X-TREE: TOKENIZING REUSABLE EXPERIENCE FOR EFFICIENT AGENT GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-step agents are trained on flat action streams: SFT and RLVR weight every token uniformly and ignore the sub-procedures that recur across tasks, the hierarchy that lets humans plan top-down from reusable routines. This structure sits unused, and flat training uses each scarce trajectory less fully than its content allows. Recent agents do use that structure, but only as LLM-written skills in context, never in the weights, so their gains do not generalize beyond retrieval. We instead recover this hierarchy from the data itself and train on it, with no LLM calls. Following text tokenizers, which build a vocabulary by counting alone, we score action spans by reusability and merge canonicalized actions into a *reusable eXperience tree* (X-Tree). Each X-Tree node captures how a frequent and success-bearing skill is composed from sub-skills, guiding efficient generalization. We integrate X-Tree into three training settings: offline RL, with each node as a training instance; online RLVR, with an adaptive skill bonus; and on-policy self-distillation, with X-Tree as the self-teacher's privileged context. Across WebArena, ScienceWorld, and WebShop at three model scales, X-Tree improves over standard recipes at matched data and budget by up to 4.5% SR on WebArena, 5.8% SR on ScienceWorld and 4.1% success on WebShop. Matched analyses attribute the gains to the X-Tree structure and the three integrations.

1 INTRODUCTION

Human learners spontaneously discover hierarchies that compose recurring action sequences, acting from goals through sub-goals to primitives (Miller, 1956; Botvinick et al., 2009). Language models trained to operate websites (Zhou et al., 2023), resolve repository issues (Jimenez et al., 2024), and act over long-horizon interactive environments (Xie et al., 2024; Xi et al., 2024) learn the opposite way. A trajectory enters training, whether SFT or RLVR, as a flat stream of actions, and the objective weights every token equally (Ouyang et al., 2022; Shao et al., 2024). This equal weighting overlooks the sub-procedures that recur across tasks, such as browse-and-select or pick-up-and-move. The hierarchy is present in the data, but the training objective does not use it. Uniform training therefore extracts less from each trajectory than the data allows, requiring more data to learn.

This underutilization is especially costly because experience is scarce. A trajectory is collected by a human (Wang et al., 2026a) or synthesized and verified by a model (Gandhi & Neubig, 2025). An environment must be built, its tasks written, and its verifier made trustworthy. None of this scales the way web text does. We therefore ask ***how to extract the most from a finite set of experience***.

To extract more insight from a given pool of experience, recent agent harnesses such as OpenClaw ask an LLM to summarize past experience as natural-language workflows and replay them *in context* (Wang et al., 2023; 2024). Nothing is compressed into the weights, so the experience cannot generalize beyond retrieval. On-policy self-distillation (OPSD) instead brings such skills into training, where a self-teacher reads them as privileged context and re-weights the learner's own tokens (Zhao et al., 2026; Lu et al., 2026). Both depend on an LLM to write the abstraction, which is costly, uncontrollable and leaves a library which cannot be reproduced from the data alone.

Language models already solve the analogous problem for text before learning begins. A recurrence-based tokenizer merges co-occurring characters into a structured vocabulary (Sennrich et al., 2016). In this paper, *we tokenize experience into reusable structures* (Figure 1). We canonicalize each action step in a trajectory into a typed token (*i.e.,* verb<role>). We introduce $\mathcal{X}$-Score, a measure

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

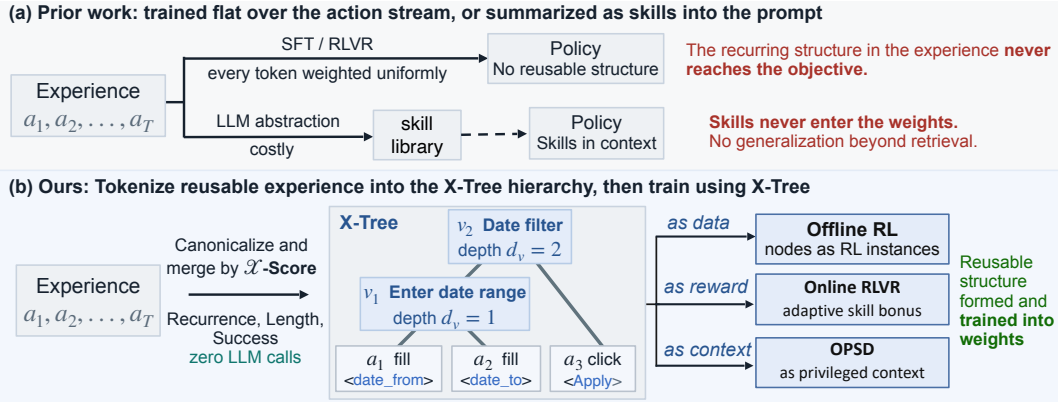


Figure 1: **Training with experience.** (a) Prior training weights every token uniformly. Harness systems use LLM-written skills in context, never in the weights. (b) We mine reusable experience by $\mathcal{X}$-Score (measuring recurrence, length, success) into the X-Tree hierarchy with zero LLM calls. Each node v is a recurring, success-bearing structure with depth d_v. We train X-Tree into model weights by integrating X-Tree as data (offline RL), as reward (online RLVR) and as context (OPSD). Figure 5 demonstrates a concrete X-Tree example from a trajectory.

of reusability that scores a span by how often it recurs, how long it is, and how often it appears in successful episodes. Guided by $\mathcal{X}$-Score, we merge adjacent pairs into *nodes*, then merge nodes into larger nodes hierarchically. This produces **reusable eXperience tree** (**X-Tree**), a deterministic, auditable hierarchy over a trajectory corpus built with zero LLM calls. Each X-Tree node captures *how a frequent and success-bearing skill is composed from sub-skills,* guiding efficient generalization.

Our preliminary result shows that X-Tree is already useful as in-context advice (Appendix C). We integrate X-Tree into three training settings (Figure 2). **1) Offline RL,** when only trajectories exist without an environment. Each X-Tree node acts as one training instance. The policy partially rolls out from the gold prefix that precedes the node. We optimize with GRPO over a step-matching reward with a depth scaled completion bonus. **2) Online RLVR,** when an environment and a verifier exist. X-Tree adds a bonus for each X-Tree skill a rollout executes with an adaptive weight that supplies signal while verifier signal is scarce and down-weights it as the signal becomes informative. **3) OPSD,** where X-Tree replaces the LLM-written skill bank that the self-teacher reads.

We experiment across three environments and three model scales at matched data and budget. Our offline RL with X-Tree gains 4.5% success rate (SR) over full-data SFT on WebArena (Zhou et al., 2023) with Qwen2.5-7B. Ablations, such as random tree, attribute the gain to X-Tree and our RL method. Our online RLVR with X-Tree stays ahead of outcome-based RL across all scales by up to 4.9% SR, leading the held-out tasks, on ScienceWorld (Wang et al., 2022). It also consistently outperforms standard recipe by up to 3.6% success and 4.6% graded score on WebShop (Yao et al., 2022). For OPSD, X-Tree as the privileged context of self-teacher matches LLM-written skills, with no LLM calls on both ScienceWorld and WebShop at all scales. We summarize our contributions:

- **X-Tree**, which tokenizes reusable experience from a trajectory pool into a deterministic hierarchy of skills with zero LLM calls. Each node captures the compositional signals to guide training.
- Three integrations of X-Tree with training, as data, reward, and context: offline RL for the trajectories-only setting; online RLVR with an adaptive, verifier-aligned skill bonus; and OPSD with X-Tree as the self-teacher's privileged context (§3.2–3.4).
- Experiments across three environments and three scales showing that X-Tree improves over standard recipes at matched data and budget by a large margin, with ablations that attribute gains to X-Tree structure and to the three integrations (§5).

2 Related Work

Hierarchy learning. Hierarchical organization of behavior is an established finding in cognitive science: capacity limits force recoding into reusable structures, and learners act from goals through sub-goals to primitives (Miller, 1956; Gobet et al., 2001; Botvinick et al., 2009), choosing decom-

positions that trade utility against planning cost (Solway et al., 2014; Correa et al., 2023). A long line of work recovers such structure from demonstrations by segmenting trajectories into options or sub-policies (Sutton et al., 1999; Konidaris et al., 2012; Fox et al., 2017; Shiarlis et al., 2018; Kipf et al., 2019; Jiang et al., 2022). We take from it the finding that bottom-up training on flat action streams overlooks structure a learner should exploit. We recover the hierarchy as a tree structure over actions handed to training rather than as a set of advice in context.

LLM-induced skills and experience reuse. Recent agents ask a proprietary model to summarize reusable experience as natural-language workflows (Wang et al., 2023; 2024; Tang et al., 2025) or executable tools (Zheng et al., 2025; Prabhu et al., 2025), or store raw experience as verbal reflections, distilled insights and retrieved trajectories (Shinn et al., 2023; Zhao et al., 2024; Zheng et al., 2024a; Su et al., 2025). Both families keep the skills in prompt or a tool registry, costing LLM calls for abstraction. Skills are never compressed into the model, not generalizing beyond retrieval. We replace the abstraction step with a deterministic miner and train the skills into the weights.

Compression as abstraction. Grammar induction over symbol sequences (Nevill-Manning & Witten, 1997) and dictionary compression (Larsson & Moffat, 2000) are the classical ancestors. Library learning compresses programs under a description-length objective (Ellis et al., 2021; Bowers et al., 2023; Grand et al., 2024). And robot policies tokenize continuous actions by vector quantization (Lee et al., 2024). Closest to us, PRISE (Zheng et al., 2024b) and Subwords-as-Skills (Yunis et al., 2023) apply Byte-Pair-Encoding (BPE) to quantized continuous-control actions. We differ in alphabet (typed actions from a canonicalizer), in merge criterion ($\mathcal{X}$-Score to measure reusability, as behavior carries more information than text), and in the use of vocabulary: not a fixed macro-action space but reusable experience as a prior to guide training as data, as credit and as context.

Multi-turn agent training. Many domains lack an online environment and a task verifier (Zhou et al., 2023; Xie et al., 2024). Our offline RL integration with X-Tree provides a solution to train from the trajectory data Gandhi & Neubig (2025) alone. Recent studies train agents online with multi-turn RL (Qi et al., 2024; Wei et al., 2025; Zhang et al., 2025) using policy-gradient (Feng et al., 2025). Unlike a learned process-reward model, our shaping signal is mined from the trajectory pool itself. On the other hand, on-policy self-distillation trains a model toward its own predictions under a privileged context (Zhao et al., 2026; Shenfeld et al., 2026; Lu et al., 2026). We keep that objective and replace the LLM-written privileged context with the X-Tree.

3 X-TREE: TOKENIZING REUSABLE EXPERIENCE

Let a trajectory be a sequence of actions $\tau = a_1 a_2 \cdots a_T$ with an episode-level success label, and $\mathcal{D}$ be a pool of trajectories for a target domain. We build X-Tree from $\mathcal{D}$ (§3.1) and integrate it with training in three settings (Figure 2), *i.e.,* Offline RL (§3.2), Online RLVR (§3.3) and OPSD (§3.4). We also integrate X-Tree with SFT training in Appendix D.

3.1 X-TREE MINING

The text BPE tokenizer (Sennrich et al., 2016) turns a character stream into a vocabulary by recursively merging the most frequent adjacent pair, which is itself a hierarchy. We apply the idea to experience with two changes: actions are first made comparable by canonicalization to reduce noise, and merges are chosen by how *reusable* the resulting pair is.

Canonicalization. We map each raw action (excluding thought) to a typed token `verb⟨role⟩`, where the role is a class or object the verb acts on, *e.g.,* `fill(bid,'2/2/26')`$\mapsto$`type⟨date⟩` (templates are in Appendix A.2). Other values in the action (*e.g.,* element id, object name) are stripped. Therefore, structurally identical actions share a symbol, which makes recurrence visible.

$\mathcal{X}$-Score: measuring reusability of a merged skill. Recurrence-based text tokenization treats each occurrence of a character pair equally. Experience carries more signal: a pair of actions is reusable when it *recurs* across the corpus, when it is *long* enough that reusing it induces better compression, and when it frequently appears in *successful* episodes.

Let u and v be two symbols that occur adjacently in the corpus, each either a canonicalized action or a node from an earlier merge, let uv be the candidate node by merging them. Let f_{uv} be the number of times the pair occurs adjacently, and ℓ_u be the number of primitive actions symbol u expands to,

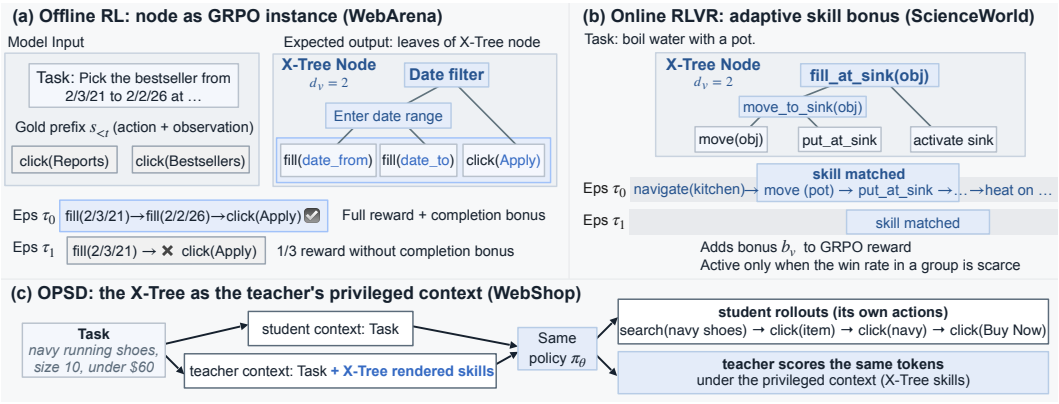


Figure 2: **Three training integrations with X-Tree.** (a) Offline RL: X-Tree nodes as RL instances, partially rolled out from gold prefix and rewarded for node completion. (b) Online RLVR: an adaptive bonus for X-Tree nodes a rollout executes plus outcome signal. (c) OPSD: the same weights, given the retrieved X-Tree skills (examples in Appendix G) as privileged context for the self-teacher.

so $\ell_u{+}\ell_v$ is the length of the candidate: a primitive action has $\ell{=}1$, merging $\texttt{type}\langle\texttt{text_input}\rangle$ with $\texttt{type}\langle\texttt{text_input}\rangle$ gives $\ell{=}2$ as it contains two primitive actions (Figure 1b). Let $\mathrm{succ}(uv)$ be the fraction of f_{uv} that lie in successful episodes. We define $\mathcal{X}$-Score of a candidate as

$$\mathcal{X}(u,v) \;=\; \underbrace{f_{uv}}_{\text{recurrence}} \cdot \underbrace{(\ell_u+\ell_v)^{p_\ell}}_{\text{length}} \cdot \underbrace{(\mathrm{succ}(uv)+\epsilon)^{p_s}}_{\text{success}}, \tag{1}$$

and recursively merge the pair with the highest $\mathcal{X}$-Score. p_ℓ and p_s set how strongly length and success weigh against recurrence (set to 1), and ϵ is a smoothing factor on the success rate. Therefore, a long, reliable, recurrent structure outranks a frequent but short or failure-prone pair, a distinction a text tokenizer cannot make.

A merge must also compress the corpus: it removes $f_{uv}-1$ symbols and adds an entry of $\ell_u{+}\ell_v$ at a cost of η, a hyper-parameter keeping the corpus from collapsing into a few giant symbols. We merge only when $f_{uv}-1 > \eta(\ell_u+\ell_v)$. Merging ends when no pair passes it or a cap is reached.

What X-Tree contains. The result of recursive merging yields a tree where the leaves are canonicalized actions. Each internal node is a composition of actions, a *skill*, whose depth d_v is one more than the deeper of its children. The whole procedure is deterministic and auditable with zero LLM calls. Applying X-Tree to a trajectory yields its *top-level tiling*, a segmentation into maximal mined skills with uncovered positions left as primitives. X-Tree is therefore a *reuse prior*: it marks which parts of a trajectory are recurring, compositional, and success-bearing (examples in Appendix B).

3.2 X-Tree for Offline RL: Each Node as one RL instance

We have $\mathcal{D}$ without an environment: no action can be executed. The only available signal is agreement with the gold trajectory, which is sparse over a long horizon. In X-Tree, each node is a reusable sub-procedure which is used as one RL training instance (Figure 2a): the input is observations and actions in the trajectory before the node, the output is actions the policy writes next (as partial rollouts), and the reward is how closely those actions match the node.

Formally, let v be a node instantiated at position t of an annotated trajectory, ℓ_v its length and d_v its depth (§3.1). Conditioned on the gold actions and observations before t (prefix $s_{<t}$), the policy generates at most ℓ_v steps. Let $\mathrm{match}_j \in \{0,1\}$ be 1 when step j reproduces the gold action: the same action and element. We concatenate the gold observation at step j from the gold trajectory if $\mathrm{match}_j{=}1$. A rollout stops when $\mathrm{match}_j{=}0$. Let $c_v = \prod_j \mathrm{match}_j$, which is 1 only when every step of the node is matched. More details in Appendix E. We adopt GRPO (Shao et al., 2024) with:

$$r \;=\; (1-\alpha)\underbrace{\tfrac{1}{\ell_v}\textstyle\sum_j \mathrm{match}_j}_{\text{step-matching reward}} + \alpha\underbrace{(1+\gamma\, d_v)}_{\text{depth factor}}\underbrace{c_v}_{\text{node completion reward}}, \tag{2}$$

with $\alpha{=}0.3$ and $\gamma{=}0.5$ as hyper-parameters. The first term is a step-matching reward for how far the rollout stays on the gold path. The second is a bonus for finishing the node. In Figure 2a, the X-Tree

node *date filter* fills two dates and clicks apply (ℓ_v=3) with d_v=2. Rollout τ_0 reproducing every action obtains full matching reward with completion bonus scaled by d_v. Rollout τ_1 that clicks apply too early stops at the second step and gets only $1/3$ matching reward, with no completion bonus.

3.3 X-TREE FOR ONLINE RLVR: AN ADAPTIVE SKILL BONUS

With an environment and a verifier, RLVR is possible. Early in training, groups with no successful rollout give GRPO zero advantage and hence no gradient. X-Tree provides the missing contrast: a rollout that carries out more mined skills is distinguished from one that does not (Figure 2b). And the signal gets down-weighted as the verifier becomes informative on its own.

Let τ_i be the i-th rollout of a group g, $R_{\text{out}}(\tau_i)$ the verifier reward, $\text{matched}(\tau_i)$ the nodes in X-Tree whose action sequence appears in τ_i (each counted once per episode): the nodes τ_i actually executes. Let b_v be the reward an execution of node v earns, specific to an environment. Let w_g be the fraction of the group the verifier counts as successes, w_{ref} a threshold, and λ_g the weight which the bonus carries in g: λ_g equals λ_0 when no rollout succeeds and falls to zero once w_g=w_{ref}, so the bonus is at full strength when the verifier cannot separate and absent on those it can. We define reward:

$$r_i \;=\; \underbrace{R_{\text{out}}(\tau_i)}_{\text{verifier}} \;+\; \lambda_g \sum_{v \in \text{matched}(\tau_i)} b_v, \qquad \lambda_g \;=\; \lambda_0 \,\text{clip}\Big(1 - \tfrac{w_g}{w_{\text{ref}}},\, 0,\, 1\Big). \tag{3}$$

b_v stands for the value of the X-Tree node v, *e.g.,* depth d_v, where a deeper node is a longer and harder composition. In Figure 2b, `fill_at_sink(obj)` is a depth-2 node. A rollout τ_0 that executes it earns the bonus $b_{\text{fill_at_sink}}$. §4 specifies b_v designs in each environment. Appendix F.2 traces λ_g over training step: an adaptive device with an operating window, not a universal add-on.

3.4 X-TREE FOR ON-POLICY SELF-DISTILLATION: A PRIVILEGED CONTEXT

OPSD trains a model on its own rollouts toward a teacher that is the *same* model conditioned on *privileged information* (Zhao et al., 2026; Shenfeld et al., 2026). A per-token distillation term is added to the RL objective, so the student receives dense token-level guidance. Hinted by Table 15, X-Tree as advice in context can enhance a model's multi-step reasoning. We therefore add X-Tree to the teacher as privileged context in OPSD. Since teacher and student are the same policy differing only in context, any gap between them is attributable to the context (LLM-written or X-Tree skills).

Let $y = (y_1, \ldots, y_{|y|})$ be the response the student sampled for a prompt s^{stu}, and s^{tea} the same prompt with retrieved X-Tree skills prepended: the response tokens are identical under both, so log-probabilities align position for position. Write $\bar{\theta}$ for the weights held fixed for the teacher's forward pass, $\delta_t = \log \pi_{\bar{\theta}}(y_t \mid s^{\text{tea}}) - \log \pi_\theta(y_t \mid s^{\text{stu}})$ for the per-token gap, positive exactly on the tokens the privileged context makes more likely. The gap also decides how much of the teacher each token copies: $g_t = \sigma(\beta\, \delta_t)$, with σ the logistic function and β the sharpness, is a confidence gate. It is near 1 where the teacher leads and near 0 where the student is already the more confident, so the student is pulled toward the teacher only on the tokens the skills inform. A coefficient c sets the weight of this term against the RL loss (set to 0.01). The objective adds one term to the RL loss:

$$\mathcal{L} \;=\; \mathcal{L}_{\text{GRPO}}(R_{\text{out}}) \;+\; c\, \underbrace{\tfrac{1}{|y|} \textstyle\sum_t g_t \left(\log \pi_{\bar{\theta}}(y_t \mid s^{\text{tea}}) - \log \pi_\theta(y_t \mid s^{\text{stu}}) \right)}_{\text{per-token distillation}}. \tag{4}$$

Minimising it raises the student probability that the privileged context finds likely. The rendered X-Tree writes each node as one line of text with its success rate and occurrence count (Appendix G.1).

4 EXPERIMENTAL SETUP

Offline RL: WebArena (Zhou et al., 2023). We mine 256 X-Tree skills from 7.9k trajectories (Gandhi & Neubig, 2025). We use Qwen2.5-7B-Instruct with SFT on half the pool, then one GRPO stage on X-Tree nodes from the other half by verl. Appendix A lists training details. Evaluation runs the 694 deterministic (non-fuzzy) tasks, 30-step cap with official reward. We evaluate success rate (SR) with normalization, *i.e.,* a task counts as success if the official evaluator returns 1, or, for the exact match tasks, if the normalized gold string appears verbatim in the final output. All baselines are based on the same harness. Appendix E shows more details.

Table 1: **Per-site SR (%) on WebArena (Qwen2.5-7B)**. Go-Browse is our reproduced result. Every RL starts from SFT on half the pool. "SFT+offlineRL" rows keep the same offline RL stage: one rewards only the outcome of a whole trajectory, the other replaces X-Tree nodes by random spans matched in node count and length. "Plain mixing" trains on all X-Tree nodes at once without completion reward. "Curriculum" trains in order of depth. "X-Tree Full" is our recipe as designed in Eq. 2. Our X-Tree with offline RL consistently outperforms other recipes. **Bold**: best per column.

Method	gitlab (187)	shopping (158)	admin (156)	reddit (112)	map (67)	wiki (14)	All (694)
w/o RL							
Qwen2.5-7B-Instruct (base)	6.4	7.8	7.7	2.4	9.0	2.4	6.5
Go-Browse (SFT-Full w/ 2 epochs)	16.8	21.7	22.9	16.4	10.4	7.1	18.4
w/ 4 epochs (matched compute)	17.5	19.4	24.4	16.1	13.4	11.9	18.8
SFT + offline RL (Ours)							
Offline RL w/ Outcome reward	15.5	23.4	23.1	15.2	7.5	14.3	18.2
Offline RL w/ Random span	18.4	20.9	26.6	14.7	11.9	14.3	19.5
SFT + offline RL (Ours) + X-Tree (Ours)							
X-Tree Plain mixing, all depths	16.1	24.5	25.4	19.9	**14.4**	**23.8**	20.7
X-Tree Curriculum, d_v=1→2...	17.8	21.5	28.6	**20.8**	12.4	**23.8**	21.2
X-Tree Full	**22.0**	**26.6**	**29.3**	18.8	10.4	14.3	**22.9**

Online RLVR: ScienceWorld (Wang et al., 2022) and WebShop (Yao et al., 2022), three scales. We use SFT warm starts from 200 (ScienceWorld) and 500 (WebShop) trajectories at 1.5B/3B/7B, then a GRPO stage. The bonus (Eq. 3) uses λ_0=0.75 and w_{ref}=0.4 on ScienceWorld, w_{ref}=0.85 on WebShop (training details in Appendix A, hyperparameter sensitivity in Appendix F.3).

i) ScienceWorld: we mine 80 X-Tree skills from 1,673 trajectories (Xi et al., 2024). We evaluate success rate (score=100) on three generalization folds (Zhang et al., 2025): G0 (seen tasks, n=1.6k), G1 (unseen variations, n=1.6k), G2 (unseen tasks, n=0.5k, trained on 19 types and evaluated on the remaining 10). A matched skill node earns $b_v = 1 + \gamma\, d_v$, granted once per episode. When two matched skills nest, the outer one is rewarded the difference between its b_v and the largest already rewarded inside it, so a nested family contributes what its largest member is worth rather than the sum. **ii)** WebShop: we mine 48 skills from 1,824 trajectories (Song et al., 2024). We evaluate two metrics on 512 held-out episodes: *success*, the fraction of purchases that meet every requirement of the task goal, and *graded score*, the partial reward in $[0, 1]$ for the requirements met. A matched X-Tree skill node earns $b_v = s_v k_v / T$: its corpus success rate s_v times the number k_v of steps in the X-Tree node that help the task goal, over the episode length T.

OPSD: ScienceWorld and WebShop, three scales. OPSD shares the X-Tree, the SFT warm start and evaluation with RLVR for both environments. It changes only the skill bank the self-teacher reads: LLM-written or the X-Tree rendering. **i)** ScienceWorld: we use our implementation of Eq. 4. The teacher reads one X-Tree rendering per task family, retrieved by the task line: the most relevant skill node to the task family. The LLM-written bank is by gpt-oss-120b from the same corpus with the same retrieval keys. **ii)** WebShop: we use the released trainer (Lu et al., 2026). The LLM-written bank is by GPT-o3. X-Tree rendered examples are in Appendix G.1.

We use Qwen2.5-Instruct and report the mean over three seeds. Appendix H shows stability analysis.

5 Results and Analysis

We validate the three training integrations (§3) over three environments and three model scales. We show the main result first and then the ablations that isolate where its gain comes from.

5.1 Offline RL on WebArena

X-Tree turns a fixed trajectory pool into +4.5 SR without any other information. With the same trajectories and no environment at training, offline RL with X-Tree reaches a 24% relative gain over Go-Browse recipe (SFT-Full) (Table 1, 22.9 *vs* 18.4). Matching RL compute with two extra SFT epochs raises SFT-Full by only 0.4, so the gain is not from compute. Our offline RL approach reliably boosts SFT performance, and adding X-Tree leads to further gains. This validates both our RL method and X-Tree structure, significantly improving the multi-turn agent learning under the same budget. Our improvement is consistent across all sites, with especially large margins

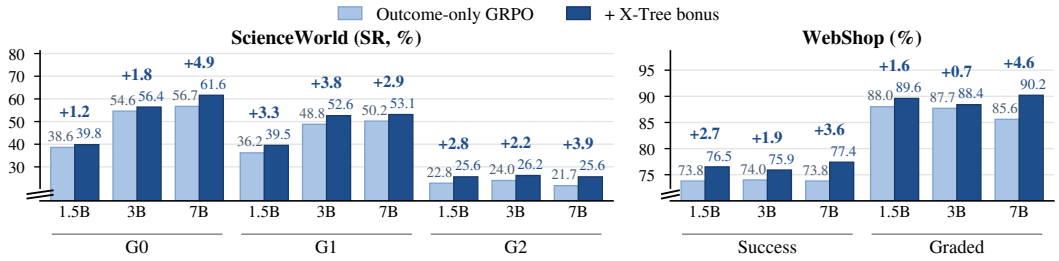


Figure 3: **Online RLVR results.** ScienceWorld SR on three generalization levels (Zhang et al., 2025); WebShop success and graded score. **Bold**: the difference of a pair. The X-Tree integration consistently outperforms outcome-only GRPO. Statistical significance in Appendix H.

on the three biggest sites compared to SFT recipe: admin improves the most (+6.4 SR), followed by gitlab, the most procedure-intensive site (+5.2 SR), and then shopping (+4.9 SR). The margin shrinks on reddit, map and wiki, since they are driven mainly by query formulation, not procedural execution. Moreover, the hierarchical reward we propose (§3.2) performs better than simply mixing X-Tree nodes with a flat reward or a depth-based curriculum training. This suggests the presence of reusable structure in trajectory data that merits further investigation. On the same budget, X-Tree captures informative signals that facilitate enhanced compositional generalization in offline RL. Our recipe also transfers to another model family (GLM-4) and larger scale (14B) (Appendix E).

5.1.1 Which part of the recipe carries the gain, the tree or the RL method?

Our recipe changes two things compared to Go-Browse SFT on the same corpus: the *training instance* (X-Tree nodes instead of whole trajectories) and a *novel RL method* (with a step-matching reward and a completion bonus). Table 2 ablates one at a time against the full recipe.

Against X-Tree. We test whether our X-Tree structure matters. Training on the whole trajectory with the same RL stage drops 3.0. Random span matches the count and length histogram of X-Tree, which keeps the chunking but moves the boundaries, which drops 3.4. A random tree with the same number of merges, which replaces $\mathcal{X}$-Score by random, drops 5.0, below SFT-only (18.4). This indicates that the reusable structure built by $\mathcal{X}$-Score provides useful training signal.

Table 2: Ablations on WebArena: *Tree Structure Validation* replaces X-Tree with other settings fixed; *Offline RL Validation* replaces the training method on identical data.

Method	SR	Δ
X-Tree Full	**22.9**	–
Tree Structure Validation		
Whole trajectory (no tree)	19.9	−3.0
Random span	19.5	−3.4
Random tree	17.9	−5.0
Offline RL Validation		
X-Tree Plain mixing	20.7	−2.2
X-Tree Curr. d_v:1→2...	21.2	−1.7
Binary outcome reward	18.2	−4.7
SFT-Full, 2 epochs	18.4	−4.5
SFT-Full, 4 epochs	18.8	−4.1

Against the RL method. We vary training on identical data (the X-Tree nodes). SFT-full is significantly behind our RL even without an environment. *Plain mixing* trains without completion bonus in Eq 2, which drops 2.2. *Curriculum* also removes the completion term and trains bottom-up by d_v=1 then 2, ..., which drops 1.7. These two variants show that X-Tree entering the reward helps harder compositions, whereas a flat reward or data curriculum under-credits reusable structures. Moreover, with X-Tree these two variants perform well on sites requiring query formulation, *i.e.,* reddit, map and wiki, while our step-matching reward and node completion bonus scaled by depth help more with procedural execution, gitlab, shopping and admin (Table 1). A binary trajectory-level reward drops 4.7, showing that the outcome signal is too sparse to learn the multi-turn reasoning. This suggests that our offline RL integration with X-Tree helps the consolidation of the reusable structure.

Takeaway. The gain comes from both our structure and method: the X-Tree node captures the composition of reusable skills which guides training with step-matching and depth-scaled reward.

5.2 Online RLVR on ScienceWorld and WebShop

Figure 3 adds the adaptive X-Tree bonus (§3.3) to outcome-only GRPO at 1.5B, 3B and 7B, from a SFT warm start on ScienceWorld and WebShop. Every pair of bars is a comparison that changes only the reward. Refer to Appendix F for further analysis. Statistical significance is in Appendix H.

ScienceWorld: X-Tree is ahead consistently and stably, with larger margin at larger scales. On seen tasks (G0, G1), X-Tree integration is above outcome-only GRPO in all comparisons and scales well with model sizes (by up to 4.9% SR). On held-out tasks (G2), X-Tree leads at every scale, by up to 3.9% SR. This indicates that X-Tree introduces reusable experience that can be captured by the policy through reward signal. Such structured experience captures how a recurrent and success-bearing skill is composed from sub-skills, which can transfer to tasks the model never trained on. We find that X-Tree is robust with a mining corpus excluding the held-out tasks (Appendix F.5).

WebShop: X-Tree is superior at every scale, consistently and stably. The adaptive X-Tree bonus improves success (by up to 3.6%) and graded score (by up to 4.6%). The gain scales well across model sizes (7B gap is the largest). This shows that the reusable structure in X-Tree and our designed training signal enhances the overall performance by facilitating multi-turn procedure progress.

5.2.1 When does the X-Tree adaptive bonus help?

The X-Tree integration with RLVR (Eq. 3) is designed to build reusable structure when the verifier signal is scarce (*e.g.,* early training stage or small rollout n), then gradually fade when it is dense. We test when the adaptive bonus truly helps by varying $rollout.n$ on ScienceWorld (Table 3).

Our recipe is ahead in $11/12$ cells. The gain is larger at smaller rollout size (from 3.6 to 4.4 at $n{=}4$), where outcome-only GRPO receives less reward signal. And the lead narrows at larger n, with potentially denser reward signal, where the adaptive bonus is designed to anneal out. This confirms that X-Tree helps with sparse reward and defers to outcome reward when the verifier signal is dense.

Table 3: Rollout number n scaling (ScienceWorld, 1.5B) on three levels.

n	Setting	G0	G1	G2
2	outcome-only	12.9	13.7	11.8
	+ X-Tree bonus	**14.6**	**15.5**	**13.0**
4	outcome-only	24.2	23.9	12.1
	+ X-Tree bonus	**28.0**	**27.5**	**16.5**
8	outcome-only	38.6	36.2	22.8
	+ X-Tree bonus	**39.8**	**39.5**	**25.6**
16	outcome-only	47.5	**49.0**	26.6
	+ X-Tree bonus	**51.0**	47.9	**27.4**

Three mechanisms (Eq. 3) explain the window. **i)** *Weight resolution*: λ_g depends on win rate among n rollouts, taking values in $\{0, \frac{1}{2}, 1\}$ at $n{=}2$. **ii)** *The gradient*: in an all-fail group the outcome terms cancel and the in-group gradient is driven by X-Tree node matching alone, keeping the training alive. **iii)** *Anneal versus learning speed*: the measured late-training λ_g exposure falls from 0.50 at $n{=}2$ to 0.18–0.26 at $n{=}16$, so the bonus is active while the policy is weak and fades once it is strong. Training dynamic of λ_g is in Appendix F.2.

Takeaway. The adaptive bonus is active when the verifier is silent (*e.g.,* early training stage or a small n) and fades when it is dense, which builds reusable structure to facilitate multi-step reasoning.

5.2.2 Is the gain from the extra data used to mine X-Tree?

In the main results, all experiments are trained by SFT and RLVR with identical data, whereas X-Tree is mined from a trajectory corpus. One question is whether X-Tree's gain comes from the mining corpus. We conduct resource-matched experiments (WebShop, 1.5B, Table 4). We mine X-Tree from the SFT data and vary the data volume, so that all resources are the identical. We find that our X-Tree still manages to capture reusable structures to better guide training with exactly the same budget.

Table 4: Resource-matched runs (WebShop, 1.5B). All rows use same resources and differ only in reward.

Warm start	outcome	+X-Tree (Δ)
500 Samples	73.8	**74.9** $(+1.1)$
1,016 Success	74.9	**78.1** $(+3.2)$
1,824 All	76.4	**77.4** $(+1.0)$

Mining X-Tree does not require many trajectories. We re-mine X-Tree from random subsets of the corpus, with other settings (*e.g.,* data, training) fixed (Table 5). The 100-trajectory X-Tree is already above outcome reward with 500 sample SFT warm start (75.2 vs. 73.8). Success reaches 76.5 with full corpus, not monotonically in between, showing the robustness and efficiency of building X-Tree.

Table 5: Mining-corpus size (WebShop, 1.5B). X-Tree is mined from N random trajectories of the corpus.

# Traj	Skills	Success (%)
50	9	68.7
100	11	75.2
200	16	72.9
500	24	74.4
1,824	48	**76.5**

Takeaway. X-Tree wins by the structure and reward design. The hierarchy, which captures how a reusable skill is composed from sub-skills, facilitates generalization under the same budget. Mining X-Tree structure does not require a large trajectory pool.

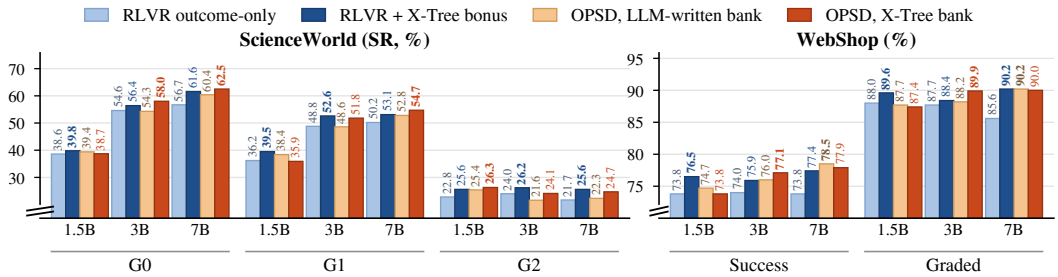


Figure 4: **Validating OPSD with X-Tree.** ScienceWorld SR on three levels (Zhang et al., 2025); WebShop success and graded score. OPSD with X-Tree changes only the privileged context of the self-teacher. X-Tree is comparable with LLM-written bank. Statistical significance in Appendix H.

5.3 OPSD ON SCIENCEWORLD AND WEBSHOP

Figure 4 validates OPSD with X-Tree against the online RLVR on ScienceWorld and WebShop at 1.5B, 3B and 7B. RLVR and OPSD integrate the same X-Tree in two ways, as a reward bonus or as the privileged context of the self-teacher. Within OPSD only the skill bank changes, LLM-written (gpt-oss-120b on ScienceWorld, GPT-o3 on WebShop) or the rendered X-Tree.

X-Tree performs on par with the LLM-written skill bank at zero LLM cost. Across both environments, different generalization levels and three model scales, our rendered X-Tree is comparable to the LLM-written bank, ahead at most of the runs (by up to $+3.7$ on ScienceWorld and $+1.7$ on WebShop). This indicates that X-Tree, mined deterministically with no LLM in the loop under the same budget, is as good to the self-teacher as an LLM-written artifact for the same environment.

X-Tree provides useful signals for OPSD, with larger models pushing better performance over RLVR. OPSD with our X-Tree is above outcome-only RLVR in most cells, and the margin grows with the model scale. At 7B, OPSD reaches $+5.8$ SR on ScienceWorld and $+4.4$ graded score on WebShop, while at 1.5B and 3B the gains stay within a few points. OPSD distills from the model's own predictions under the skill text, so its gain depends on how well the base model reads and follows that text. Compared to the X-Tree bonus in RLVR, OPSD is ahead at 7B on ScienceWorld seen tasks (G0, G1) and comparable at smaller scales, while RLVR is ahead of outcome-only throughout. This indicates that the two integrations of X-Tree are both effective ways of incorporating the reusable structure: X-Tree enters RLVR as a reward signal which helps from smaller to larger models; while X-Tree, acting as privileged context in OPSD, further improves the performance once the model can well exploit the skill text. We ablate the skill bank in Appendix G.

6 CONCLUSION

Limited experience is the current bottleneck in training multi-step agents. Standard recipes consume the action sequence uniformly and bottom-up, without the hierarchy that lets human learners generalize more efficiently with the same data. We tokenize reusable experience to recover that hierarchy from a trajectory pool as X-Tree, with zero LLM calls. Each X-Tree node captures how a recurrent and success-bearing skill is composed from sub-procedures, guiding efficient generalization. X-Tree enters training in three settings: as the training instance in offline RL, as an adaptive bonus that keeps online RLVR ahead while the outcome signal is scarce, and as the privileged context in OPSD where it matches the LLM-written skill bank. Analyses attribute the gain to the X-Tree structure and our integration design. These results hold across three environments and three model scales.

Limitations. We mine X-Tree once from a fixed pool, and integrate X-Tree with training without improving it. Mining from the policy's own trajectories would potentially let the structure and the policy improve together in a loop. We evaluate the three integrations one per setting and never combine them. A possible future work is to train a single model on X-Tree nodes, taking the X-Tree bonus, and distill from X-Tree at once. X-Tree also only reorganizes experience that already exists. Because it records which pairs recur and which succeed, it marks where a corpus is thin, identifying the trajectories worth synthesizing to fill the gap.

AI USE STATEMENT

We used an AI assistant (Claude Code) in two roles. For writing, it polished text drafted by the authors and typeset tables from the result files the authors provided. The authors reviewed and edited all of its output. For retrieval and discovery, it helped locate related work and verify bibliographic details, which the authors read and selected. The research questions, methods, experiments and conclusions are the authors' own, and the authors take full responsibility for the content.

ETHICS STATEMENT

This work uses publicly available benchmarks and simulated environments. It does not involve human-subject experiments, collection of personally identifying information, or deployment that affects individuals. We follow the ICLR Code of Ethics and report the methods and limitations of the study transparently.

REPRODUCIBILITY STATEMENT

The canonicalizers, miner and training configurations and the evaluation harness will be released with the paper. The evaluation protocols are stated in §4. The spread behind every cell of the online tables is in Appendix H.

REFERENCES

Kaikai An, Fangkai Yang, Liqun Li, Junting Lu, Sitao Cheng, Shuzheng Si, Lu Wang, Pu Zhao, Lele Cao, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Baobao Chang. Thread: A logic-based data organization paradigm for how-to question answering with retrieval augmented generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. arXiv:2406.13372.

Matthew M. Botvinick, Yael Niv, and Andrew C. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

Maddy Bowers, Theo X. Olausson, Lionel Wong, Gabriel Grand, Joshua B. Tenenbaum, Kevin Ellis, and Armando Solar-Lezama. Top-down synthesis for library learning. *Proceedings of the ACM on Programming Languages*, 7(POPL), 2023.

Sitao Cheng, Liangming Pan, Xunjian Yin, Xinyi Wang, and William Yang Wang. Understanding the interplay between parametric and contextual knowledge for large language models. *arXiv preprint arXiv:2410.08414*, 2024a.

Sitao Cheng, Ziyuan Zhuang, Yong Xu, Fangkai Yang, Chaoyun Zhang, Xiaoting Qin, Xiang Huang, Ling Chen, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Call me when necessary: LLMs can efficiently and faithfully reason over structured environments. *arXiv preprint arXiv:2403.08593*, 2024b.

Sitao Cheng, Tianle Li, Xuhan Huang, Xunjian Yin, and Difan Zou. Differentiable evolutionary reinforcement learning. *arXiv preprint arXiv:2512.13399*, 2025a.

Sitao Cheng, Xunjian Yin, Ruiwen Zhou, Yuxuan Li, Xinyi Wang, Liangming Pan, William Yang Wang, and Victor Zhong. Atomic skills are the prerequisite: When reinforcement learning synthesizes compositional reasoning, and when it only amplifies. *arXiv preprint arXiv:2512.01970*, 2025b.

Carlos G. Correa, Mark K. Ho, Frederick Callaway, Nathaniel D. Daw, and Thomas L. Griffiths. Humans decompose tasks by trading off utility and computational cost. *PLoS Computational Biology*, 19(6):e1011087, 2023.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021.

Lang Feng et al. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025. NeurIPS 2025.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

Apurva Gandhi and Graham Neubig. Go-browse: Training web agents with structured exploration. *arXiv preprint arXiv:2506.03533*, 2025.

Fernand Gobet, Peter C. R. Lane, Steve Croker, Peter C.-H. Cheng, Gary Jones, Iain Oliver, and Julian M. Pine. Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, 5(6): 236–243, 2001.

Gabriel Grand, Lionel Wong, Maddy Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. Lilo: Learning interpretable libraries by compressing and documenting code. In *International Conference on Learning Representations*, 2024.

Xiang Huang, Sitao Cheng, Yuhong Bao, Shanshan Huang, and Yuzhong Qu. MarkQA: A large scale KBQA dataset with numerical reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023a. arXiv:2310.15517.

Xiang Huang, Sitao Cheng, Yiheng Shu, Yuheng Bao, and Yuzhong Qu. Question decomposition tree for answering complex questions over knowledge bases. In *AAAI Conference on Artificial Intelligence*, 2023b.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. QueryAgent: A reliable and efficient reasoning framework with environmental feedback-based self-correction. *arXiv preprint arXiv:2403.11886*, 2024a.

Xiang Huang, Jiayu Shen, Shanshan Huang, Sitao Cheng, Xiaxia Wang, and Yuzhong Qu. TARGA: Targeted synthetic data generation for practical reasoning over structured data. *arXiv preprint arXiv:2412.19544*, 2024b.

Yiding Jiang, Evan Zheran Liu, Benjamin Eysenbach, J. Zico Kolter, and Chelsea Finn. Learning options via compression. In *Advances in Neural Information Processing Systems*, 2022.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=VTF8yNQM66`.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. *arXiv preprint arXiv:2411.13504*, 2024.

Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, 2019.

George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3): 360–375, 2012.

N. Jesper Larsson and Alistair Moffat. Offline dictionary-based compression. *Proceedings of the IEEE*, 88(11):1722–1732, 2000.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *International Conference on Machine Learning*, pp. 26991–27008, 2024.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, 1956.

Craig G. Nevill-Manning and Ian H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Viraj Prabhu et al. Walt: Web agents that learn tools. *arXiv preprint arXiv:2510.01524*, 2025.

Zehan Qi et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2016.

Zhihong Shao et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pp. 4654–4663, 2018.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. Optimal behavioral hierarchy. *PLoS Computational Biology*, 10(8): e1003779, 2014.

Yifan Song et al. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*, 2024.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö. Arık. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. *arXiv preprint arXiv:2501.10893*, 2025.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

Xiangru Tang, Tianyu Hu, Muyang Ye, Daniel Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, Arman Cohan, and Mark Gerstein. ChemAgent: Self-updating memories in large language models improves chemical reasoning. In *International Conference on Learning Representations*, 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. *arXiv preprint arXiv:2305.16291*, 2023.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, et al. Opencua: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems*, 38:139756–139806, 2026a.

Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, et al. Webxskill: Skill learning for autonomous web agents. *arXiv preprint arXiv:2604.13318*, 2026b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Zhepei Wei et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025.

Zhiheng Xi et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *arXiv preprint arXiv:2207.01206*, 2022.

David Yunis, Justin Jung, Falcon Dai, and Matthew Walter. Subwords as skills: Tokenization for sparse-reward reinforcement learning. *arXiv preprint arXiv:2309.04459*, 2023.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *arXiv preprint arXiv:2507.22844*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *AAAI Conference on Artificial Intelligence*, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Boyuan Zheng et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, 2024a.

Ruijie Zheng et al. Prise: Llm-style sequence compression for learning temporal action abstractions in control. *arXiv preprint arXiv:2402.10450*, 2024b.

Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. RuleArena: A benchmark for rule-guided reasoning with LLMs in real-world scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. arXiv:2412.08972.

Ruiwen Zhou, Maojia Song, Xiaobao Wu, Sitao Cheng, Xunjian Yin, Yuxi Xie, Zhuoqun Hao, Wenyue Hua, Liangming Pan, Soujanya Poria, and Min-Yen Kan. Epistemic context learning: Building trust the right way in LLM-based multi-agent systems. *arXiv preprint arXiv:2601.21742*, 2026.

Shuyan Zhou et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. EfficientRAG: Efficient retriever for multi-hop question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. arXiv:2408.04259.

A IMPLEMENTATION DETAILS

A.1 TRAINING DETAILS

Tables 6 to 8 show the hyperparameters behind experiments. Table 8 covers the X-Tree in-context evaluation in Appendix C. The offline stage is our node-as-instance offline RL based on verl. ScienceWorld runs with the code of Zhang et al. (2025) with our reward bonus. WebShop's RLVR is based on verl-agent Feng et al. (2025), and the OPSD is based on the code of (Lu et al., 2026).

Hinted by Table 5, building X-Tree does not need a large trajectory pool to help. However, previous studies (Cheng et al., 2025b) show that RL generalization requires a decent SFT-built base. Every online run in this paper therefore starts from a small warm start: on ScienceWorld and WebShop we sample 200 and 500 trajectories from the trajectory pool. Within every comparison we report, all experiments (including RLVR and OPSD) start from the same warm-start checkpoint for a model scale in an environment.

A.2 CANONICALIZATION EXAMPLES

The purpose is to better recognize reusable structures in the corpus. Tables 9 to 11 show the templates from a raw action to a canonical token. Properties of the template are three-fold: **1)** *deterministic*: written once per environment from the action space, **2)** *complete*: no action can fail to canonicalize and none is silently dropped, and **3)** *no loss of information*: every trimmed value, *e.g.,* an element id, typed text, an object name, is kept so it could still be used after X-Tree mining. Chain-of-Thought never enters the alphabet.

On ScienceWorld the object class $c(\cdot)$ is an ordered keyword lexicon of nine classes, tried in order: `door`, `instrument`, `paint`, `container`, `device`, `plant`, `animal`, `substance`, `location`, with the literal head noun kept when nothing matches and `location` the closed list of the ten ScienceWorld rooms. Instance numbers and containment qualifiers are stripped before the match and preserved. This is what makes `move thermometer to metal pot` and `move thermometer to ceramic cup` the same symbol, $\texttt{put}\langle c(Y)\rangle = \texttt{put}\langle\texttt{container}\rangle$, while keeping both distinct from `move thermometer to stove`. Simplified examples in Figure 1 and 2 are for illustration purpose.

B A CASE STUDY

Tables 12–14 study X-Tree nodes from the three environments to illustrate the semantics. First, the boundaries are meaningful: the most frequent WebArena depth-2 node is a two-field form followed by its submit button, and the same canonical node instantiates as a report filter on the shop admin site and as a route query on the map, which are reusable structures of real-world meanings. Second, depth composes: the ScienceWorld chain S8 $\rightarrow$ S19 $\rightarrow$ S20 $\rightarrow$ S21 adds one primitive per depth, from "go to the kitchen via the hallway" to "fetch the thermometer," so a depth-6 node is a subprocedure while its descendants remain reusable on their own. Tree-structured decomposition also helps answer complex questions over knowledge bases (Huang et al., 2023b). X-Tree mines such a tree from trajectories instead. Third, the $\mathcal{X}$-Score separates structures that the verifier rewards from those it does not: on WebShop, the node that engage an option type named by the goal (color, size, flavor) succeed in 81–97% of their occurrences, the universal macros (search, click item, buy) sit at the corpus mean of 0.56, and the pagination chains and untyped option clicks sit at 0.14–0.36. This is the separation the $\mathcal{X}$-Score merge and the option-recall gate exploit (§5.2.2). The libraries also contain low-value macros (WebArena's menu-then-submenu click pair recurs 1,219 times), which is why the offline reward scales with depth.

One trajectory through the tree. Figure 5 tokenizes one trajectory with the ScienceWorld X-Tree. The task is *to measure the temperature of an unknown substance and put it in a box that depends on the reading*. The agent starts in the art studio and fetches the thermometer from the kitchen, which reads 1 °C, below the task's threshold of 50 degrees, so it uses the blue box. That last choice rests on a numerical comparison. The merges turn the 18 actions into six tokens: four X-Tree nodes and two primitives that stay unmerged. The first token, S21, is the depth-6 chain of Table 13 that fetches the thermometer from the kitchen. The second, S29, walks to the bedroom and looks

Table 6: **X-Tree construction and SFT hyperparameters.** The canonicalization and merging are per environment. Merging stops at the compression constraint §3.1 or the skill cap.

Setting	WebArena	ScienceWorld	WebShop
***X-Tree Mining** (Eq. 1)*			
p_ℓ, p_s	1, 1	1, 1	1, 1
Success floor ϵ	0.001	0.001	0.001
η	0.3	0.5	0.5
Skill cap	256	80	60
Max. skill length (actions)	12	8	8
Min. pair frequency	4	5	5
Min. skill success rate	0	0.9	0
Min. task types per skill	1	2	1
Mining corpus (trajectories)	7,974	1,673	1,824
Primitive action types	139	84	17
Skills mined (bound by)	256 (cap)	80 (cap)	48 (compression)
SFT warm start			
Epochs	2	5	5
Learning rate	2×10^{-5}	1×10^{-5}	1×10^{-5}
Global batch (sequences)	4	256	256
Max. sequence (tokens)	24,000	4,096	8,192

Table 7: **RL hyperparameters for the three environments.**

Setting	WebArena (offline)	ScienceWorld	WebShop
Rollouts per group n	2	8	8
Optimizer steps	805	100	150
Units per step	6 nodes	8 tasks	16 tasks
Learning rate	5×10^{-7}	1×10^{-6}	1×10^{-6}
KL to the warm start	none	0.01	0.01
Mini-batch	2	256	64
Max. prompt (tokens)	16,384	6,000	4,096
Max. response (tokens)	16,384	1,024	512
Temperature (train / eval)	0.7 / 0	1.0 / 0.4	1.0 / 0.4
Episode cap (env. steps)	node horizon	30	15
Invalid-action penalty	–	0.1	0.1
Reward constants			
α, γ (Eq. 2)	0.3, 0.5	–	–
λ_0, w_{ref} (Eq. 3)	–	0.75, 0.4	0.75, 0.85
Node bonus b_v	–	$1 + 0.5\, d_v$	$s_v k_v / T$, $\times 10$
Self-distillation coefficient c (OPSD)	–	0.01	0.01
Gate sharpness β (Eq. 4)	–	5	0

around. Both are built on S2 (open the door to the hallway, go there, open the next door), which occurs 1,803 times in the corpus and here completes once toward the kitchen and once toward the bedroom. Under the node-per-row SFT format of Appendix D, each token is one training row, so this trajectory gives six rows instead of 18.

C X-TREE AS SKILLS IN THE CONTEXT

We test X-Tree as as advice in context under an interaction harness (Huang et al., 2024a; Cheng et al., 2024b) shared with four LLM-abstraction baselines (AWM (Wang et al., 2024), SkillWeaver (Zheng et al., 2025), WALT (Prabhu et al., 2025), WebXSkill (Wang et al., 2026b)) and a no-skill control. Only the induction algorithm differs. Other settings are identical. We re-implement the baselines based on the same harness and base model for fair comparison. We use the same

Table 15: **X-Tree as advice in context** (MiniWoB++, SR, %). Only the skill bank in context changes. Four LLM-written baselines and X-Tree.

Skill bank	Qwen2.5-7B	Qwen3-8B	Qwen3-14B
no skills	44.9	60.4	68.1
AWM	41.4	59.9	69.0
SkillWeaver	47.6	61.3	68.3
WALT	46.3	58.1	68.5
WebXSkill	43.8	59.8	68.6
X-Tree	**49.8**	**65.1**	**72.6**

Table 8: **X-Tree construction and deployment for the prompted-agents setting** (Table 15). Merging follows Eq. 1 and stops at the compression constraint or the skill cap.

Setting	MiniWoB++
Backbones	Qwen2.5-7B, Qwen3-8B, Qwen3-14B
Mining corpus	episodes of the evaluated backbone on the 53 training task families, 40 per family
Trajectories (successful)	2,125 (886) / 2,120 (1,116) / 2,120 (1,239)
Filtering	none, failures kept with their label
Canonical action	`verb⟨role⟩`, role from the DOM (Table 10)
p_ℓ, p_s	1, 1.5
Success floor ϵ	0.001
η	0.3
Skill cap	200
Max. node length	10
Min. pair frequency	4
Min. node success rate	0
Skills mined (bound by)	90 / 71 / 60 (compression)
Deployment	advisory hint per step, top-8 skills by utterance n-gram
Evaluation	42 task families (26 seen, 16 unseen) $\times$ 20 seeds $=$ 840 episodes

Table 9: **Canonicalization on WebArena.** A BrowserGym action string to `verb⟨role⟩`: verb is from the function name, role from the accessibility-tree role of the element the action addresses.

Raw action	Canonical	Kept in slots
***Verb**, from the BrowserGym function name*		
`click(bid)`, `dblclick(bid)`	`click`	element id
`fill(bid, text)`, `type(bid, text)`	`type`	element id, typed text
`select_option(bid, opt)`	`select`	element id, option value
`press(bid, key)`, `keyboard_press(key)`	`key`	element id, key name
`hover`, `focus`, `clear`, `scroll`	unchanged	element id
`goto(url)`, `go_back()`, `go_forward()`	`navigate`	url
`drag_and_drop(from, to)`	`drag`	both element ids
`upload_file(bid, path)`	`upload`	element id, path
`send_msg_to_user(m)`, `report_infeasible(m)`	`stop`	message text
`noop()`	`noop`	—
***Role**, from the accessibility tree of the addressed element*		
`link`	`link`	element name
`button`	`button`	element name
`checkbox`, `radio`, `tab`	unchanged	element name
`option`, `menuitem`, `menuitemcheckbox`, `menuitemradio`	`option`	element name
`textbox`, `searchbox`	`text_input`	element name
`combobox`, `listbox`	`select`	element name
any other role	the role verbatim	element name

model as inference to mine skills for the baselines. Note that our X-Tree requires zero LLM calls. Table 8 lists the detailed settings. Results are shown in Table 15.

On MiniWoB++, X-Tree leads every baseline at three backbones, by up to 3.8%, while some baselines fall below the no-skill setting, since the LLMs that write the skills are relatively small. Agents tend to conform to misleading input (Zhou et al., 2026). And how a model combines the context with its own knowledge is uncertain (Cheng et al., 2024a; Jin et al., 2024). Training the skills into the weights, as in our main experiments, avoids that dependence. This validates the quality of reusable structure in X-Tree without integration with training, built deterministically without LLM calls.

D X-TREE INTEGRATION WITH SFT

Segmentation. Standard SFT trains one action per training row with a full context (system prompt, goal, observation, history) for every row. We test X-Tree integration with SFT by *one X-Tree node per row*: the target is the node's full action span (several actions generated one time) and the context

Table 10: **Canonicalization on MiniWoB++.** The verb is from the agent's primitive action. The role is from the addressed DOM element by the ordered rule below, first matching row wins. The element label (nearest label, ARIA label, text, placeholder, value or id) is kept in the slots. In our corpus the environment reports every input as a typed tag, so the last rule assigns its role.

Raw action or element	Canonical	Kept in slots
Verb		
`click(ref)`, `submit(ref)`, `clear(ref)`	`click`, `submit`, `clear`	element label
`noop`	`noop`	—
`type(ref, text)`	`type`	typed text, element label
`select(ref, value)`	`select`	option value, element label
`key(k)`	`key`	key name
***Role**, from the DOM element (first matching rule)*		
ARIA role `button`, `link`, `checkbox`, `radio`, `tab`, `dialog` or `option`	the ARIA role	element label
tag `button`, or `input` of type `submit` or `button`	`button`	element label
tag `a`	`link`	element label
tag `select` / `option`	`select` / `option`	element label
`input` of type `checkbox` / `radio`	`checkbox` / `radio`	element label
tag `textarea`, or `input` of any other type	`text_input`	element label
class contains `tab` / `dialog`, or tag `dialog`	`tab` / `dialog`	element label
tag `p`, `span`, `div`, `label`, `td`, `th`, `li`	`text`	element label
any other tag, *e.g.,* `input_text`, `input_checkbox`	the tag name (`element` if none)	element label

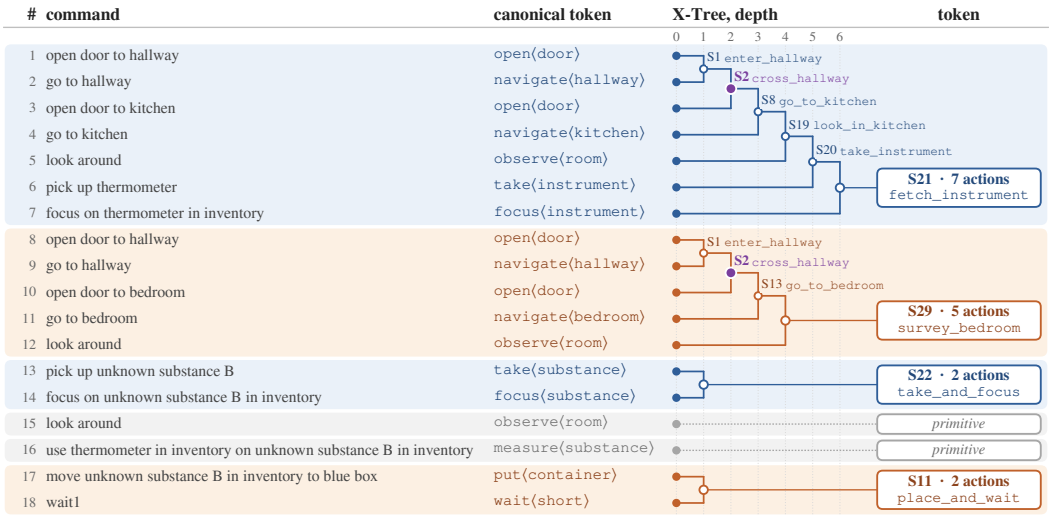


Figure 5: **An example trajectory "tokenized" by the X-Tree (ScienceWorld).** Task: *measure the temperature of unknown substance B in the bedroom, then place it in the green box above 50 degrees and in the blue box below.* Rows are the trajectory's 18 actions with their canonical tokens. The tree beside them is the X-Tree over the actions, with depth growing to the right, and each shaded band is one token of the tiling. S2 (purple) recurs inside two different tokens.

is encoded once per node. A length-matched sliding window baseline is the random cut over action stream with matched length as X-Tree node. The three formats supervise the identical action streams (*i.e.,* trained over the same trajectory corpus). On the ScienceWorld 300-random-episode corpus, the vanilla SFT costs 1,766 tokens per supervised action against 446 for the X-Tree node SFT, a $4.0\times$ saving, with 97.6% of raw's tokens being repeated prompts under the same training settings.

Test-time protocol. A node-trained policy outputs one turn per node: a `<planning>` block and one `<action>` tag per action. The evaluator executes the tags in order, one environment step each, and returns no observation until the turn ends (the last action is executed). The next prompt carries the observation after the last executed action and the turn's joined action string, which is the same

Table 11: **Canonicalization on ScienceWorld and WebShop.** Both environments take text commands, so the rule is a template match rather than a role lookup. X and Y are object phrases, and $c(\cdot)$ is the nine-class lexicon defined in the text.

Command	Canonical primitive	Kept in slots
ScienceWorld		
`focus on` X	`focus`$\langle c(X)\rangle$	X
`go to` X, `teleport to` X	`navigate`$\langle X\rangle$ if X is a room, else `navigate`$\langle c(X)\rangle$	X, teleport flag
`open` X, `close` X	`open`$\langle c(X)\rangle$, `close`$\langle c(X)\rangle$	X
`pick up` X	`take`$\langle c(X)\rangle$	X
`put down` X, `drop` X	`put`$\langle$`ground`$\rangle$	X, surface form
`move` X `to` Y	`put`$\langle c(Y)\rangle$	X, Y
`activate` X, `deactivate` X	`activate`$\langle c(X)\rangle$, `deactivate`$\langle c(X)\rangle$	X
`pour` X `into` Y, `pour` X `in` Y	`pour`$\langle c(Y)\rangle$	X, Y
`dunk` X `into` Y, `dunk` X `in` Y	`dunk`$\langle c(Y)\rangle$	X, Y
`mix` X	`mix`$\langle c(X)\rangle$	X
`look around`	`observe`$\langle$`room`$\rangle$	—
`look at` X, `look in` X, `examine` X	`observe`$\langle c(X)\rangle$	X, surface form
`inventory`	`observe`$\langle$`inventory`$\rangle$	—
`task`, `reset task`	`observe`$\langle$`task`$\rangle$	surface form
`read` X	`read`$\langle c(X)\rangle$	X
`use` X `on` Y	`measure`$\langle c(Y)\rangle$ if X is a thermometer, stopwatch or ruler, else `use`$\langle c(Y)\rangle$	X, Y
`use` X	`use`$\langle c(X)\rangle$	X
`connect` X `to` Y	`connect`$\langle c(Y)\rangle$	X, Y
`disconnect` X	`disconnect`$\langle c(X)\rangle$	X
`eat` X, `flush` X	`eat`$\langle c(X)\rangle$, `flush`$\langle c(X)\rangle$	X
`wait1`, `wait`	`wait`$\langle$`short`$\rangle$, `wait`$\langle$`long`$\rangle$	—
a bare integer	`disambiguate`$\langle$`choice`$\rangle$	the integer
anything else (catch-all)	the first word as verb, the rest as role	rest of the command
WebShop		
`search[`q`]`	`search`$\langle$`query`$\rangle$	query text
`click[`X`]`, X a product id	`click`$\langle$`item`$\rangle$	the product id
`click[`X`]`, X a value the goal names	`click`$\langle$`option-`*`type`*$\rangle$	the value
`click[`X`]`, any other value	`click`$\langle$`option`$\rangle$	the value
`click[buy now]`	`click`$\langle$`buy`$\rangle$	—
`click[back to search]`	`click`$\langle$`back`$\rangle$	—
`click[next >]`, `click[< prev]`	`click`$\langle$`next`$\rangle$, `click`$\langle$`prev`$\rangle$	—
`click[description]`	`click`$\langle$`desc`$\rangle$	—
`click[features]`	`click`$\langle$`features`$\rangle$	—
`click[reviews]`	`click`$\langle$`reviews`$\rangle$	—
`click[attributes]`	`click`$\langle$`attrs`$\rangle$	—

Table 12: **Case study (a): X-Tree nodes in WebArena.** Canonical nodes are written in the `verb`$\langle$`role`$\rangle$ tokens of Table 9. Occ. counts the occurrence of the node. The grounded occurrence reads element labels from the page's accessibility tree.

Canonical node	Depth	Occ.	Grounded occurrence	Reading
`type`$\langle$`text_input`$\rangle$ → `type`$\langle$`text_input`$\rangle$ → `click`$\langle$`button`$\rangle$	2	209	fill "From *" 01/01/2025; fill "To *" 03/24/2025; click "Show Report" (shop admin); fill "From" Times Square; fill "To" Central Park; click "Go" (map)	complete a two-field form and submit: a report filter and a route query are the same node
`type`$\langle$`text_input`$\rangle$ → `click`$\langle$`button`$\rangle$	1	430	fill searchbox "Search for projects, issues, etc." *project*; click "Search" (GitLab)	type a query and submit it
`type`$\langle$`text_input`$\rangle$ → `type`$\langle$`text_input`$\rangle$	1	88	fill "From" Grand Central Station; fill "To" Empire State Building (map)	the two-field entry that recurs inside the first row
`click`$\langle$`link`$\rangle$ → `click`$\langle$`link`$\rangle$ → `click`$\langle$`link`$\rangle$	2	161	click "Forums"; click a page link; click the next page (Reddit)	navigate, then paginate
`click`$\langle$`link`$\rangle$ → `click`$\langle$`link`$\rangle$	1	1,219	click "SALES"; click a submenu entry (shop admin)	menu navigation

node-level history the SFT rows are built with. An action that the environment rejects returns its error message as that step's observation. The remaining tags still execute. Then, the policy sees the failure at its next turn. Every run has the same budget of 30 environment actions per episode. The standard SFT row executes one action per turn as in (Zhang et al., 2025).

Table 13: **Case study (b): X-Tree nodes in ScienceWorld.** ScienceWorld nodes are long, so each canonical node is written as its merge of two children, in the primitives of Table 11. Occ. counts occurrences in the 1,673 AgentTraj-L trajectories.

Canonical node	Depth	Occ.	Grounded occurrence	Reading
S8=S2+navigate⟨kitchen⟩	3	415	open door to hallway; go to hallway; open door to kitchen; go to kitchen	go to the kitchen via the hallway
S31=put⟨device⟩+S30	4	151	move metal pot containing gallium to stove; activate stove; examine gallium; use thermometer on gallium; examine; use thermometer	heat a substance and monitor its temperature
S60=navigate⟨foundry⟩+S59	6	25	go to foundry; open blast furnace; move metal pot to blast furnace; activate blast furnace; examine lead; use thermometer on lead; examine; use thermometer	melt a metal and monitor it
S5=observe⟨room⟩+S4	4	381	look around; connect battery anode to black wire terminal 1; connect battery cathode to blue wire terminal 1; connect black wire terminal 2 to cathode in buzzer; …	wire a circuit from battery through wires to a load
S27=S26+take⟨animal⟩	4	241	open door to outside; go to outside; look around; focus on baby wolf; pick up baby wolf	find and collect an animal outdoors
S43=S7+put⟨container⟩	4	90	open door to hallway; go to hallway; open door to workshop; go to workshop; move baby wolf in inventory to blue box	carry the animal to a container in another room
S15=wait⟨short⟩+S14	3	381	wait1; look around; move aluminum foil to blue box; wait1	observe, then sort an object into a box

Table 14: **Case study (c): X-Tree nodes in WebShop.** Canonical nodes are written in the primitives of Table 11. Occ. counts occurrences in the 1,824 ETO trajectories. ETO is the corpus with failures, so the fraction of a node's occurrences in successful trajectories (corpus mean 0.56) is reported.

Canonical node	Depth	Occ.	Success	Reading
search⟨query⟩ → click⟨item⟩ → click⟨option-color⟩ → click⟨buy⟩	3	94	0.81	engage the goal's color option before buying
search⟨query⟩ → click⟨item⟩ → click⟨option-size⟩ → click⟨option-color⟩ → click⟨buy⟩	4	25	0.92	two typed options, both named by the goal
search⟨query⟩ → click⟨item⟩ → click⟨option-flavor-name⟩	2	31	0.97	a typed option the goal names
click⟨option-color⟩ → click⟨buy⟩	1	139	0.84	the closing move once the right option is set
search⟨query⟩ → click⟨item⟩ → click⟨buy⟩	2	741	0.57	buy without touching options: a universal macro at the corpus mean
search⟨query⟩ → click⟨item⟩	1	1,770	0.57	a universal macro
search⟨query⟩ → click⟨next⟩ → click⟨next⟩ → click⟨next⟩ → click⟨item⟩	4	10	0.30	paging deep into the results: failure-associated
search⟨query⟩ → click⟨next⟩ → click⟨item⟩ → click⟨buy⟩	3	14	0.14	buying after paging past the first results
search⟨query⟩ → click⟨item⟩ → click⟨option⟩ → click⟨option⟩ → click⟨option⟩ → click⟨buy⟩	4	14	0.36	untyped options clicked in sequence

SFT integration with X-Tree outperforms the standard one at all scales (Table 16). While consuming $4\times$ fewer tokens for the identical supervised actions, our X-Tree with SFT leads by $+4.9$, $+2.4$ and $+9.1$. The length-matched baseline further validates the effectiveness of X-Tree: a sliding window with the same span lengths is on par with the vanilla SFT, underperforming X-Tree significantly. This indicates that X-Tree truly merges meaningful reusable structures that guides training to save compute budget and improve performance.

Table 16: **SFT at 1.5B/3B/7B on ScienceWorld** (SR pooled over G0 and G1).

SFT row	1.5B	3B	7B
one standard action	11.7	16.7	20.8
length-matched span	12.0	17.4	19.3
an X-Tree node	**16.6**	**19.1**	**29.9**

E Offline RL on WebArena

E.1 Rollout observations for offline RL

We have the trajectory pool providing the thought, action and observation at each interaction turn. At an X-Tree node boundary the policy is given the gold prefix (*i.e.,* previous observation and action steps) and output one action per turn. If the action reproduces the exact recorded gold action (thought excluded) and the node has steps left, the recorded next observation in the trajectory is appended and the rollout continues. If it does not, the rollout ends there. So the loop never has to invent an

Table 17: Official evaluator and normalized SR (WebArena, Qwen2.5-7B, mean over three passes).

Method	official SR	SR (main text)
base model	5.8	6.5
SFT-Full (Go-Browse), 2 epochs	17.2	18.4
SFT-Full, 4 epochs	17.4	18.8
whole-trajectory GRPO	18.7	19.9
whole-trajectory GRPO, binary outcome reward	17.6	18.2
random-node GRPO	18.5	19.5
random-tree GRPO	17.0	17.9
X-Tree plain mixing	19.3	20.7
X-Tree sequential	20.2	21.2
X-Tree mixing + hier. reward	21.5	22.9

observation for a state the gold trajectory does not contain. The step-matching reward of Eq. 2 is the fraction of the node's steps reached before that stop. Observations are replayed in Go-Browse's compact form and the whole prefix is retained.

E.2 Evaluation protocol

Every run is scored on the same 694 deterministic tasks (the fuzzy-match tasks of the 812-task set are excluded) under one harness, three passes per run with greedy decoding. The evaluation has no randomness. The pass-to-pass variance comes from environment state and service timing. All sites are reset between runs. We report normalized SR mean over passes. Every per-site row is validated against its pass totals. Statistical significance is in Appendix H. Table 17 lists the official evaluator's success rate next to the normalized SR (Qwen2.5-7B). The normalizer is frozen in code and applied identically to all runs. It recovers 4–11 of the 45 `exact_match` tasks per pass.

The WebArena X-Tree node set used for training is $5{,}148$ X-Tree nodes, $3.5\text{k}/1.2\text{k}/0.4\text{k}$ of them at depth 1/2/3 (the three levels take most of the nodes). The tree-block of Table 2 are built from that same node source and trained from the same SFT warm start with the settings of Table 7. Random spans match the X-Tree nodes exactly in row count ($5{,}148$) and in horizon histogram. The random tree matches the row count but is shallower by construction since it replaces $\mathcal{X}$-Score by random pairing (88% of its nodes are two actions, against 43% for the mined tree). The whole-trajectory baseline has one span per trajectory ($1{,}989$ rows), so we train for equivalent gradient steps to match the compute with our recipe for fair comparison.

The released Go-Browse-WA-7B model scores 18.0 SR (19.0 normalized SR) on our 694 deterministic tasks as the mean of three passes, against a published 21.7% on the 812-task set under the same harness. The gap is within what the excluded fuzzy-match tasks and the harness differences account for. Every run in the paper is scored under the same harness.

E.3 Transferability to larger scale and model family

Table 18 shows the transferability of our recipe discussed in §5.1. On Qwen2.5-14B, our recipe adds $+1.6$ over full-data SFT. On GLM-4-9B, it adds $+2.0$ over the SFT-full recipe. This demonstrates that our offline RL integration with X-Tree can transfer to a larger scale and another model family, further showing the effectiveness of X-Tree and the integration method.

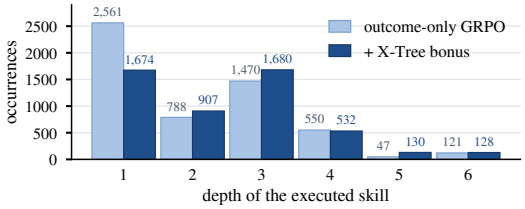
Table 18: Scale and model-family transfer of offline RL with X-Tree (WebArena, SR, %).

Backbone	SFT-Full	+X-Tree recipe	Δ
Qwen2.5-7B	18.4	**22.9**	$+4.5$
Qwen2.5-14B	24.1	**25.7**	$+1.6$
GLM-4-9B	14.9	**16.9**	$+2.0$

F Online RLVR on ScienceWorld

F.1 Quantitative analysis of the trained policy on X-Tree skills

We test whether the X-Tree skills end up in the trained policy. We trace the actions the trained policy takes at evaluation and match them against the X-Tree nodes the reward matched (1.5B on G0 Figure 6). We find that *skill coverage does not separate the two policies, but the depth does.*



	outcome-only	+X-Tree
episodes with a mined skill	94.6%	99.3%
distinct skills executed	51	50
skills per episode	3.33	3.04
mean executed depth	2.11	**2.39**
episodes with a depth≥3 skill	85.4%	**87.5%**

Figure 6: What the trained policies execute at test time (1.5B, G0 fold). Left: occurrences of executed X-Tree skills by depth. Right: properties of the trained models.

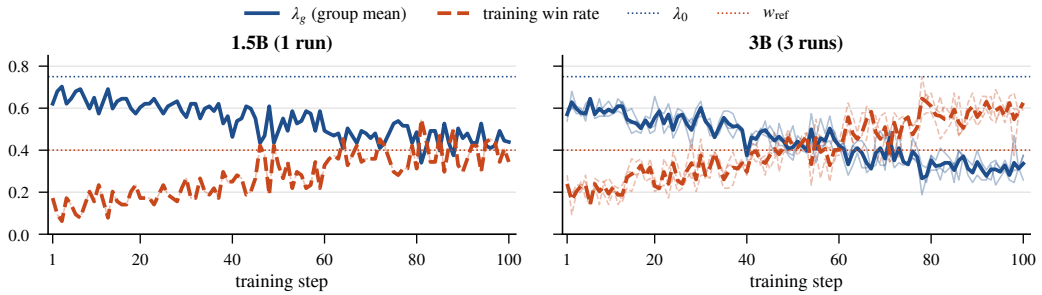


Figure 7: **How λ_g anneals during training (ScienceWorld).** Per training step, the group-mean weight λ_g of Eq. 3 (solid) and the training win rate (dashed). λ_0=0.75 and w_{ref}=0.4 are dotted. 3B: the three runs behind Figure 3 (faint) and their mean (bold). 1.5B: one run.

The outcome-only policy executes a comparable number of X-Tree skills in its episodes. However, the X-Tree bonus policy's histogram moves mass out of lower depth into depth-3 and depth-5. The mean executed depth rises from 2.11 to 2.39, and our recipe executes *fewer* skills per episode, which indicates that the policy runs fewer and longer composites. This matches the finding that RL composes the atomic skills a model already holds into larger ones (Cheng et al., 2025b).

F.2 The dynamic of λ_g over training steps

The weight λ_g in Eq. 3 is a dynamic property of training rather than a direct add-on. It down-weights the bonus as a task's group win rate approaches w_{ref}. Figure 7 traces the logged group-mean λ_g against the training win rate on ScienceWorld, step by step, for runs of Figure 3. The two scales anneal at the rates the learning curves dictate: the 3B win rate rises from $\sim$ 0.2 to $\sim$ 0.6 and the mean λ_g falls from 0.57 to 0.34, while at 1.5B the win rate ends at 0.34 and λ_g is 0.44. On WebShop, where w_{ref} is 0.85, the dynamic takes λ_g from 0.71 at a 2% win rate to 0.11 at 72%. λ_g never reaches zero on a task the policy has not learned perfectly, which is within expectation: the shaping fades by competence, not by step count. Reward structures can also be discovered by automated search (Cheng et al., 2025a). Ours keeps a fixed form and adapts only its weight to the win rate.

F.3 Sensitivity to λ_0 and w_{ref}

The two hyperparameters of Eq. 3 are selected by the training-side validation: take the run with the highest validation success points. No testing set numbers enter the choice. We ran two ablations in opposite directions at 1.5B, n=8 (Table 19): a gentler weight and a front-loaded one that weights early episodes more and anneals sooner. Both remain comparable to or above outcome-only GRPO and below our setting. So the recipe is not very sensitive to the choice of both hyperparameters.

Table 19: **Sensitivity to λ_0 and w_{ref}** (ScienceWorld, 1.5B, n=8, SR on G0 / G1).

Setting	λ_0	w_{ref}	G0 / G1
outcome-only GRPO	–	–	38.6 / 36.2
gentler weight	0.5	0.4	39.4 / 37.6
front-loaded, anneals sooner	1.5	0.25	38.9 / 36.4
selected (Ours)	0.75	0.4	39.8 / 39.5

F.4 Ablation of X-Tree structure

We ablate X-Tree by replacing the skill set the adaptive bonus reads (Table 20). The first ablation draws spans at arbitrary positions, which leaves the bonus far fewer matching. The second control

Table 21: **Results with held-out tasks removed from the mining corpus** (ScienceWorld, 1.5B). Top: X-Tree changes from the held-out tasks removed. Bottom: performance with X-Tree original and re-mined, everything else fixed.

	full corpus	**testing task types removed**
trajectories after the length filter	1,673	1,248
# X-Tree nodes	80	80 (66 kept, 14 new)
grounded skills the bonus reward can match	2,698	1,305
G2 SR, outcome + X-Tree bonus (three runs)	**25.6**	24.8
G2 SR, outcome-only GRPO		22.8

Table 22: **The X-Tree rendering example for the color option**, as the self-teacher reads it.

```
### TASK: goal requires a color option ###
The instruction names a color.  Expert episodes that selected the goal's color on the
product page before buying, and how they fared:
- search<query> -> click<item> -> click<option-color> -> click<buy>:  81% success (94
episodes, ABOVE base) - search for the product; open a result item; select the goal's
color option; click Buy Now.
- click<option-color> -> click<option> -> click<buy>:  62% success (34 episodes,
ABOVE base) - select the goal's color option; select an option the goal does NOT ask
for (wasted step); click Buy Now.
[four more chains]
Plan:  search with the product words plus the color value; open a matching item;
click the exact color value the instruction asks for (it must be visible on the
page); select any other required options the same way; then Buy Now.
```

keeps everything except $\mathcal{X}$-Score to merge: 80 skills with the length and depth histograms of X-Tree, each drawn at random from canonical action pairs. Other settings are fixed for fair comparison.

The ablation validates our X-Tree structure. On G0, only our $\mathcal{X}$-Score mined bank is above outcome-only GRPO ($+1.2$ against -0.5 and -0.2 for the two controls). On G1, our X-Tree leads by $+3.3$, with random spans recovering $+2.2$ of it, indicating the superiority of X-Tree. On the held-out fold, three banks are comparable. We therefore attribute the bonus's gain to the X-Tree structure mined with $\mathcal{X}$-Score.

Table 20: Ablation of X-Tree (ScienceWorld, 1.5B). SR (%), difference to outcome-only in parentheses. "#Match" is the number of nodes the bonus can match.

Skills	**# Match**	**G0**	**G1**	**G2**
Empty (outcome-only)	–	38.6	36.2	22.8
Random spans	576	38.4 (-0.2)	38.4 ($+2.2$)	–
Random tree	2,109	38.1 (-0.5)	36.8 ($+0.6$)	**26.7** ($+3.9$)
X-Tree	2,698	**39.8** ($+1.2$)	**39.5** ($+3.3$)	25.6 ($+2.8$)

F.5 Robustness of X-Tree from a held-out mining corpus

In the main part, we test X-Tree on ScienceWorld with three generalization levels (Zhang et al., 2025; Huang et al., 2024b; 2023a). A question is whether X-Tree mined from a corpus never covering the held-out task can transfer. We test by removing the testing types from the mining corpus (Table 21). We observe that the grounded skills our bonus reward matches drops by about half. Training the held-out runs (G2, 1.5B) with the re-mined X-Tree with other settings fixed, X-Tree still lead over outcome-only GRPO by $+2.0$, showing that X-Tree still works well with a smaller matching surface. This suggests the robustness and transferability of X-Tree, and that X-Tree does not require a strong and large scale trajectory pool to perform, echoing findings in Table 5.

G OPSD on WebShop

G.1 The X-Tree rendered privileged context

The self-teacher reads X-Tree as plain text by a fixed rendering template with no LLM calls. The template produces one rendering per goal-option type (color, size, flavor, etc) and one general rendering. A rendering opens with a header naming the option type, then lists up to six mined X-Tree-node-based chains selecting that option, ranked by corpus success rate, one line each in a fixed format: the canonical chain, success rate and occurrence count, a tag against the corpus base rate (`above`, `at`, `below`), and a gloss that maps every token to a fixed phrase (`click`⟨`option-color`⟩ to "select

Table 24: **Spread behind ScienceWorld.** SR (%) as mean ± sample standard deviation behind Figure 3 and the OPSD results of Figure 4 (top) and Table 3 (bottom, 1.5B).

Scale	Setting	G0	G1	held-out G2
1.5B	Outcome-only GRPO	38.6 ± 2.5	36.2 ± 0.3	22.8 ± 1.3
	+ X-Tree bonus	39.8 ± 1.0	39.5 ± 1.7	25.6 ± 1.3
	+ OPSD, LLM-written bank	39.4 ± 1.7	38.4 ± 2.1	25.4 ± 2.0
	+ OPSD, X-Tree bank	38.7 ± 2.7	35.9 ± 1.8	26.3 ± 1.8
3B	Outcome-only GRPO	54.6 ± 6.0	48.8 ± 3.1	24.0 ± 3.8
	+ X-Tree bonus	56.4 ± 3.7	52.6 ± 5.6	26.2 ± 2.3
	+ OPSD, LLM-written bank	54.3 ± 1.4	48.6 ± 2.2	21.6 ± 0.9
	+ OPSD, X-Tree bank	58.0 ± 7.7	51.8 ± 5.4	24.1 ± 1.5
7B	Outcome-only GRPO	56.7 ± 8.4	50.2 ± 3.6	21.7 ± 1.6
	+ X-Tree bonus	61.6 ± 8.7	53.1 ± 4.6	25.6 ± 1.5
	+ OPSD, LLM-written bank	60.4 ± 6.5	52.8 ± 5.1	22.3 ± 2.8
	+ OPSD, X-Tree bank	62.5 ± 5.1	54.7 ± 4.5	24.7 ± 1.9
Budget	**Setting**	**G0**	**G1**	**held-out G2**
n=2	Outcome-only GRPO	12.9 ± 5.4	13.7 ± 6.8	11.8 ± 6.6
	+ X-Tree bonus	14.6 ± 3.2	15.5 ± 3.5	13.0 ± 2.0
n=4	Outcome-only GRPO	24.2 ± 3.4	23.9 ± 3.3	12.1 ± 7.8
	+ X-Tree bonus	28.0 ± 0.9	27.5 ± 1.3	16.5 ± 4.9
n=16	Outcome-only GRPO	47.5 ± 2.6	49.0 ± 3.1	26.6 ± 3.2
	+ X-Tree bonus	51.0 ± 1.5	47.9 ± 2.2	27.4 ± 1.9

the goal's color option", and a `click⟨option⟩` the goal did not ask for to "wasted step"). It closes with a one-sentence plan, which states the task's rules outright (Zhou et al., 2025). Table 22 shows the rendering for the color option.

The general rendering states the corpus and lists the option-free chains in the same format. Retrieval is the released trainer's keyword rule: an instruction that contains "color:" receives the general rendering followed by the color rendering, prepended to teacher's prompt. Organizing procedural knowledge into such logic units helps the models understanding (An et al., 2025), and a better retriever is worth future exploration (Zhuang et al., 2024).

G.2 Ablation on skill bank to the self-teacher

We ablate the skill bank to the self-teacher in OPSD. Table 23 changes only the privileged context with other settings fixed. An empty privileged context leaves it at the outcome-only level, so the distillation mechanism on its own does not produce the gain. Scrambling the words of the X-Tree bank, which keeps its vocabulary but destroys its chains, drops 0.6. The X-Tree remains the most effective way to maintain the useful information from a trajectory pool with no LLM call, the comparison Figure 4 draws against the LLM-written bank.

Table 23: **Ablation on skill bank to the self-teacher (WebShop, 3B).**

Teacher's bank	Success
outcome-only RLVR: 74.0	
Empty (no privileged text)	74.2
X-Tree, words scrambled	76.5
Mined X-Tree	**77.1**

H Stability and scope: the spread behind each mean

The main tables carry means only for better illustration. This section states how each mean is formed and how stable it is. All results are means over three seeds of the same configuration, same as compared methods and prior work. A paired comparison takes the per-run difference between a run and its control and reports its mean Δ together with $t = \Delta/(\mathrm{sd}(\Delta)/\sqrt{n})$ over the n paired runs.

ScienceWorld. Table 24 gives the mean and standard deviation (std) of the runs that enter each cell. The std ranges are widest at 7B and at the smallest rollout budgets based on outcome reward. The X-Tree-based run is tighter in most runs, so the gain is stable, not bought with extra variance. The held-out G2 column is the noisiest at every scale.

WebShop. Table 25 gives every cell with its spread. Std ranges from 1.6 to 3.9 points on success and 1.0 to 2.3 on the graded score. The X-Tree bonus is ahead of outcome-only on every evaluation

Table 25: **Spread behind WebShop.** Success and graded score (%) as mean ± sample standard deviation behind Figure 3 and Figure 4. Δ (t) is the paired success difference against the outcome-only row of the same scale and its t-statistic.

Scale	Setting	Success	Graded	Δ (t)
1.5B	RLVR, outcome-only	73.8 ± 1.9	88.0 ± 1.0	—
	+ X-Tree bonus	76.5 ± 1.6	89.6 ± 1.1	$+2.7$ (7.0)
	OPSD, LLM-written bank	74.7 ± 1.9	87.7 ± 1.0	$+0.9$ (0.8)
	OPSD, X-Tree bank	73.8 ± 1.7	87.4 ± 1.1	$+0.0$ (0.1)
3B	RLVR, outcome-only	74.0 ± 3.0	87.7 ± 1.6	—
	+ X-Tree bonus	75.9 ± 3.5	88.4 ± 1.6	$+1.9$ (2.4)
	OPSD, LLM-written bank	76.0 ± 2.0	88.2 ± 1.2	$+2.1$ (2.6)
	OPSD, X-Tree bank	77.1 ± 1.6	89.9 ± 1.2	$+3.2$ (4.6)
7B	RLVR, outcome-only	73.8 ± 3.9	85.6 ± 1.8	—
	+ X-Tree bonus	77.4 ± 1.9	90.2 ± 1.0	$+3.6$ (2.1)
	OPSD, LLM-written bank	78.5 ± 3.0	90.2 ± 2.3	$+4.7$ (8.4)
	OPSD, X-Tree bank	77.9 ± 2.7	90.0 ± 1.5	$+4.1$ (3.1)

Table 26: Resource-matched controls and mining-corpus size, with their spreads (WebShop, 1.5B). Success (%) as mean ± sd over three runs. *Top*: both runs of a row share the warm start and the RL data, and X-Tree is mined from the SFT trajectory. "Anchor" is the warm start's own success before RL. *Bottom*: the X-Tree is mined from N random trajectories.

	Anchor	outcome-only	+X-Tree
X-Tree mined from the warm start trajectories			
500 samples	24.2	73.8 ± 1.9	74.9 ± 1.0
1,016 successful	33.1	74.9 ± 1.5	78.1 ± 2.1
1,824 all	32.5	76.4 ± 4.1	77.4 ± 1.1
N trajectories in mining corpus. All trained from the 500-sample warm start			
N=50, 9 skills	-	-	68.7 ± 0.7
N=100, 11 skills	-	-	75.2 ± 1.9
N=200, 16 skills	-	-	72.9 ± 2.1
N=500, 24 skills	-	-	74.4 ± 0.8
N=1,824 (full), 48 skills	-	-	$\mathbf{76.5} \pm 1.6$

run of every cell of Figure 3, which is the property a mean alone does not show. Table 26 adds the spreads behind Tables 4 and 5, showing the stability of our results.

WebArena. For the main results (Table 1), the spread is 0.44 to 1.15 points of SR. Our full recipe is 22.9 ± 0.6 (passes $23.5/22.8/22.3$) against 18.4 ± 0.6 for the SFT-Full reference ($18.7/18.7/17.7$), so the $+4.5$ gap is about seven times the larger of the two spreads.