WEASEL: OUT-OF-DOMAIN GENERALIZATION FOR WEB AGENTS VIA IMPORTANCE-DIVERSITY DATA SELECTION

Fatemeh Pesaran zadeh[1*] **Seyeon Choi**[1*] **Xing Han Lù**[2,3] **Siva Reddy**[2,3,4] **Gunhee Kim**[1†]

[1]Seoul National University [2]McGill University [3]Mila – Quebec AI Institute [4]Canada CIFAR AI Chair

fatemehpesaran@vision.snu.ac.kr

ABSTRACT

Large language models (LLMs) have enabled web agents that follow natural language goals through multi-step browser interactions. However, agents fine-tuned on specific trajectories and domain often struggle to generalize out of domain, and offline training can be compute-inefficient due to noisy, redundant trajectories and long accessibility-tree (AXTree) states. To address both issues, we propose WEASEL, a trajectory selection method for offline training of web agents. WEASEL selects a fixed-budget subset of trajectory steps by optimizing an objective that balances unary importance with pairwise diversity over states, websites, and interaction patterns, solving efficiently with a greedy algorithm. We further improve efficiency with action-centered AXTree pruning that keeps only content around the ground-truth action target, and we mitigate style mismatch for reasoning-native models by replacing expert traces with model-generated, style-consistent rationales. Across AgentTrek and NNetNav training datasets, evaluations in WebArena, WorkArena, and MiniWob, and experiments with Qwen2.5-7B, Gemma3-4B, and Qwen3-8B, WEASEL improves out-of-domain performance while reducing training cost, producing roughly 9.7-12.5$\times$ training speedups over standard fine-tuning.

1 INTRODUCTION

Large Language Models (LLMs) have moved beyond text generation to support multi-step reasoning, planning, and action in interactive environments (Wang et al., 2023; Liu et al., 2023). This progress has accelerated interest in LLM-based web agents that map natural-language goals to sequences of browsering (Qi et al., 2024; Zhou et al., 2024a). Recent improvements are fueled by strong foundation models and large-scale instruction data (Ou et al., 2024; Murty et al., 2025; Xu et al., 2025b), along with agent-specific training such as imitation learning and reinforcement learning (Qi et al., 2025; Wei et al., 2025), enabling agents to solve increasingly complex, long-horizon web tasks.

Despite this progress, recent web agents remain limited in two key respects. First, agents fine-tuned on specific trajectories and environments often fail to generalize to out-of-domain tasks and settings. Performance often drops substantially when deployed on the websites, layouts, or interaction patterns that differ from those seen during training (Murty et al., 2025). Recent studies mostly evaluate under in-benchmark settings (Qi et al., 2024; Lai et al., 2024; Zhou et al., 2024b; Shen et al., 2025; Wei et al., 2025), training on a subset of tasks from a benchmark environment and testing on held-out tasks within the same environment (e.g., WebArena-to-WebArena in Figure 1). This limits their applicability in real-world Web tasks where distribution shifts are inevitable.

Second, offline web interaction data are often noisy and redundant (Nekoei et al., 2025; Xu et al., 2025a); expert trajectories can be overly long while containing relatively sparse task-relevant signal, due to redundant steps, drifting behaviors, or partially misaligned actions. Thus, models are prone to overfitting to narrow patterns rather than learning robust, transferable behaviors. This motivates improving both learning efficiency and generalization ability by curating an informative and diverse subset of trajectories and pruning distracting page content.

*Equal contribution. †Corresponding author.

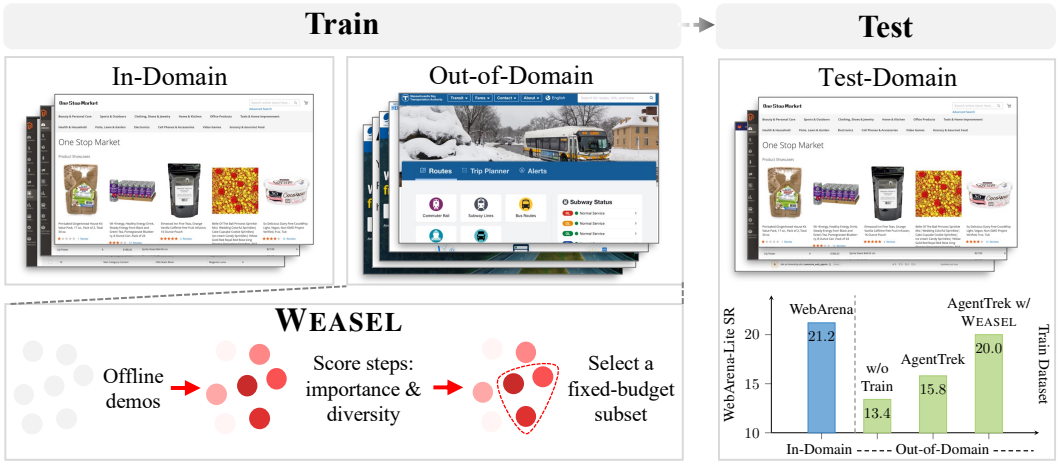


Figure 1: Overview of WEASEL. Conventional trained web agents show a sharp performance drop under out-of-domain shifts to unseen websites and interaction patterns. WEASEL tackles this challenge via novel trajectory selection: it scores offline demonstration steps for goal relevance and diversity, then applies greedy subset selection under a fixed budget. Agents trained with WEASEL generalize better to unseen test domains, achieving a +4.2 gain in zero-shot transfer from AgentTrek (Xu et al., 2024)→WebArena-Lite (Qi et al., 2024) with Qwen3-8B (Team, 2025). Note that bar-top labels indicate the training data source.

To address these two challenges, we propose WEASEL (**WE**b **A**gents with trajectory **SEL**ection), a data selection approach for offline web agent training that improves both learning efficiency and out-of-domain generalization. WEASEL formulates a fixed-budget subset selection problem with a quadratic objective that balances unary importance and pairwise diversity (§2.2). The unary importance prioritizes the steps that are semantically relevant to the goal, while the pairwise diversity encourages coverage over heterogeneous states, websites, and interaction patterns. Since the resulting subset selection problem is NP-hard in general, we employ an efficient greedy algorithm that scales to large collections of long trajectory in practice (§2.3). WEASEL improves (1) out-of-domain generalization by explicitly encouraging diversity, which mitigates overfitting to website-specific artifacts (Figure 1), and (2) training efficiency by removing redundant and noisy steps from long trajectories, achieving higher accuracy with up to 12.5× training speed-up compared to training on the full data (Table 1).

To further improve training efficiency, we introduce target-centered pruning. In web agent settings, each state can contain a large accessibility tree (AXTree), where action-relevant cues may be diluted by substantial irrelevant page content (Deng et al., 2023; Kerboua et al., 2025a). Thus, we prune each state by retaining AXTree content localized around the ground-truth action target, compactly preserving the information most relevant for predicting the expert action (§2.4). This pruning is orthogonally applicable and effectively reduces both the input length and training cost.

For reasoning-native models such as Qwen3 (Yang et al., 2025), we find that fine-tuning with reasoning traces written by a different model can be detrimental. The issue arises when the reasoning style in the training data differs from that used in the pretraining of the the model; as a result, it can harm both out-of-domain generalization and overall performance. To address this, we let the model self-generate style-consistent reasoning traces while preserving the original action supervision in the trajectories (§2.5). We find this step crucial for improving out-of-domain generalization in reasoning-native models (Table 4).

In summary, our contributions are as follows.

- To the best of our knowledge, we propose WEASEL, the first data selection approach for offline web agent training that is explicitly designed to achieve two goals: (1) improved out-of-domain generalization and (2) more compute-efficient learning. WEASEL formulates a subset selection objective that balances unary importance and pairwise diversity. We develop a greedy algorithm that scales to large trajectory collections. We also introduce

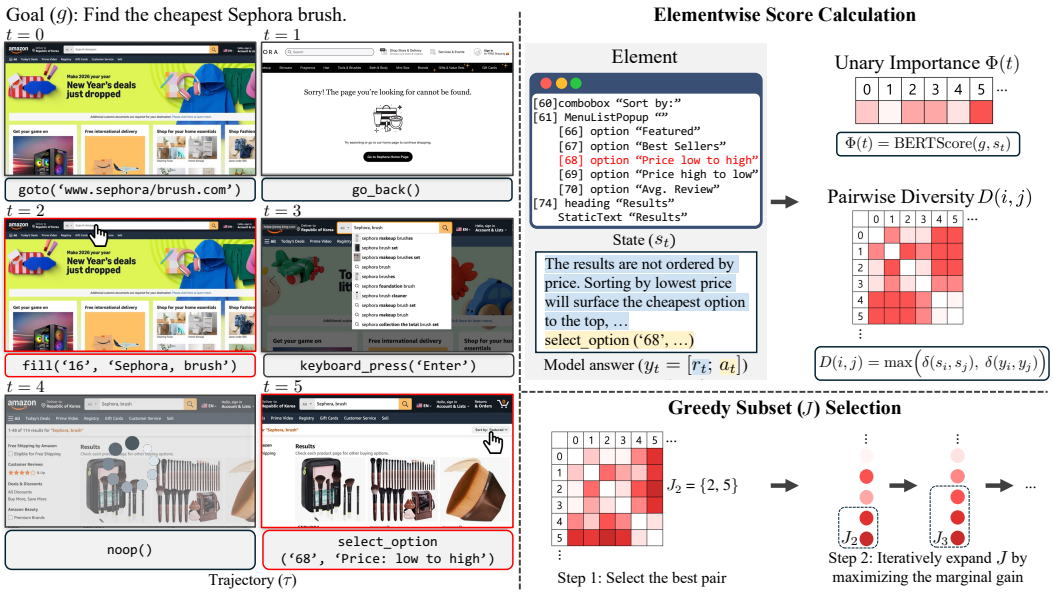


Figure 2: (**Left**): An example of a curated trajectory after applying WEASEL. Although the original collected data contain noisy steps ($t = 4$), and erroneous actions ($t = 0$), WEASEL selects a compact subset that retains only the most informative steps (in red) for the goal. (**Right**): Overview of WEASEL. We first perform element-wise score calculation using unary importance and pairwise diversity. WEASEL then applies a greedy subset selection procedure to derive an informative subset J. The algorithm initializes J by selecting the best-scoring pair that maximizes the objective, and then iteratively adds the element k that yields the largest marginal gain, until the budget is met (Algorithm 1).

action-centered AXTree pruning to remove redundant context and accelerate training. For reasoning-native models, we synthesize style-consistent reasoning traces to mitigate style mismatch during fine-tuning.

- We perform a diverse evaluation; web agents trained from two offline datasets, AgentTrek (Xu et al., 2024) and NNetNav (Murty et al., 2025), are evaluated across multiple benchmarks (e.g., WebArena (Zhou et al., 2024a), WorkArena (Drouin et al., 2024a), and MiniWob (Shi et al., 2017; Liu et al., 2018)) in a zero-shot manner, using multiple model families (e.g., Qwen2.5-7B (Qwen et al., 2025), Gemma3-4B (Team et al., 2025) and Qwen3-8B (Team, 2025)). We provide ablations that isolate the effects of subset selection, state pruning, and reasoning synthesis. Beyond substantial out-of-domain generalization gains (see Table 1), WEASEL improves training efficiency over standard SFT baselines, achieving 9.7-12.5× training speed-ups depending on the model.

2 WEASEL

WEASEL is a data selection method for training of offline web agents from long, noisy demonstrations. Given an expert trajectory dataset, WEASEL constructs a compact supervision set by selecting a fixed-budget subset of informative yet diverse steps. To further reduce training time, we prune each Web state to retain only action-relevant AXTree content (§2.4). For reasoning-native models, self-reasoning synthesis mitigates reasoning-style mismatch when the training traces are produced by a different model (§2.5).

2.1 PROBLEM FORMULATION

Setup. Let $\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}$ denote an offline dataset of expert web trajectories. Each trajectory $\tau \in \mathcal{D}$ is a sequence $\tau = \left(g, \{(s_t, h_t, a_t, r_t)\}_{t=0}^{T-1}\right)$, where g is a natural-language goal, s_t is a Web state, h_t is an interaction history up to step t, $a_t \in \mathcal{A}$ is an expert action (e.g., `click`, `type`,

`scroll`, `goto`), and r_t is a reasoning trace. We learn a policy π_θ that imitates the expert in an offline manner by predicting both the action and the reasoning trace at each step, conditioning on the goal, interaction history, and Web state, i.e., $\pi_\theta(a_t, r_t \mid g, h_t, s_t)$. We denote the model answer by $y_t = [r_t;\, a_t]$, formed by concatenating the reasoning trace and action.

States and actions for web agents. We represent each state s_t as a linearization of the page accessibility tree (AXTree). We write the linearized node sequence as $V_t = [v_{t,1}, v_{t,2}, \ldots, v_{t,K_t}]$, where each node $v_{t,k}$ has a unique identifier (e.g., a `bid`), and $K_t = |V_t|$ is the number of nodes in the linearization. For the action space $\mathcal{A}$, we follow BrowserGym (de Chezelles et al., 2025) to include node-grounded actions that reference an AXTree element via its identifier (e.g., `click`, `type`) as well as non-node actions that do not require an identifier (e.g., `goto`, `noop`).

Goal. Since raw web trajectories include many redundant steps and noisy, overly long demonstrations, WEASEL constructs a compact supervision set by selecting a fixed-budget subset of steps $J \subseteq \{0, \ldots, T-1\}$ with $|J| = T_0 \ll T$. This yields the curated dataset $\tilde{\mathcal{D}} = \bigcup_{\tau \in \mathcal{D}} \{(g, h_t, s_t, a_t, r_t)\}_{t \in J^*(\tau)}$, where $J^*(\tau)$ is obtained by our goal-conditioned importance-diversity objective (§2.2) and solved using a greedy algorithm (§2.3).

2.2 GOAL-CONDITIONED SUBSET SELECTION

Trajectories in the training set for Web agents are often unnecessarily long. In AgentTrek (Xu et al., 2024), a single trajectory contains up to 45 state-action pairs, although only a small fraction is directly relevant to the task goal. Training on long, redundant trajectories is also prone to overfitting to environment-specific patterns. We therefore cast trajectory curation as a subset selection problem that retains only the most informative steps while maintaining diversity among the retained steps to improve generalization.

Algorithm 1 WEASEL subset selection (greedy)

input Goal g, trajectory $\{(s_t, y_t)\}_{t=0}^{T-1}$, budget T_0, weight λ
1: Compute $\Phi(t) = \mathrm{BERTScore}(g, s_t)$ for all t
2: Compute (or cache on-demand) $D(i, j)$ via Eq. 3
3: $(i_1, i_2) \leftarrow \arg\max_{i<j}\ \Phi(i) + \Phi(j) + \lambda D(i, j)$
4: $J \leftarrow \{i_1, i_2\}$
5: **for** $m = 3$ to T_0 **do**
6: $\quad i_m \leftarrow \arg\max_{k \notin J}\ \Phi(k) + \lambda \sum_{i \in J} D(k, i)$
7: $\quad J \leftarrow J \cup \{i_m\}$
8: **end for**
output $J^* \leftarrow \mathrm{sort}(J)$

Given a trajectory of length T, we select a compact subset of $T_0 \ll T$ steps; let $J \subset \{0, \ldots, T-1\}$ denote the selected indices with $|J| = T_0$. We maximize the objective that combines unary importance and pairwise diversity:

$$\underset{J \subseteq \{0,\ldots,T-1\}}{\text{maximize}} \quad \sum_{j \in J} \Phi(j) \;+\; \lambda \sum_{\substack{i<j \\ i,j \in J}} D(i, j) \quad \text{s.t.} \quad |J| = T_0, \tag{1}$$

where $\Phi(j)$ is an *importance* score and $D(i, j)$ is a *diversity* score, and λ controls the tradeoff.

For importance $\Phi(j)$, we score each step s_t by its semantic relevance to the goal:

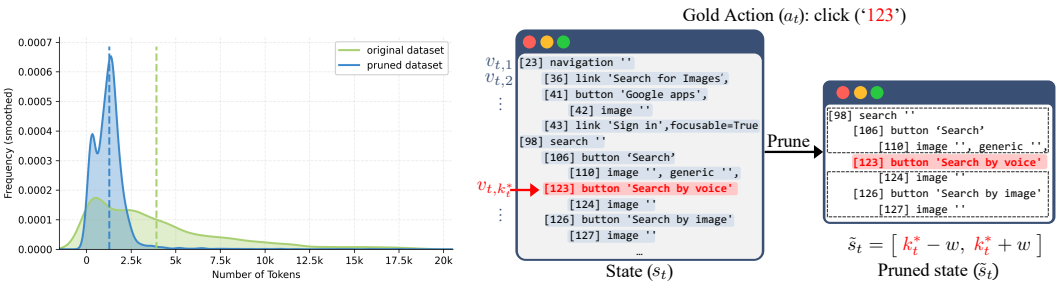
$$\Phi(t) = \mathrm{BERTScore}(g, s_t), \tag{2}$$

where we use the BERTScore (Zhang et al., 2020) to quantify how well the content of step t aligns with the goal g. In principle, BERTScore can be replaced by any embedding similarity; we study alternative similarity models in Section C. However, maximizing importance alone can select redundant, highly similar states and may encourage overfitting to website-specific patterns. We therefore incorporate a pairwise diversity term $D(i, j)$ defined as the maximum pseudo-distance between either the states or the associated model answers:

$$D(i, j) = \max\Big(\delta(s_i, s_j),\ \delta(y_i, y_j)\Big), \tag{3}$$

where $y_i = [r_i;\, a_i]$ denotes the model answer for step i formed by concatenating its reasoning trace r_i and action a_i, and the pseudo-distance $\delta(x, y)$ between two text sequences x and y is defined as

$$\delta(x, y) \triangleq 1 - \mathrm{BERTScore}(x, y), \tag{4}$$



(a) Token distribution of 10K subsamples of AgentTrek (Xu et al., 2024) before pruning (green) and after gold-target-centered pruning (blue). Pruning substantially reduces long-tail states, making the resulting sequences more manageable for training.

(b) An illustration of **Target-centered Pruning**. Given a state s_t in the form of AXTree and gold action a_t, we retain only the AXTree elements within a fixed window of size w centered at the target index k_t^*, producing the pruned state $\tilde{s}_t$. The k-th node in the linearized AXTree at step t is denoted $v_{t,k}$ (e.g., $v_{t,1}$, $v_{t,2}$), and v_{t,k_t^*} is the gold target node.

Figure 3: Token distribution before/after pruning (left) and an illustration of target-centered pruning (right).

so that $\delta(x, y)$ is small when x and y are semantically similar. Together, these terms yield the subset selection objective in Eq. 1 under the constraint in set size. Unless otherwise stated, we set $\lambda = 1$ in all experiments.

2.3 The Greedy Algorithm

While the problem in Eq. 1 is NP-hard (Garey & Johnson, 2002), it can be formulated as a variant of the max-sum diversification problem, which maximizes a modular quality term plus a sum of pairwise distances under a cardinality constraint (Borodin et al., 2017). For metric distances, a greedy algorithm achieves a constant-factor guarantee, including a 2-approximation (Borodin et al., 2017, Theorem 4.1). In our problem, $D(i, j)$ is defined using semantic pseudo-distances (e.g., $1 - \mathrm{BERTScore}$ and a max-composition) that do not satisfy metric properties, so this guarantee does not formally apply. Nevertheless, greedy selection remains an efficient and effective heuristic in practice.

We first initialize the selected set by choosing the best pair:

$$(i_1, i_2) = \arg \max_{0 \le i < j < T} \Big(\Phi(i) + \Phi(j) + \lambda D(i, j) \Big), \qquad J_2 = \{i_1, i_2\}. \tag{5}$$

Starting from this seed, for $m = 3, 4, \ldots, T_0$, we expand the set by adding the index that yields the largest marginal gain with respect to the objective:

$$\Delta(k \mid J_{m-1}) := \Phi(k) + \lambda \sum_{i \in J_{m-1}} D(k, i),$$
$$i_m = \arg \max_{k \notin J_{m-1}} \Delta(k \mid J_{m-1}), \tag{6}$$
$$J_m = J_{m-1} \cup \{i_m\}.$$

We output $J^\star = J_{T_0}$ as the selected subset of indices. The full procedure is summarized in Algorithm 1.

In terms of complexity, computing all unary scores $\Phi(t)$ takes $O(T)$. In addition, precomputing all pairwise distances $D(i, j)$ costs $O(T^2)$ (or they can be cached on demand). Given cached distances, each greedy iteration evaluates marginal gains for up to T candidates, leading to an overall cost of $O(T_0 T)$ for the selection stage.

2.4 Target-centered Pruning

Web states can be prohibitively long (Deng et al., 2023; Lù et al., 2024; Kerboua et al., 2025a) because the accessibility tree (AXTree) includes highly repeated and peripheral content (e.g., headers,

sidebars, duplicated menus). In AgentTrek, a single linearized state can reach 180K tokens (Xu et al., 2024), far exceeding the context limits of most training and inference setups (Figure 3a). This length is not only costly but also noisy; only a small subset of nodes informs the expert's next action. To address this, we propose target-centered pruning, which preserves a compact neighborhood around the target element of the expert action, since local context is shown to be more informative for learning node-grounded web behaviors (Liu et al., 2025).

Specifically, at step t, let $V_t = [v_{t,1}, \ldots, v_{t,K_t}]$ be the linearized AXTree node sequence. For node-grounded actions, let k_t^* be the gold target index such that $v_{t,k_t^*} = u_t^*$, where $u_t^* \in V_t$ denotes the corresponding *target node*. We use (V_t, k_t^*) to define action-centered pruning. Given a fixed window size $w \in \mathbb{N}$, we keep a contiguous window of length $(2w{+}1)$ centered at k_t^*. Let

$$\mathcal{K}_t = \left[\, k_t^* - w,\ k_t^* + w \,\right] \cap \{1, \ldots, K_t\}. \tag{7}$$

We retain the subsequence $\tilde{V}_t = [\, v_{t,k} : k \in \mathcal{K}_t \,]$ and denote the pruned (linearized) state as $\tilde{s}_t$ (Figure 3b). For non-node actions (e.g., `goto`, `noop`), we instead keep a fixed-length prefix under the same length budget (e.g., $2w{+}1$ or a fixed L) to obtain $\tilde{V}_t$. We compare our pruning method against prior state-pruning baselines in Table 3, and further analyze the importance of retaining target-local context in Figure 4. This target-centered pruning is applied as a preprocessing step, prior to running WEASEL.

2.5 SELF-REASONING SYNTHESIS FOR REASONING MODELS

Reasoning-native models are fine-tuned to emit intermediate reasoning traces before producing final outputs (DeepSeek-AI, 2025; Team, 2025). When fine-tuning such models on expert traces r_t produced by a different model or annotation process, we observe that naive fine-tuning can introduce a style mismatch. In practice, this mismatch destabilizes training and may degrade performance, sometimes even falling below the base model before fine-tuning.

To mitigate this, we replace expert reasoning traces with *self-synthesized* traces that match the base model's reasoning style (Zelikman et al., 2022), while keeping the action supervision unchanged. For each selected step $t \in J^*$, we prompt the target reasoning model to generate a trace $\hat{r}_t$ conditioned on the goal, history, pruned state, and the expert action:

$$\hat{r}_t \sim q(\, r \mid g, h_t, \tilde{s}_t, a_t \,), \tag{8}$$

where q is implemented by prompting the base model to produce a plausible rationale consistent with a_t. Refer to Section D.2 for full details of the prompt. We then fine-tune the policy to predict both the synthesized trace and the expert action:

$$\max_{\theta} \sum_{\tau \in \mathcal{D}} \sum_{t \in J^\star(\tau)} \log \pi_\theta\big(a_t, \hat{r}_t \mid g, h_t, \tilde{s}_t\big). \tag{9}$$

We ablate the effect of self-reasoning synthesis in Table 4.

3 EXPERIMENTS

3.1 SETUP

Baselines. We compare WEASEL against standard supervised fine-tuning (SFT) baselines under an offline training using AgentTrek (Xu et al., 2024) and NNetNav (Murty et al., 2025) datasets. For LLMs of agents, we use the pretrained Qwen2.5-7B-Instruct (Qwen et al., 2025), Gemma3-4b-IT (Team et al., 2025) and Qwen3-8B (Yang et al., 2025). For all models and datasets, we report performance obtained by reproducing the training process using the hyperparameters specified in Section A. We test baselines that are fine-tuned with different configurations of training sets: (i) fine-tune on the full AgentTrek training set of 52K datapoints with multiple epoch budgets (e.g., 2/4 epochs, depending on the model), (ii) on a uniformly sampled subset of 10K datapoints using the same epoch count as WEASEL, (iii) on a sampled subset via LLM-as-Judge (Zheng et al., 2023), whose details can be found §D.1.

Table 1: Success rates (SR) under a zero-shot transfer setting. We fine-tune Qwen2.5-7B-Instruct (Qwen et al., 2025), Gemma3-4b-IT (Team et al., 2025), and Qwen3-8B (Team, 2025) on AgentTrek (Xu et al., 2024), and evaluate on different benchmarks, WebArena-Lite (Liu et al., 2024), WebArena (Zhou et al., 2024a), MiniWob (Shi et al., 2017; Liu et al., 2018), and WorkArena (Drouin et al., 2024b), with no training.

Model	WebArena-lite SR	WebArena SR	MiniWob SR	WorkArena L1 SR	WorkArena L2 SR	Training # Data	Training # Epochs	Training Time	Training Speed-up
Qwen2.5-7B-Instruct	6.7	5.2	41.8	4.8	0.0	–	–	–	–
+ Full (52K steps)	8.5	8.7	44.6	12.1	0.4	52K	4	136.0 hr	1.0×
+ Pruning (52K steps)	8.5	8.6	47.7	**12.4**	0.4	52K	4	62.0 hr	2.2×
+ Pruning + Sampling (10K steps)	8.5	8.1	46.7	9.8	3.0	10K	4	12.0 hr	11.3×
+ Pruning + LLM-Judge (10K steps)	5.5	7.8	45.4	8.5	3.0	10K	4	12.0 hr	11.3×
+ WEASEL (10K steps)	**13.3**	**9.5**	**48.0**	**12.4**	**4.7**	10K	4	12.0 hr	11.3×
Gemma3-4B-IT	5.5	2.7	27.4	3.6	0.0	–	–	–	–
+ Full (52K steps)	6.7	4.3	28.6	3.3	0.0	52K	2	80.0 hr	1.0×
+ Pruning (52K steps)	6.1	4.8	28.2	2.7	0.0	52K	2	30.0 hr	2.7×
+ Pruning + Sampling (10K steps)	6.7	4.7	29.4	2.4	2.1	10K	2	6.4 hr	12.5×
+ Pruning + LLM-Judge (10K steps)	6.7	5.2	29.3	**4.5**	2.1	10K	2	6.4 hr	12.5×
+ WEASEL (10K steps)	**7.9**	**5.5**	**30.6**	**4.5**	**3.0**	10K	2	6.4 hr	12.5×
Qwen3-8B	13.4	18.0	61.1	35.2	1.7	–	–	–	–
+ Full (52K steps)	16.4	18.2	59.4	33.3	2.1	52K	2	88.5 hr	1.0×
+ Pruning (52K steps)	15.2	12.7	40.3	15.5	2.6	52K	2	44.0 hr	2.0×
+ Pruning + Sampling (10K steps)	15.8	17.5	61.4	33.9	3.4	10K	2	7.0 hr	12.6×
+ Pruning + LLM-Judge (10K steps)	18.2	16.6	**61.9**	35.2	3.8	10K	2	7.0 hr	12.6×
+ WEASEL (10K steps)	**20.0**	**19.2**	**61.9**	**38.8**	**4.3**	10K	2	8.3 hr	10.7×

Table 2: Success rates (SR) under a different zero-shot transfer setting where NNetNav-Live dataset (Murty et al., 2025) is used for training instead of AgentTrek in Table 1.

Model	WebArena-Lite SR	WebArena SR	MiniWob SR	WorkArena L1 SR	WorkArena L2 SR	Training Speed-up
Qwen2.5-7B-Instruct	6.7	5.2	**41.8**	4.8	0.0	–
+ Full (52K steps)	10.3	6.9	38.9	5.2	6.4	1.0×
+ Pruning (52K steps)	6.7	3.2	38.4	1.8	1.3	1.7×
+ Pruning + Sampling (10K steps)	12.1	7.4	32.3	7.0	6.0	9.7×
+ WEASEL (10K steps)	**14.0**	**8.3**	**41.8**	**7.6**	**6.8**	9.7×

Evaluation of Out-of-Domain Generalization. While agents are trained on AgentTrek or NNetNav, we measure their success rates (SR) on WebArena-Lite (Liu et al., 2024), WebArena (Zhou et al., 2024a), MiniWob (Shi et al., 2017; Liu et al., 2018), and WorkArena (Drouin et al., 2024a). For consistent benchmark execution, we follow the default environment configuration and evaluation settings provided by the AgentLab framework (de Chezelles et al., 2025; Drouin et al., 2024a). We report aggregate SR over the full evaluation set for each benchmark. For WorkArena, we additionally report SR on Level-1 (L1) and Level-2 (L2) tasks to show the performance on simpler and more compositional workflows.

3.2 RESULTS

Table 1 reports zero-shot transfer success rates; the agents are trained on AgentTrek and evaluated on WebArena-Lite, WebArena, MiniWob, and WorkArena (L1/L2) with no additional fine-tuning. WEASEL delivers the strongest accuracy-efficiency trade-off; it achieves the best SR across benchmarks while reducing training cost by an order of magnitude compared to full-data SFT (about 10.7-12.5× speed-up, depending on the model). Notably, WEASEL yields strong transfer performance on the harder benchmarks, e.g., reaching 19.2 SR on WebArena and 4.3 (L2) on WorkArena with Qwen3-8B, and 13.3 SR on WebArena-Lite with Qwen2.5-7B-Instruct. Under the same 10K-step training budget, WEASEL also outperforms standard baselines, including uniform subsampling and LLM-as-judge selection. We note that for Qwen3-8B, the observed training speed-up is slightly lower than that of the other 10K baselines. This is due to the reasoning synthesis in §2.5, which uses self-generated rationales that are longer on average than the original AgentTrek traces, increasing the token budget, but yielding performance that outperforms full fine-tuning.

Table 2 reports the results obtained by replacing the AgentTrek training set with NNetNav-Live (Murty et al., 2025), which is constructed from real-world browser interactions. We fine-tune Qwen2.5-7B-Instruct for 4 epochs under the identical training setting and compare against full-data SFT and

random subsampling. As in Table 1, WEASEL outperforms all baselines across all benchmarks. Moreover, training on 10K datapoints yields a 9.7× speed-up relative to the full 52K setting. This shows WEASEL's robustness with the changed training set.

Pruning. Table 3 compares pruning strategies under a fixed pruning ratio on a 10K random subset of AgentTrek. We report WebArena-Lite success rate (SR) and training speed-up. We fix the token budget across methods (32% of the original tokens). We test three pruning baselines: (1) *Prune-by-Bid*, which keeps a prefix up to the gold bid for each web state; (2) *Semantic* Tan et al. (2025), which ranks nodes by semantic relevance to the gold action using BERTScore; and (3) *Target-centered + Semantic*, an ablation that forms the pruned state by taking the union of nodes selected by Target-centered (ours) and Semantic. Since these methods rely on node-grounded actions, we apply the same fixed-length prefix truncation to non-node actions (i.e., the first L nodes). Our Target-centered method in §2.4 achieves the best SR, outperforming all baselines. Notably, all pruning variants yield the same 2× training speed-up relative to the unpruned data (i.e., cutting training time by 50%). Please refer to §E for more details on the experiments, and §B for additional experiments on the effect of target-centered pruning.

Table 3: Comparison of state-truncation strategies under the same budget (about 32% of the original tokens). We fine-tune Qwen2.5-7B-Instruct on sampled 10K datapoints of AgentTrek, and report success rates (SR) on WebArena-Lite. Training speed-up is relative to the original (untruncated) dataset.

Method	WebArena-Lite	Speed-up
Original Data	10.3	1.0×
Prune-by-Bid	5.5	2.0×
Semantic Tan et al. (2025)	7.3	2.0×
Target-centered + Semantic	6.7	2.0×
Target-centered (Ours)	**10.3**	2.0×

Reasoning Synthesis. Table 4 reports an ablation on Qwen3-8B on WebArena-Lite, isolating the effects of data selection and reasoning synthesis (RS) in §2.5. For a fair comparison, all variants are fine-tuned under the same training setting with 2 epochs on a 10K subset of AgentTrek. Our experiments show that combining data selection with RS. WEASEL achieves the best result, reaching 20.0% SR, indicating that they are complementary.

Table 4: Ablation on isolating the effects of data selection and reasoning synthesis (RS). Qwen3-8B are fine-tuned with 10K training datapoints on WebArena-Lite for 2 epochs.

Method	Data Selection	Reasoning Synthesis	WebArena -Lite
Qwen3-8B	–	–	13.5
SFT (Random)	✗	✗	15.8
SFT (Random + RS)	✗	✓	17.6
WEASEL (Ours)	✓	✓	**20.0**

3.3 ANALYSES

Importance Score. Table 5 ablates the unary importance score Φ on WebArena-Lite (Liu et al., 2024; Zhou et al., 2024a) by varying the text paired with the goal when computing Φ. For a fair comparison, all variants fine-tune Qwen2.5-7B-Instruct for 4 epochs under the same training setting. Using the goal-state pairing (ours) yields the best transfer performance, 13.3 SR, outperforming goal-reasoning, and goal-state summary, where state summaries are generated with Flan-T5-XL (Chung et al., 2022).

Table 5: Ablation of the unary importance score Φ on WebArena-Lite. We compare different text paired with the goal when computing Φ, including the state, the reasoning trace, and a summary of the state.

Importance term	WebArena-Lite
Qwen2.5-7B-Instruct	6.7
Goal-State summary	11.5
Goal-Reasoning	10.3
Goal-State (Ours)	**13.3**

Diversity Score. In Table 7, we ablate the pairwise diversity term by isolating state diversity, model answer diversity, and their max-composition. All results are obtaiend by fine-tuning Qwen2.5-7B-Instruct for 4 epochs, changing only the definition of $D(i, j)$. State-only diversity yields 7.3 SR on WebArena-Lite, model answer-only improves to 12.1, and the max-composition performs the best at 13.3. This indicates the two signals are complementary; state diversity promotes coverage over UI and interaction contexts, while model answer diversity

Table 6: Ablation of the data selection objective on WebArena-Lite. We isolate the unary importance term (Φ) and the pairwise diversity term (D) in our subset selection objective.

Method	Unary Φ	Pairwise D	WebArena-Lite
Qwen2.5-Instrcut-7B	–	–	6.7
SFT (+ Unary)	✓	✗	10.3
SFT (+ Diversity)	✗	✓	7.9
WEASEL (Ours)	✓	✓	**13.3**

Table 7: Ablation of the pairwise diversity term $D(i, j)$ on WebArena-Lite. We compare using diversity computed from (1) states only ($\delta(s_i, s_j)$), (2) model answer only ($\delta(y_i, y_j)$), and (3) the proposed distance (Eq. 3)

Diversity term	WebArena-Lite
Qwen2.5-7B-Instrcut	6.7
State-only	7.3
Reasoning-only	12.1
WEASEL (Ours)	**13.3**

captures diverse intents and rationales; taking the maximum preserves diversity in either space, reducing redundancy and improving generalization.

Effect of Unary and Binary term. Table 6 isolates the unary importance term Φ and the pairwise diversity term D in our datas selection objective. All variants fine-tune Qwen2.5-Instruct-7B for 4 epochs under identical settings, with the only change of objective terms. Selecting data using the unary term alone improves performance to 10.3, while using the diversity term alone yields a smaller gain of 7.9. Combining both terms in WEASEL achieves the best result at 13.3 SR, indicating that importance and diversity provide complementary signals for constructing an effective training subset.

4 Related Work

Training Data and Environment for Web Agents. Early progress on LLM-based web agents has been driven by training with online interaction and reinforcement learning. WebRL (Qi et al., 2024; 2025), TTI (Shen et al., 2025), WebAgent-R1 (Wei et al., 2025), and AutoWebGLM (Lai et al., 2024) improve agent performance through interactive training and environment rollouts. However, these approaches are often developed and evaluated in in-benchmark settings, which may not fully reflect performance under distribution shift. More recently, large-scale foundation datasets have enabled scaling supervision for web agent training. AgentTrek (Xu et al., 2025b) synthesizes large-scale web agent trajectories from tutorial resources, while NNetNav (Murty et al., 2025) collects broad supervision from real-world browsing and retroactive labeling. Despite this progress, agents trained on such data can still exhibit substantial performance drops in out-of-domain test settings when evaluated on websites and interaction patterns that differ from those seen during training (Murty et al., 2025). Our work complements these efforts by treating out-of-domain transfer and training efficiency as first-class goals of *data selection*, selecting a fixed-budget subset that is both goal-relevant and diverse to reduce redundancy and improve robustness under distribution shift.

Data Selection and State Pruning for Web Agents. Selecting an informative subset of training data is a classic problem in machine learning, with coreset and data subset selection methods studied for efficiency and robustness (Sener & Savarese, 2018; Mirzasoleiman et al., 2020; Killamsetty et al., 2021). Related ideas have also been explored for preference optimization of LLMs, where careful curation can reduce training cost while improving alignment (Deng et al., 2025). Similar trends appear in instruction tuning and pretraining, where curated subsets or optimized data mixtures can match or improve performance at lower cost (Xie et al., 2023; Bukharin et al., 2024). To our knowledge, we propose WEASEL, the first principled subset selection method tailored to Web agent training, which improves out-of-domain generalization and training efficiency by jointly prioritizing goal-relevant steps and promoting diversity among selected steps. Several lines of work reduce web observation states primarily for test-time efficiency or context limitations, for example via learned contextualization/summarization modules or LLM-based retrieval over AXTree lines (Lee et al., 2025; Kerboua et al., 2025a;b). Our target-centered pruning instead targets training efficiency and trims states without introducing any additional modules or parameters.

5 CONCLUSION

We presented WEASEL, a data selection framework for compute-efficient offline web agent training that explicitly targets out-of-domain transfer. WEASEL selects a fixed-budget subset of trajectory steps by optimizing a simple objective that balances goal-conditioned importance and pairwise diversity, solved efficiently with a greedy algorithm. In addition, we introduced action-centered AXTree pruning to further reduce redundant input context, and self-reasoning synthesis to better fine-tune reasoning-native models. Across multiple web agent benchmarks and model families, WEASEL consistently improves zero-shot transfer performance under domain shift while substantially reducing training cost.

ETHICS STATEMENT

Our work makes offline training of web agents more practical by reducing the amount of redundant supervision needed to achieve strong transfer to unseen websites and interaction patterns. At the same time, improving web agent capability can increase the risk of misuse (e.g., automating spam, scraping, or unauthorized actions on websites). We encourage responsible release practices, including clear usage policies, rate limits, and safety evaluations on harmful or policy violating tasks, and we recommend deploying agents with safeguards such as action constraints and human oversight for high stakes.

REPRODUCIBILITY STATEMENT

To support reproducibility, we report all training hyperparameters, model settings, and evaluation protocols used in our experiments. We will make our implementation and configurations available upon publication and provide scripts to reproduce data selection and fine-tuning under the same experimental setup.

REFERENCES

Allan Borodin, Aadhar Jain, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions, and dynamic updates. *ACM Transactions on Algorithms (TALG)*, 13(3): 1–25, 2017.

Alexander Bukharin, Shiyang Li, Zhengyang Wang, Jingfeng Yang, Bing Yin, Xian Li, Chao Zhang, Tuo Zhao, and Haoming Jiang. Data diversity matters for robust instruction tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3411–3425, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.195.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, 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, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. Expert Certification.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

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

Xun Deng, Han Zhong, Rui Ai, Fuli Feng, Zheng Wang, and Xiangnan He. Less is more: Improving llm alignment via preference data selection, 2025.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024a.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Lo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024b.

Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

Imene Kerboua, Sahar Omidi Shayegan, Megh Thakkar, Xing Han L, Lo Boisvert, Massimo Caccia, Jrmy Espinas, Alexandre Aussem, Vronique Eglin, and Alexandre Lacoste. Focusagent: Simple yet effective ways of trimming the large context of web agents, 2025a.

Imene Kerboua, Sahar Omidi Shayegan, Megh Thakkar, Xing Han L, Massimo Caccia, Vronique Eglin, Alexandre Aussem, Jrmy Espinas, and Alexandre Lacoste. Lineretriever: Planning-aware observation reduction for web agents, 2025b.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning, 2021.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. Autowebglm: A large language model-based web navigating agent, 2024.

Dongjun Lee, Juyong Lee, Kyuyoung Kim, Jihoon Tack, Jinwoo Shin, Yee Whye Teh, and Kimin Lee. Learning to contextualize web pages for enhanced decision making by llm agents, 2025.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration, 2018.

Jiarun Liu, Jia Hao, Chunhong Zhang, and Zheng Hu. Wepo: Web element preference optimization for llm-based web navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26614–26622, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. Visualagentbench: Towards large multimodal models as visual foundation agents, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*, 2024.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models, 2020.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild, 2025.

Hadi Nekoei, Aman Jaiswal, Patrice Bechard, Oleh Shliazhko, Orlando Marquez Ayala, Mathieu Reymond, Massimo Caccia, Alexandre Drouin, Sarath Chandar, and Alexandre Lacoste. Just-in-time episodic feedback hinter: Leveraging offline knowledge to improve llm agents adaptation, 2025.

Tianyue Ou, Frank F. Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale, 2024.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

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. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach, 2018.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, and Aviral Kumar. Thinking vs. doing: Agents that reason by scaling test-time interaction, 2025.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3135–3144. PMLR, 06–11 Aug 2017.

Jiejun Tan, Zhicheng Dou, Wen Wang, Mang Wang, Weipeng Chen, and Ji-Rong Wen. Htmlrag: Html is better than plain text for modeling retrieved knowledge in rag systems. In *Proceedings of the ACM on Web Conference 2025*, pp. 1733–1746, 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram, Morgane Rivire, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gal Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, Andrs Gyrgy, Andr Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Pluciska, Harman Singh, Harsh

Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Pder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Lonard Hussenot. Gemma 3 technical report, 2025.

Qwen Team. Qwen3 technical report, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.

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

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining, 2023.

Ran Xu, Kaixin Ma, Wenhao Yu, Hongming Zhang, Joyce C. Ho, Carl Yang, and Dong Yu. Retrieval-augmented gui agents with generative guidelines, 2025a.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. *arXiv preprint arXiv:2412.09605*, 2024.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488. Curran Associates, Inc., 2022.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

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. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

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. Webarena: A realistic web environment for building autonomous agents, 2024a.

Yifei Zhou, Qianlan Yang, Kaixiang Lin, Min Bai, Xiong Zhou, Yu-Xiong Wang, Sergey Levine, and Erran Li. Proposer-agent-evaluator(pae): Autonomous skill discovery for foundation model internet agents, 2024b.

A Experimental Details

Training setup and hyperparameters We report the hyperparameters for training WEASEL in Table 8. We fine-tune base models with LoRA (Hu et al., 2021) adapter with the configuration in Table 8. For computing BERTScore similarities used in WEASEL, we use roberta-large (Liu et al., 2019) as the encoder. After applying WEASEL with $T_0 = 3$ to each dataset (AgentTrek and NNetNav-Live), we obtain 12K selected datapoints from AgentTrek (Xu et al., 2024) and 16K from NNetNav-Live (Murty et al., 2025); for consistency across datasets, we uniformly sample 10K datapoints from each for training.

Table 8: Training hyperparameters for fine-tuning Qwen2.5-7B-Instruct, Gemma3-4B-IT, and Qwen3-8B on AgentTrek.

Hyperparameter	**Qwen2.5-7B-Instruct**	**Gemma3-4B-IT**	**Qwen3-8B**
Training epochs	4	2	2
Training set size	10K	10K	10K
Batch size	8	16	8
Optimizer	AdamW	AdamW	AdamW
Learning rate	2e-5	2e-5	1e-6
LR schedule	Cosine	Cosine	Cosine
Mixed precision	BF16	BF16	BF16
LoRA rank	8	8	8
LoRA alpha	8	8	8
LoRA dropout	0.0	0.0	0.0

B Effect of Target-centered Pruning

In Figure 4, we further validate the effect of locality in our target-centered pruning strategy (§2.4). We shift the retained context away from the target index k_t^* while keeping the token budget identical. Concretely, for an offset $o \geq 0$, we define

$$\mathcal{K}_t^{(o)} = \Big(\big[\, k_t^* - o - w,\ k_t^* - o \,\big] \ \cup\ \{k_t^*\} \cup \big[\, k_t^* + o,\ k_t^* + o + w \,\big] \Big) \cap \{1, \ldots, K_t\}, \tag{10}$$

and keep $\tilde{V}_t^{(o)} = [\, v_{t,k} : k \in \mathcal{K}_t^{(o)} \,]$, and report the success rate in Figure 4. As o increases, the success rate decreases roughly linearly, with random pruning performing the worst. This supports that AXTree elements near the target action are the most informative, and it motivates our design of target-centered pruning.

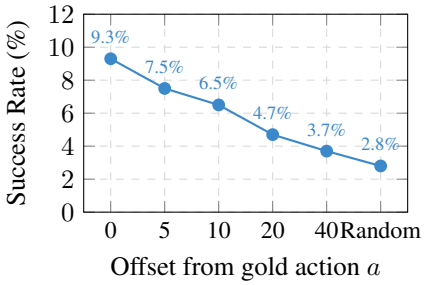


Figure 4: Success rate decreases as the pruning offset increases, while Random serves as a weak baseline. Results are reported on WebArena-Lite.

C Alternative Similarity Model for Φ and D

Table 9 evaluates how the choice of similarity encoder affects WEASEL by replacing BERTScore with a learned embedding model for computing both Φ and D. Under the same 10K-step training budget and identical hyperparameters, the embedding-based variant Qwen3-Embedding-4B (Zhang et al., 2025) still substantially outperforms the SFT baseline with random subsampling, indicating that WEASEL does not rely on a specific similarity function. In our experiments, BERTScore with roberta-large (Liu et al., 2019) yields slightly better transfer performance than the embedding alternative, and we therefore adopt it as the default in the main results.

Table 9: Effect of the similarity model used in WEASEL for computing Φ and D on WebArena-Lite. All 10K runs use the same training hyperparameters; only the similarity encoder is changed.

Method	# Steps	WebArena-Lite SR
Qwen2.5-7B-Instrcut	–	6.7
SFT (Random)	10K	8.5
WEASEL (Embedding; `Qwen3-Embedding-4B`)	10K	12.1
WEASEL (BERTScore; `roberta-large`)	10K	13.3

D PROMPT USED FOR EXPERIMENTS

D.1 LLM-AS-A-JUDGE EXPERIMENT DETAILS

We use the following prompt to score each datapoint for the LLM-as-a-judge filtering step. For generation, we set the sampling parameters to temperature of 0.0 and top-p of 0.9. After generating a score, we group datapoints by their assigned score and construct the final set by sampling equally from scores 4 and 5, yielding 10K datapoints in total.

Prompt for LLM-as-a-Judge

```
You are evaluating how useful a single state-action step is for
    solving a web task.

Task goal:
{GOAL}

State + action description (AXTree + history + assistant output):
{STATE_BLOCK}

On a scale from 1 to 5, where
- 1 = not helpful at all for achieving the goal,
- 2 = slightly helpful but not critical,
- 3 = somewhat helpful but not critical,
- 4 = helpful and critical,
- 5 = very helpful or directly crucial,
rate how much this step helps to solve the goal.

Provide 2-3 concise sentences of reasoning that cite specific
    observations/actions.
End with a line exactly like:
Score: <number between 1 and 5>
```

D.2 SELF-REASONING SYNTHESIS DETAILS

We use the prompt shown in Appendix X to self-synthesize reasoning. The prompt is largely adapted from the BrowserGym-style instructions de Chezelles et al. (2025). For generation, we set the sampling parameters to temperature of 0.2 and top-p of 0.9. After generation, we parse the model output and replace the original datasets `think` and `memory` blocks with those extracted from the generated response.

Prompt for reasoning synthesis

```
# General Instructions
```

```
You are a UI Assistant, your goal is to help the user perform tasks
using a web browser. You can communicate with the user via a chat,
in which the user gives you instructions and in which you can send
back messages. You have access to a web browser that both you and
the user can see, and with which only you can interact via specific
commands.

Review the instructions from the user, the current state of the
page and all other information to find the best possible next
action to accomplish your goal. Your answer will be interpreted and
executed by a program, make sure to follow the formatting
instructions.

## Goal:
{GOAL}

# Observation of current step:

## AXTree: {STATE_BLOCK}
# History of interaction with the task:
{HISTORY}

# Action space:

16 different types of actions are available.

noop(wait_ms: float = 1000)
    Description: Do nothing, and optionally wait for the given time
    (in milliseconds).
    Examples:
        noop()

        noop(500)

send_msg_to_user(text: str)
    Description: Sends a message to the user.
    Examples:
        send_msg_to_user('Based on the results of my search, the
        city was built in 1751.')

scroll(delta_x: float, delta_y: float)
    Description: Scroll horizontally and vertically. Amounts in
    pixels, positive for right or down scrolling, negative for left
    or up scrolling. Dispatches a wheel event.
    Examples:
        scroll(0, 200)

        scroll(-50.2, -100.5)

fill(bid: str, value: str)
    Description: Fill out a form field. It focuses the element and
    triggers an input event with the entered text. It works for
    <input>, <textarea> and [contenteditable] elements.
    Examples:
        fill('237', 'example value')

        fill('45', 'multi-line\nexample')

        fill('a12', 'example with "quotes"')

select_option(bid: str, options: str | list[str])
    Description: Select one or multiple options in a <select>
    element. You can specify option value or label to select.
    Multiple options can be selected.
```

```
    Examples:
        select_option('a48', 'blue')

        select_option('c48', ['red', 'green', 'blue'])

click(bid: str, button: Literal['left', 'middle', 'right'] =
'left', modifiers: list[typing.Literal['Alt', 'Control', 'Meta',
'Shift']] = [])
    Description: Click an element.
    Examples:
        click('a51')

        click('b22', button='right')

        click('48', button='middle', modifiers=['Shift'])

dblclick(bid: str, button: Literal['left', 'middle', 'right'] =
'left', modifiers: list[typing.Literal['Alt', 'Control', 'Meta',
'Shift']] = [])
    Description: Double click an element.
    Examples:
        dblclick('12')

        dblclick('ca42', button='right')

        dblclick('178', button='middle', modifiers=['Shift'])

hover(bid: str)
    Description: Hover over an element.
    Examples:
        hover('b8')

press(bid: str, key_comb: str)
    Description: Focus the matching element and press a combination
    of keys. It accepts the logical key names that are emitted in
    the keyboardEvent.key property of the keyboard events:
    Backquote, Minus, Equal, Backslash, Backspace, Tab, Delete,
    Escape, ArrowDown, End, Enter, Home, Insert, PageDown, PageUp,
    ArrowRight, ArrowUp, F1 - F12, Digit0 - Digit9, KeyA - KeyZ,
    etc. You can alternatively specify a single character you'd
    like to produce such as "a" or "#". Following modification
    shortcuts are also supported: Shift, Control, Alt, Meta.
    Examples:
        press('88', 'Backspace')

        press('a26', 'Control+a')

        press('a61', 'Meta+Shift+t')

focus(bid: str)
    Description: Focus the matching element.
    Examples:
        focus('b455')

clear(bid: str)
    Description: Clear the input field.
    Examples:
        clear('996')

drag_and_drop(from_bid: str, to_bid: str)
    Description: Perform a drag & drop. Hover the element that will
    be dragged. Press left mouse button. Move mouse to the element
    that will receive the drop. Release left mouse button.
    Examples:
```

```
        drag_and_drop('56', '498')

upload_file(bid: str, file: str | list[str])
    Description: Click an element and wait for a "filechooser"
    event, then select one or multiple input files for upload.
    Relative file paths are resolved relative to the current
    working directory. An empty list clears the selected files.
    Examples:
        upload_file('572', 'my_receipt.pdf')

        upload_file('63', ['/home/bob/Documents/image.jpg',
        '/home/bob/Documents/file.zip'])

go_back()
    Description: Navigate to the previous page in history.
    Examples:
        go_back()

go_forward()
    Description: Navigate to the next page in history.
    Examples:
        go_forward()

goto(url: str)
    Description: Navigate to a url.
    Examples:
        goto('http://www.example.com')

Only a single action can be provided at once. Example:
fill('a12', 'example with "quotes"')


# Abstract Example

Here is an abstract version of the answer with description of the
content of
each tag. Make sure you follow this structure, but replace the
content with your
answer:

<think>
Think step by step. If you need to make calculations such as
coordinates, write them here. Describe the effect
that your previous action had on the current content of the page.
</think>

<memory>
Write down anything you need to remember for next steps. You will
be presented
with the list of previous memories and past actions.
</memory>

<action>
One single action to be executed. You can only use one action at a
time.
</action>

# Concrete Example

Here is a concrete example of how to format your answer.
Make sure to follow the template with proper tags:

<think>
```

```
My memory says that I filled the first name and last name, but I
can't see any
content in the form. I need to explore different ways to fill the
form. Perhaps
the form is not visible yet or some fields are disabled. I need to
replan.
</think>

<memory>
I clicked on bid 32 to activate tab 2. The accessibility tree
should mention
focusable for elements of the form at next step.
</memory>

<action>
fill('a12', 'example with "quotes"')
</action>

# Instructions and Guidelines
You must regenerate the assistant's reasoning (<think>) and short
memory
(<memory>) for the next step.

- The <think> block is your internal reasoning about what to do
next.
- The <memory> block is a very short summary (12 sentences) of
important information
  that should be carried over to the next step.
- The <memory> block MUST NOT be empty. Always write at least one
sentence.
- Do NOT copy the whole conversation into <memory>. Only keep what
is useful later.
- Use the action provided below verbatim inside <action>. Do not
change or omit it.
- The action is the next step to execute; assume it has NOT been
done yet.
  Reason about why this action should be taken next, rather than
  describing
  it as already completed.

Action to execute (for your <action> block):
{action_block}

Now, for the CURRENT input, respond in EXACTLY the following
format, with
no extra text before or after the tags:

<think>
[concise reasoning consistent with the context above]
</think>

<memory>
[short memory to carry over; at least one sentence]
</memory>

<action>
{action_block}
</action>
```

E PRUNING EXPERIMENT DETAILS

Target-centered pruning. We use a fixed window size w=60 for target-centered pruning, and a larger window of w=120 for non-node actions. For `Static` nodes that do not have an index, we exclude them from the window-size count.

Semantic pruning. We rank leaf nodes by semantic similarity to a query formed by concatenating the `goal` and the `gold answer`. Similarity is based on the BERTScore (Zhang et al., 2020) using the `paraphrase-mpnet-base-v2` encoder, and we retain the top 80 leaf nodes. Each node is represented by concatenating its own text with its ancestor context (i.e., the parent chain).

Target-centered + Semantic pruning. We first select 20 leaf nodes via semantic pruning and, independently, select 60 nodes via target-centered pruning with a window size of 30. The final node set is taken as the union of the two selected sets.

Across pruning methods, we tune the pruning hyperparameters to keep the dataset-level average token count comparable.