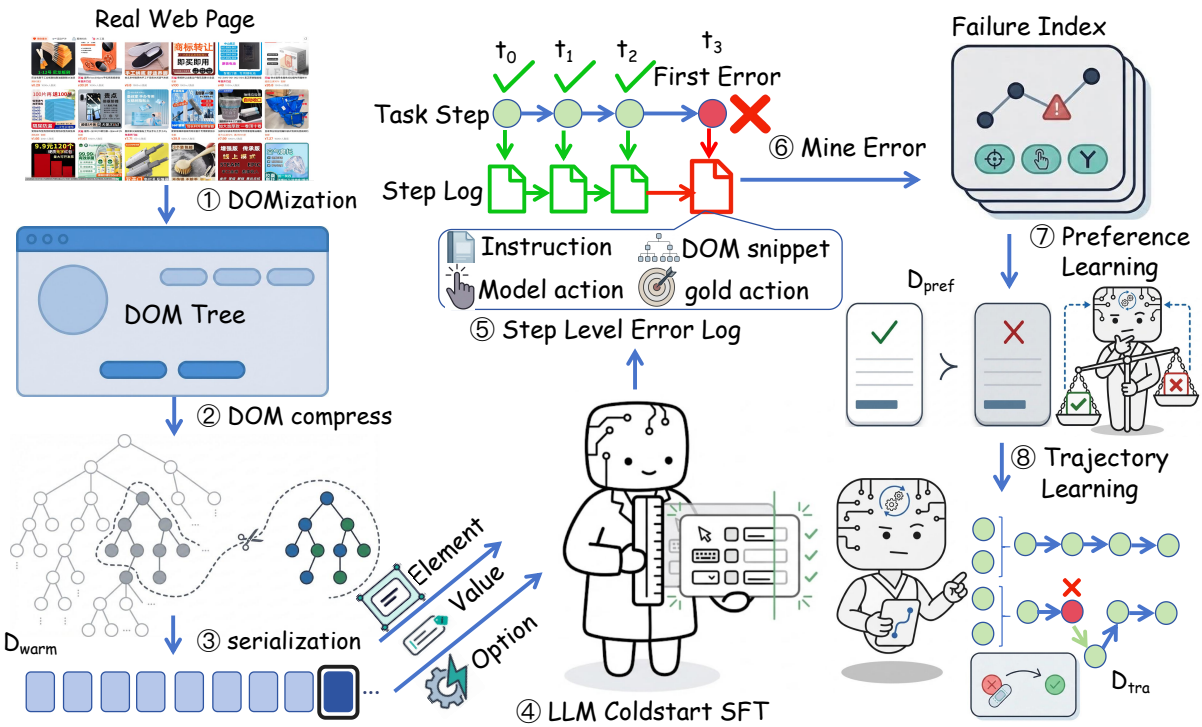


Figure 2: Overview of the FailAlign pipeline. The process begins with DOMization, compression, and serialization of raw web pages to construct structured inputs for LLM Coldstart SFT. During task execution, the system records granular step-level logs, which are subsequently mined to pinpoint the First Error and aggregate these instances into a unified Failure Index. This error-centric dataset drives two distinct optimization phases: Preference Learning, which contrasts specific correct and incorrect actions, and Trajectory Learning, which operates at the trajectory level to refine the model's holistic planning capabilities by learning from complete task execution paths.

out triggering runtime exceptions such as "element not found" errors. Within this executable pool, we serialize each structured action and encode it with Qwen3-Embedding-8B (Zhang et al., 2025); similarity ranking is used only at this final stage to select near-miss negatives a_t^- that are most confusable with the gold action a_t^*. Each validated pair (a_t^*, a_t^-) is then formatted into a preference comparison $(x, c_t, E_t, a_t^* \succ a_t^-)$. After applying operation constraints, executability filtering, and similarity ranking across all failure instances in $\mathcal{I}$, we obtain $\mathcal{D}_{\text{pref}}$, a refined dataset of approximately 43K high-signal comparisons that trains the policy to resolve ambiguities between actions that appear locally plausible in c_t but lead to different task outcomes.

Usage. We optimize the policy π_θ on the preference dataset $\mathcal{D}_{\text{pref}}$ using Direct Preference Optimization (DPO). This approach explicitly shapes the policy's likelihood distribution to satisfy $\pi_\theta(a_t^* \mid x, c_t, E_t) \succ \pi_\theta(a_t^- \mid x, c_t, E_t)$ for all preference pairs, where the identical step context (x, c_t, E_t) ensures the comparison isolates decision

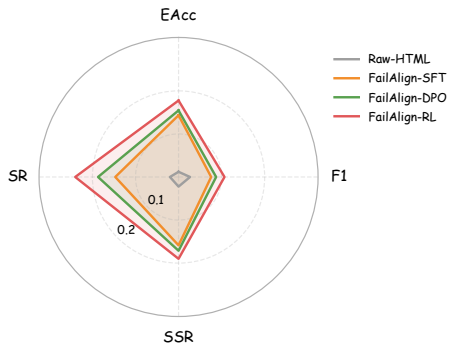


Figure 3: Normalized deltas over the Qwen3-32B base across four metrics, showing progressive gains from each training stage.

quality from contextual variations. The resulting policy exhibits refined discrimination capability between semantically similar actions that would otherwise lead to divergent task outcomes.

2.4 Mitigating Error Cascades in Long Trajectories

Long-Horizon Goal. Beyond local correction and hard-negative disambiguation, our final goal is

Dataset	# Instances	Avg. Tokens
$\mathcal{D}_{\text{warm}}$	77,750	442.3
$\mathcal{D}_{\text{pref}}$	43,205	1,997.1
$\mathcal{D}_{\text{tra}}$	1,009	15,375.7

Table 2: Overview of instance counts and average input length across training datasets.

to mitigate error cascades in which early near-miss decisions compound into downstream failures over long horizons. Given an instruction x and an initial state s_0, we aim to learn a policy π_θ that improves trajectory-level robustness while maintaining per-step correctness.

Trajectory-Level Signal Construction. Leveraging the first-error index $\mathcal{I}$ from Section 2.2, we construct trajectory-level supervision on trajectories that contain a mined first-error step. For each task, we form the rollout trajectory $\hat{\tau} = (s_0, \hat{a}_0, \ldots, s_T)$ as defined in Section 2.1. Let $t^\star$ be the indexed first-error step for this trajectory. We repair $\hat{\tau}$ by resetting the environment to the golden state $s^*_{t^\star}$ recorded during warm-up data collection, then appending the corresponding gold suffix.

$$\tau^\star = \underbrace{(s_0, \hat{a}_0, \ldots, \hat{a}_{t^\star-1})}_{\text{executed prefix}} \oplus \underbrace{(s^*_{t^\star}, a^*_{t^\star}, \ldots, s^*_T)}_{\text{gold suffix}} \quad (1)$$

where $\oplus$ denotes sequence concatenation. This reset is used only during offline data construction so that the reference suffix can execute in the intended context despite prefix errors; it is not assumed at test time. The resulting dataset $\mathcal{D}_{\text{tra}}$ comprises approximately 1,000 repaired trajectories, concentrating supervision on first failure correction while preserving trajectory structure from Section 2.1.

Usage. We use $\mathcal{D}_{\text{tra}}$ as the task pool for Group Relative Policy Optimization (GRPO). For a sampled trajectory $\tau = (a_0, \ldots, a_{T-1})$, we define the step reward

$$r_t = 2 \cdot \mathbb{I}[a_t = a^*_t] - 2 \cdot \mathbb{I}[a_t \neq a^*_t]. \quad (2)$$

where exact match includes the operation, target element, and the value field when applicable. We further add a terminal bonus $r_{\text{term}} = 3$ when the full task succeeds under the benchmark reference rule, and 0 otherwise. The trajectory reward is the undiscounted return

$$R(\tau) = \sum_{t=0}^{T-1} r_t + r_{\text{term}}. \quad (3)$$

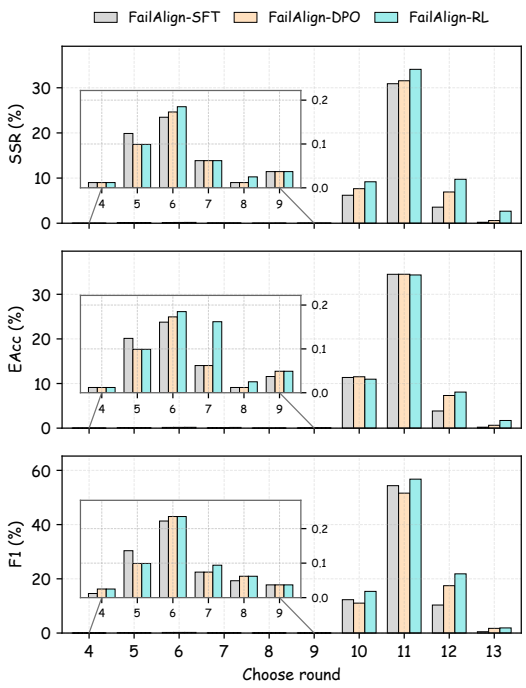


Figure 4: Round-wise SSR, EAcc, and F1 for SFT, DPO, and RL, with an inset highlighting early rounds.

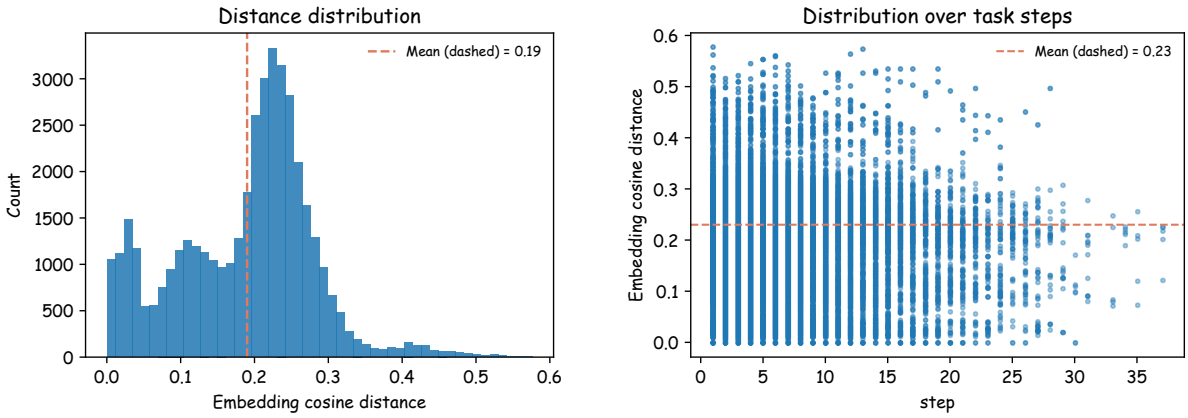
For each instruction x, GRPO samples a group of trajectory completions from π_θ and optimizes group-relative advantages under $R(\tau)$. This training encourages plans that remain reference-consistent over long horizons and mitigates error cascades without training a separate critic.

3 Experiment

3.1 Implementation Details

We utilize Qwen3-32B as the foundational model for all experiments. Our training pipeline sequentially progresses through supervised fine-tuning on the initial and failure-aligned corpora ($\mathcal{D}_{\text{warm}}$), followed by preference alignment via DPO ($\mathcal{D}_{\text{pref}}$) and trajectory-level optimization via GRPO ($\mathcal{D}_{\text{tra}}$). All models are trained on 8 NVIDIA A800 GPUs using LLaMA-Factory[1] and Verl[2]. For the supervised stages, we adopt a learning rate of 5×10^{-5} with a cosine decay scheduler and a global batch size of 256. In the subsequent alignment stages, we reduce the learning rate to 1×10^{-6} for DPO (with $\beta = 0.1$) and 5×10^{-7} for GRPO with a group size of 5 to ensure stable convergence. Unless otherwise stated, we keep the tokenizer, prompt structure, action format, and candidate presentation protocol fixed across stages so that performance

[1] https://github.com/hiyouga/LLaMA-Factory
[2] https://github.com/volcengine/verl



(a) Distribution of positive and negative examples in the $\mathcal{D}_{\text{pref}}$.

(b) Distribution of candidate difficulty and steps in the $\mathcal{D}_{\text{tra}}$.

Figure 5: Difficulty distributions for the DPO and RL stages.

differences are attributable to training signals rather than interface changes.

3.2 Evaluation Metrics

Element Accuracy (EAcc). Element Accuracy indicates the correctness of the chosen element: $\text{EAcc} = \frac{1}{N}\sum_{t=1}^{N} \mathbb{I}\big[\hat{e}_t = e_t^*\big]$.

Operation F1. Operation F1 evaluates operation prediction at the token level. We compute precision P and recall R between predicted and gold operation tokens: $\text{F1} = \frac{2PR}{P+R}$.

Step Success Rate (SSR). SSR counts a step as successful only when both the target element is right and the operation prediction is fully correct: $\text{SSR} = \frac{1}{N}\sum_{t=1}^{N} \mathbb{I}\big[(\hat{o}_t = o_t^*) \wedge (\hat{e}_t = e_t^*)\big]$.

Task Success Rate (SR). SR measures the proportion of tasks completed with all steps successful: $\text{SR} = \frac{1}{M}\sum_{i=1}^{M} \mathbb{I}\big[\forall t \in \tau_i,\ \text{step}_t \text{ is successful}\big]$. SR is a stricter, long-horizon metric and is typically much lower than SSR.

3.3 Evaluation Benchmark

Mind2Web. The Mind2Web dataset (Deng et al., 2023) is a replayable benchmark built from real website traces. It comprises 2,350 open-ended tasks collected from 137 websites spanning 31 domains, including travel, shopping, and finance. Unlike synthetic environments, Mind2Web requires agents to interpret raw HTML and perform step-by-step interactions to complete complex instructions. The test set is stratified into Cross-Task, Cross-Website, and Cross-Domain splits, which lets us measure transfer across held-out tasks and websites under the same reference protocol.

3.4 Top-K Rolling Evaluation

At each step, let C_t denote the candidate elements returned by the Mind2Web candidate-generation pipeline (osnlpteam, 2023). We reuse the benchmark-provided relevance score between the task instruction and each candidate element, sort C_t by this score, and retain the top-K candidates with $K = 50$ by default. The retained list is partitioned into groups of 10 candidates, yielding $\lceil K/10 \rceil = 5$ rounds in the default configuration. In each round, the model selects one preferred option from the current group. From round 2 onward, we append the preferred option from the previous round to the current group for re-comparison. We stop after all groups are processed and use the final preferred option as the predicted element. This rolling procedure only changes how a fixed candidate pool is presented under context limits; it does not alter candidate identities or the final exact-match scoring rule. Unless otherwise noted, all reported baselines use the same candidate pool, ranking, grouping, and rolling protocol. The purpose of this design is to preserve protocol parity under context limits rather than to introduce extra search power for any particular model family.

3.5 Evaluation Baselines

Closed-Source Models. The evaluation includes frontier proprietary models from OpenAI (OpenAI, 2025a,b), Google (Anil et al.), and Anthropic (Anthropic, 2025). These systems provide strong externally available comparators under the same evaluation protocol.

Open-Source Models. We include strong open-source reasoning and code models spanning a wide parameter range, together with a same-backbone

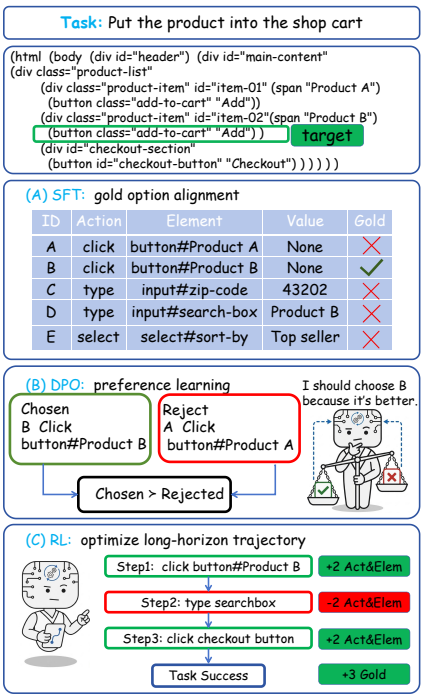


Figure 6: Data formats for the three objectives: SFT multiple-choice supervision, DPO chosen-rejected comparisons, and RL trajectory rewards.

Qwen3-32B baseline and a Raw-HTML variant. This design helps separate gains from model scale, base model choice, and input representation.

3.6 Main Results

Table 1 shows the results on three test sets. Overall, FailAlign delivers consistent gains across Task, Domain, and Website, with steady improvements from SFT to DPO to RL, indicating cumulative benefits across stages. Compared to the same-scale baseline, FailAlign-RL improves both SSR and SR, suggesting better local decision quality and stronger long-horizon stability in this protocol. FailAlign also outperforms the Raw-HTML baseline, indicating that failure-aligned supervision and structured context compression provide complementary gains over directly training on raw page inputs. The improvements remain visible on the more challenging Domain and Website splits, suggesting that the method is not merely memorizing task-specific action patterns from the warm-up stage. At the same time, absolute SR remains low for all models, so we interpret these gains as improved robustness within the benchmark rather than a solved long-horizon web-control problem.

4 Analysis

Ablation Study. Figure 3 illustrates how each stage of FailAlign contributes to overall performance. We utilize the Raw-HTML configuration as a baseline, which reflects the capabilities of a model trained on standard uncompressed HTML sequences without explicit error-correction mechanisms. The specialized variants consistently improve over this baseline across all four evaluation metrics. We observe that FailAlign-RL yields the largest gains, particularly in SR and EAcc, suggesting that recovery-oriented trajectory training is especially helpful after local disambiguation has been improved by earlier stages. The steady outward expansion of the performance frontier from FailAlign-SFT through FailAlign-DPO to FailAlign-RL indicates that error-driven refinement provides cumulative benefits throughout the pipeline.

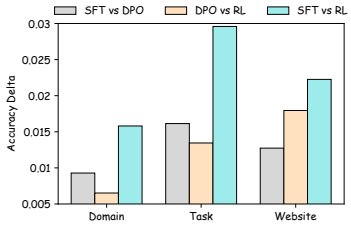


Figure 7: Accuracy deltas from SFT to DPO and RL across Domain, Task, and Website splits.

Round-wise Analysis. As illustrated in Figure 4, continuous alignment training yields consistent performance gains across all metrics. Both FailAlign-DPO and FailAlign-RL improve over FailAlign-SFT, validating the efficacy of FailAlign. Notably, FailAlign-RL shows a clear advantage in scenarios requiring multi-turn reasoning, which is consistent with its trajectory-level objective. Under the Mind2Web protocol, this stage improves long-horizon stability beyond what can be obtained from step-level supervision alone.

Dataset Composition and Difficulty. As presented in Table 2, $\mathcal{D}_{\text{warm}}$ comprises the largest number of instances but the fewest input tokens, serving as a warm-up stage for basic action grounding. In contrast, $\mathcal{D}_{\text{pref}}$ and $\mathcal{D}_{\text{tra}}$ are derived mainly from error mining. We further evaluate the complexity of these candidates using embedding cosine distance in Figure 5. The distance between DPO positive and negative pairs concentrates around 0.19, indicating near-miss comparisons that are challenging enough to support robust preference learning. The RL candidates center near 0.23 and involve tasks with approximately 11 steps, providing informative long-horizon training signals for the later stage.

Each Stage Training Signal. Figure 6 provides a unified walkthrough of our data formats under three objectives. Given a natural-language task and a bracket-serialized DOM snapshot, the dataset defines a structured action space. For SFT, supervision is a k-way option table that annotates a single reference action aligned with the target element, while keeping the remaining plausible distractors as alternative choices. For DPO, supervision is expressed as a chosen versus rejected pair constructed from two competing options, explicitly encoding preference learning over closely related actions. For RL, signal comes from trajectory rollouts composed of multiple steps, where exact step matches receive positive reward, mismatches receive negative reward, and successful completion receives a terminal bonus, enabling optimization beyond single-step correctness.

Error Analysis The analytical results presented in Figure 7 illustrate the progressive rectification of prediction errors throughout the FailAlign training pipeline. We observe consistent positive accuracy deltas across all primary evaluation splits, including Domain, Task, and Website, indicating that the subsequent DPO and RL stages successfully recover failure cases that were mispredicted during the initial SFT phase. The incremental improvements from SFT to DPO and from DPO to RL demonstrate continuous policy refinement as the model receives more structured feedback. The substantial cumulative gain from the initial SFT baseline to the final RL stage supports the view that iterative failure-aligned optimization improves reliability on the Mind2Web splits studied here.

5 Related Work

Web-agent perception and context construction. Recent work has shown that observation design is central to web-agent performance. AgentOccam (Yang et al., 2025) and LCoW (Lee et al., 2025) improve action quality by reducing irrelevant context and preserving task-relevant structure, while Mind2Web-style systems also rely on candidate generation pipelines such as MindAct (osnlpteam, 2023). Our action-grounded compression follows this line, but it is used as a supporting supervision interface rather than the main novelty.

Learning from interaction failures. Web-agent training has moved beyond pure imitation toward reinforcement and self-generated trajectories, including DigiRL (Bai et al., 2024), WebRL (Qi et al., 2025), and WebAgent-R1 (Wei et al., 2025). Unlike methods that target online exploration or open-ended recovery, FailAlign mines a first-error anchor from replayable rollouts and uses it to derive both step-level and trajectory-level supervision under a fixed reference protocol.

Preference learning and critical-step supervision. Preference optimization methods such as DPO (Rafailov et al., 2024) sharpen local decision boundaries, but web actions are only meaningfully comparable under the same instruction, DOM context, and candidate set. FailAlign constructs state-conditional near-miss comparisons from the same step context and focuses supervision on the first erroneous decision in each failing rollout, reducing suffix noise.

Benchmark scope and the offline-to-online gap. Mind2Web (Deng et al., 2023) provides a controlled replayable protocol, whereas WebArena (Zhou et al., 2024), WebCanvas (Pan et al., 2024), BrowserGym (Chezelles et al., 2025), and Mind2Web2 (Gou et al., 2025) place more emphasis on online interaction and stochasticity. Recent analyses further show that gains under offline references may overstate online robustness (Xue et al., 2025). We therefore scope our empirical claims to the Mind2Web reference protocol.

6 Conclusion

We present FailAlign, a failure-aligned learning framework for long-horizon HTML-based action code generation that couples action-grounded context compression, step-level first-error mining, hard-negative DPO, and trajectory-level GRPO within a single pipeline. On Mind2Web, FailAlign improves element grounding and structured action correctness, yielding gains in EAcc, Operation F1, SSR, and SR across Task, Domain, and Website splits. We view these results as evidence that failure-aligned supervision improves robustness under the Mind2Web reference protocol.

7 Limitations

We evaluate FailAlign only on Mind2Web, a replayable reference benchmark, and therefore do not extrapolate directly to live settings such as Mind2Web2 or WebArena (Gou et al., 2025; Zhou

et al., 2024). Our first-error index and GRPO rewards rely on exact references, while Stage 3 uses recorded golden-state resets during offline construction and both training and evaluation inherit the benchmark candidate-generation interface. These design choices improve protocol control but leave broader online generalization to future work.

Ethics Statement

All data used in this work is derived from publicly available, open-source resources, and we follow the original dataset terms and licenses when constructing training signals and reporting results. We do not collect new private user information, and our processing focuses on task-relevant DOM/visible context; any released artifacts should be used in compliance with website terms and applicable policies. As with other web automation and code-generation systems, the approach could be misused for unintended automation; we encourage responsible deployment with appropriate access controls, logging, and safeguards. In particular, any real-world use should avoid automated interactions with authenticated, rate-limited, or policy-restricted websites without explicit authorization.

References

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. Gemini: A family of highly capable multimodal models. corr, abs/2312.11805, 2023. doi: 10.48550. *arXiv preprint ARXIV.2312.11805*, pages 24–28.

Anthropic. 2025. Introducing claude 4.5 sonnet. https://www.anthropic.com/news/claude-sonnet-4-5. Accessed: 2025-09-30.

Hao Bai and 1 others. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. 2025. The browsergym ecosystem for web agent research. *Preprint*, arXiv:2412.05467.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Preprint*, arXiv:2306.06070.

Boyu Gou, Zanming Huang, Yuting Ning, Yu Gu, Michael Lin, Weijian Qi, Andrei Kopanev, Botao Yu, Bernal Jiménez Gutiérrez, Yiheng Shu, Chan Hee Song, Jiaman Wu, Shijie Chen, Hanane Nour Moussa, Tianshu Zhang, Jian Xie, Yifei Li, Tianci Xue, Zeyi Liao, and 7 others. 2025. Mind2web 2: Evaluating agentic search with agent-as-a-judge. *Preprint*, arXiv:2506.21506.

Dongjun Lee and 1 others. 2025. Learning to contextualize web pages for enhanced decision making by llm agents. In *International Conference on Learning Representations*.

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

OpenAI. 2025a. Introducing chatgpt-5.1. https://openai.com/index/gpt-5-1/. Accessed: 2025-11-12.

OpenAI. 2025b. Introducing chatgpt-5.2. https://openai.com/index/introducing-gpt-5-2/. Accessed: 2025-12-11.

osnlpteam. 2023. Mindact.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. 2024. Webcanvas: Benchmarking web agents in online environments. *Preprint*, arXiv:2406.12373.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *Preprint*, arXiv:2411.02337.

QwenTeam. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li.

2025. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *Preprint*, arXiv:2505.16421.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. An illusion of progress? assessing the current state of web agents.

Ke Yang and 1 others. 2025. Agentoccam: A simple yet strong baseline for llm-based web agents. In *International Conference on Learning Representations*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. *Preprint*, arXiv:2307.13854.

A Example of Evaluation Prompt

Evaluation Prompt

```
(html (body (div id=0 (div button my store open store locator (div (small Open until 8:00pm ) (span San Diego ) ) ) (div
     button delivery zip code 43202 open (div (small Deliver to ) (span 43202 ) ) ) ) (div (div (button id=1 show
    sorting options modal (span Sort ) ) (button show filter modal color ) (button filter products by in stock ) (
    button show filter modal material ) (button filter products by top seller ) (button filter products by last chance
    ) (button filter products by new products ) (button id=2 show more filters modal (span All filters ) ) ) (div (div
    id=3 (a frnuft - 20-piece flatware set (img frnuft 20-piece flatware set, stainless ) ) (div (div (div (span FRNUFT
     ) (span 20-piece flatware set ) ) (span (span $ ) (span 12 ) (span (span . ) 99 ) ) (span review: 4.7 out of 5 (
    span (407) ) ) ) (div (button button add to bag, frnuft, 20-piece ) (button button save to igift, frnuft, 20-piece
    ) (button button save to favorites, frnuft, 20-piece ) ) ) ) (div (input checkbox select for product comparison, 20-
    piece checkboxname on ) (span id=4 Available for delivery ) (span In stock at San Diego, CA ) ) ) ) ) )
```

Based on the HTML webpage above, try to complete the following task:

Task: Find the cheapest set of stainless steel spoons, which are available in stock in San Diego, California.

Previous actions:
- `[span] Columbus -> CLICK`
- `[searchbox] Search by ZIP code or city, state -> TYPE: san diego`
- `[button] Submit -> CLICK`
- `[span] Select store -> CLICK`
- `[link] Flatware -> CLICK`

What should be the next action? Please select from the following choices (If the correct action is not in the page above, please select A. `None of the above`):

A. None of the above

B. `(div id=0 (div button my store open store locator (div`

C. `(button id=1 show sorting options modal (span Sort ) )`

D. `(button id=2 show more filters modal (span All filters )`

E. `(div id=3 (a frnuft - 20-piece flatware set (img frnuft)`

F. `(span id=4 Available for delivery )`

Answer Example:
Answer: C.
Action: SELECT
Value: Pickup

B DOM Representation and Context Compression Details

B.1 Raw HTML

```
<!-- Raw HTML (excerpt; noisy & redundant).
     Many nodes are irrelevant to action
    selection. -->
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8"/>
  <title>Flatware | My Store</title>
  <script src="/static/vendor/react.bundle.
    js"></script>
  <script>window.__ANALYTICS__ = {
    sessionId:"...", ab:"exp_42" };/*
    dozens of tracking beacons, feature
    flags, personalization, etc. */</script>

  <style>/* large CSS bundle, icons,
    responsive rules, animations ... */</
    style>
</head>
<body>
  <div id="cookie-banner" class="overlay">
    <p>We use cookies for performance and
    personalization...</p>
    <button aria-label="Accept all">Accept</
    button>
    <button aria-label="Manage preferences">
    Manage</button>
  </div>
  <header class="site-header">
    <nav>
      <a href="/">Home</a><a href="/shop">
    Shop</a><a href="/deals">Deals</a>
      <form role="search"><input
    placeholder="Search products"/></form>
      <svg><!-- icons --></svg>
    </nav>
  </header>
  <main id="content">
    <!-- Store context bar (relevant
    context but surrounded by wrappers) -->
    <section class="store-bar">
      <!-- (kept) root candidate for choice
     B after compression -->
      <div class="store-block">
        <div class="left">
          <div class="spacer"></div>
          <div class="store-entry">
            <button aria-label="my store"
    data-testid="store-locator">
              my store
              <span class="icon"></span>
            </button>
            <div class="meta">
              <small>Open until 8:00pm</
    small>
              <span>San Diego</span>
            </div>
          </div>
        </div>
        <div class="right">
          <button aria-label="delivery zip
    code" data-testid="delivery-zip">
            delivery zip code 43202
          </button>
          <div class="meta">
            <small>Deliver to</small><span>
    43202</span>
          </div>
        </div>
      </div>
    </section>
    <!-- Toolbar (many sibling buttons;
    only a few become candidate roots) -->
    <section class="toolbar">
      <div class="toolbar-inner">
        <!-- (kept) choice C after
    compression -->
        <button id="sortBtn" aria-label="
    show sorting options modal">
          <span>Sort</span><span class="
    icon"></span>
        </button>
        <!-- mostly irrelevant filters for
    this step -->
        <button aria-label="show filter
    modal color">color</button>
        <button aria-label="filter products
     by in stock">in stock</button>
        <button aria-label="show filter
    modal material">material</button>
        <button aria-label="filter products
     by top seller">top seller</button>
        <button aria-label="filter products
     by last chance">last chance</button>
        <button aria-label="filter products
     by new products">new</button>
        <!-- (kept) choice D after
    compression -->
        <button id="allFiltersBtn" aria-
    label="show more filters modal">
          <span>All filters</span>
        </button>
      </div>
    </section>
    <!-- Product grid (deep nesting,
    repeated wrappers, ads, rec widgets,
    etc.) -->
    <section class="grid">
      <div class="grid-row">
        <div class="sponsored-slot">
          <div class="ad">ADVERTISEMENT</
    div>
          <script>/* ad network injection */
    </script>
        </div>
        <!-- (kept) choice E after
    compression -->
        <div class="product-tile" data-sku=
    "FRNUFT-20">
          <a href="/p/frnuft-20-piece-
    flatware-set" class="product-link">
            frnuft - 20-piece flatware set
            <img alt="frnuft 20-piece
    flatware set, stainless"
                 src="/img/frnuft.jpg"/>
          </a>
          <div class="details">
            <div class="names"><span>FRNUFT
    </span><span>20-piece flatware set</
    span></div>
            <div class="price"><span>$</
    span><span>12</span><span>.99</span></
    div>
            <div class="reviews">review:
    4.7 out of 5 <span>(407)</span></div>
```

```
            <div class="cta">
              <button aria-label="add to
    bag">add to bag</button>
              <button aria-label="save to
    igift">save to igift</button>
              <button aria-label="save to
    favorites">save to favorites</button>
            </div>
          </div>
          <div class="availability">
            <input type="checkbox" aria-
    label="select for product comparison"/>
            <!-- (kept) choice F after
    compression -->
            <span class="delivery">
    Available for delivery</span>
            <span class="store-stock">In
    stock at San Diego, CA</span>
          </div>
        </div>
        <aside class="recommendations">
          <h3>Recommended for you</h3>
          <div class="carousel">...</div>
        </aside>
      </div>
    </section>
  </main>
  <footer>
    <div class="legal">Terms  Privacy
    Accessibility  ...</div>
    <script>/* more telemetry & hydration
    scripts */</script>
  </footer>
</body>
</html>
```

B.2 Compressed DOM

```
(html
  (body
    (div id=0
      (div button my store open store
    locator
        (div (small Open until 8:00pm) (
    span San Diego)))
      (div button delivery zip code 43202
    open
        (div (small Deliver to) (span 43202)
    )))

    (div
      (div
        (button id=1 show sorting options
    modal (span Sort))
        (button show filter modal color)
        (button filter products by in stock)

        (button show filter modal material)
        (button filter products by top
    seller)
        (button filter products by last
    chance)
        (button filter products by new
    products)
        (button id=2 show more filters
    modal (span All filters))))

    (div
      (div id=3
        (a frnuft - 20-piece flatware set
          (img frnuft 20-piece flatware
    set, stainless))
        (div
          (div
            (div (span FRNUFT) (span 20-
    piece flatware set))
            (span (span $) (span 12) (span
    (span .) 99))
            (span review: 4.7 out of 5 (
    span (407))))
          (div
            (button button add to bag,
    frnuft, 20-piece)
            (button button save to igift,
    frnuft, 20-piece)
            (button button save to
    favorites, frnuft, 20-piece))))
      (div
        (input checkbox select for product
    comparison, 20-piece checkboxname on)
        (span id=4 Available for delivery)
        (span In stock at San Diego, CA)))))


# Candidate roots (A--F) used by the policy
     for action selection:
# A: None of the above
# B: subtree rooted at (div id=0 ...)
# C: subtree rooted at (button id=1 ...)
# D: subtree rooted at (button id=2 ...)
# E: subtree rooted at (div id=3 ...)
# F: subtree rooted at (span id=4 ...)
```

Raw HTML vs. compressed DOM. A straightforward way to represent webpage state is to serialize the raw page HTML (Raw HTML), which maximizes fidelity but introduces substantial boilerplate and weakly action-related tokens (e.g., layout wrappers, UI chrome, and auxiliary modules). To understand whether providing HTML context is beneficial in principle, we include a Raw-HTML variant trained with the same supervision format. As shown in Table 1, adding raw HTML context already improves over the base model (Qwen3-32B), indicating that webpage HTML does contain useful grounding cues for action prediction. However, this gain comes with a clear efficiency drawback: the raw serialization is unnecessarily long and noisy, making the prompt budget dominated by irrelevant content and consequently limiting the model's ability to focus on discriminative evidence.

In contrast, our compressed DOM representation preserves the actionable structure and local evidence while aggressively removing non-essential tokens, and it explicitly aligns candidate subtrees to the decision options. This design improves the information-to-noise ratio under a fixed context budget, which translates into consistently stronger performance across splits and metrics in Table 1. In particular, our FailAlign-based training with com-

pressed inputs (SFT/DPO/RL) yields further gains beyond the Raw-HTML-trained variant, supporting the claim that *how* we represent HTML context matters: naively providing more HTML is helpful but inefficient, whereas targeted compression retains the useful grounding signal and enables more effective learning.

B.3 First-Error Mining and Rolling Evaluation Protocol

First-error mining. For each rollout trajectory, we evaluate every step with the same exact-match rule used in the main evaluation: operation type and target element id must both match the Mind2Web reference. If the trajectory contains at least one mismatch, we keep only the earliest mismatch as the first-error anchor and add it to the failure index. This index is then reused for DPO pair construction and repaired-trajectory construction.

Top-K rolling evaluation. At each step, we rank all candidates by relevance score, keep the top-K candidates ($K = 50$ by default), and split them into groups of 10 candidates. The model performs one choice per group. From the second group onward, the preferred option from the previous group is appended to the current group for re-comparison. After all groups are processed, the last preferred option is used as the final element prediction. This rolling process changes only presentation order and keeps the underlying candidate identities unchanged.