WIST: Web-Grounded Iterative Self-Play Tree for Domain-Targeted Reasoning Improvement

Fangyuan Li[1,2*], **Pengfei Li**[1,2*], **Shijie Wang**[3], **Junqi Gao**[1], **Jianxing Liu**[1†], **Biqing Qi**[3†], **Yuqiang Li**[2,3†]
[1]Harbin Institute of Technology
[2]Shanghai Innovation Institute
[3]Shanghai Artificial Intelligence Laboratory
lifangyuan0212@gmail.com, jx.liu@hit.edu.cn, qibiqing@pjlab.org.cn, liyuqiang@pjlab.org.cn

Abstract

Recent progress in reinforcement learning with verifiable rewards (RLVR) offers a practical path to self-improvement of language models, but existing methods face a key trade-off: endogenous self-play can drift over iterations, while corpus-grounded approaches rely on curated data environments. We present **WIST**, a **W**eb-grounded **I**terative **S**elf-play **T**ree framework for domain-targeted reasoning improvement that learns directly from the open web without requiring any pre-arranged domain corpus. WIST incrementally expands a domain tree for exploration, and retrieves and cleans path-consistent web corpus to construct a controllable training environment. It then performs Challenger–Solver self-play with verifiable rewards, and feeds learnability signals back to update node posteriors and guide subsequent exploration through an adaptive curriculum. Across four backbones, WIST consistently improves over the base models and typically outperforms both purely endogenous self-evolution and corpus-grounded self-play baselines, with the Overall gains reaching **+9.8** (*Qwen3-4B-Base*) and **+9.7** (*OctoThinker-8B*). WIST is also domain-steerable, improving *Qwen3-8B-Base* by **+14.79** in medicine and *Qwen3-4B-Base* by **+5.28** on PhyBench. Ablations further confirm the importance of WIST's key components for stable open-web learning. Our Code is available at https://github.com/lfy-123/WIST.

1 Introduction

Self-improvement of large language models (LLMs) without human supervision (Clune, 2019; Pourcel et al., 2025) is a key step toward more general intelligence. Recent progress in reinforcement learning with verifiable rewards (RLVR) (Hurst et al., 2024; Guo et al., 2025) shows that when feedback comes from automatically checkable outcomes (e.g., mathematical correctness, program execution, or deterministically verifiable structured outputs), LLMs' reasoning can be reliably strengthened. Unlike costly human annotation, such feedback can be generated at scale, enabling iterative generate–evaluate–update cycles at low marginal cost and offering a practical path toward self-evolving language models.

Figure 1: Comparison of R-Zero, SPICE, and our WIST.

Motivated by this goal, prior work has explored several routes to self-improvement. A common approach bootstraps from seed data or existing task collections via self-training and synthetic data generation (e.g., STaR (Zelikman et al., 2022), MetaMath (Yu et al., 2023), Self-Instruct (Wang et al., 2023)). Another line (Zhao et al., 2025a) adopts self-play and automatic curricula in verifiable environments (especially code), often implementing *generate–verify–learn* loops with adversarial or cooperative role specialization. More recently, R-Zero (Huang et al., 2025) pursues fully endogenous self-evolution by creating tasks from zero external data and deriving rewards from internal signals such as self-consistency. In contrast, SPICE (Liu et al., 2025a) emphasizes external knowledge: it treats a large corpus as an environment and uses corpus-grounded verifiable QA under information asymmetry to mitigate hallucination accumulation and stagnation. Despite these advances, a core tension remains: purely endogenous generation

*Equal contribution
†Corresponding authors

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 31563–31585
July 2-7, 2026

can drift and degrade over iterations, while corpus-dependent approaches rely on curated sources and often struggle to cover specialized domains, shifting the burden to building and maintaining high-quality data environments.

We propose **WIST**, a **W**eb-grounded **I**terative **S**elf-play **T**ree framework that improves reasoning by enabling models to discover and learn domain-relevant knowledge from the open web. The open web is rich but noisy and unstructured, making domain-relevant verifiable signals hard to extract. Inspired by prior work on structuring such data for learning (Gao et al., 2025; Cao et al., 2025), WIST organizes exploration with a dynamically expanding domain tree: starting from a user-specified domain label, the model incrementally decomposes the domain into finer-grained concepts down to leaf-level knowledge points. Each sampled root-to-leaf path then triggers corpus acquisition, where WIST retrieves and cleans path-consistent web documents to construct a lightweight, continuously refreshed corpus pool. Conditioned on the retrieved corpus, WIST runs a Challenger–Solver self-play loop with verifiable rewards, and converts the resulting learnability feedback into node-wise posterior updates. These posteriors guide subsequent path sampling, yielding an adaptive curriculum that increasingly focuses on the model's weak yet learnable regions. Compared with fully endogenous self-evolution (e.g., R-Zero), WIST grounds training in retrieved corpus to mitigate signal drift; compared with corpus-grounded self-play (e.g., SPICE), WIST removes reliance on a fixed curated corpus by expanding coverage through structured open-web exploration (Figure 1). We present additional related work in Appendix B.

Empirically, WIST delivers consistent gains across diverse backbones. For example, on *Qwen3-4B-Base*, WIST improves the Overall score from 33.3 to 43.1 (+9.8), outperforming R-Zero (40.6) and SPICE (41.8); on *Qwen3-8B-Base*, it reaches 46.7 (vs. 42.1 base), exceeding R-Zero (45.5) and SPICE (46.0); and on *OctoThinker-8B-Hybrid-Base*, it improves from 22.9 to 32.6 (+9.7), surpassing both baselines. Moreover, WIST is inherently domain-steerable: by switching only the target domain label to **physics**, it yields measurable gains on PhyBench (EED score 4.73 $\rightarrow$ 10.01 in 50 steps), demonstrating that open-web corpus can support domain-specific self-evolution without relying on any carefully curated domain corpus. Ablations further show that reward-guided exploration stabilizes training and that the tree structure is essential for maintaining coverage and reliably mining high-value knowledge from the open web.

Our contributions include:

- We introduce **WIST**, a web-grounded self-play Tree framework that enables *domain-targeted* reasoning improvement without requiring a manually curated domain corpus.
- We propose a dynamically expanding domain tree with posterior-guided path sampling, which structures open-web exploration and induces an adaptive curriculum from learnability feedback.
- We demonstrate strong gains on mathematical and general reasoning benchmarks across multiple backbones, and validate domain steering to physics through systematic ablations.

2 Preliminaries

2.1 Reinforcement Learning with Verifiable Rewards

Reinforcement Learning with Verifiable Rewards (RLVR) is a paradigm for fine-tuning models in domains where response quality can be deterministically verified. Given a prompt (question) x, a policy LLM π_θ generates an answer $\hat{y} \sim \pi_\theta(\cdot \mid x)$. RLVR assumes a rule-based verifier

$$v : \mathcal{Y} \times \mathcal{Y} \rightarrow \{0, 1\}, \tag{1}$$

which compares a generated answer $\hat{y}$ against a reference answer y^* and returns 1 if $\hat{y}$ is equivalent to y^* under task-specific criteria (e.g., normalized exact match, symbolic equivalence, or deterministic format constraints), and 0 otherwise. This induces a binary reward:

$$r(\hat{y}; y^*) \triangleq v(\hat{y}, y^*). \tag{2}$$

Such verifiable rewards are especially effective for tasks with unambiguous correctness (e.g., mathematical reasoning) and form the basis of the Solver training reward in our work.

2.2 Group Relative Policy Optimization Done Right

We optimize π_θ using Dr. GRPO (Liu et al., 2025b), a group-based policy optimization method tailored to RLVR that avoids value-function fitting. For each prompt x, we sample a group of G responses $\{\hat{y}_i\}_{i=1}^{G}$ and compute rewards $\{r_i\}_{i=1}^{G}$. Dr. GRPO

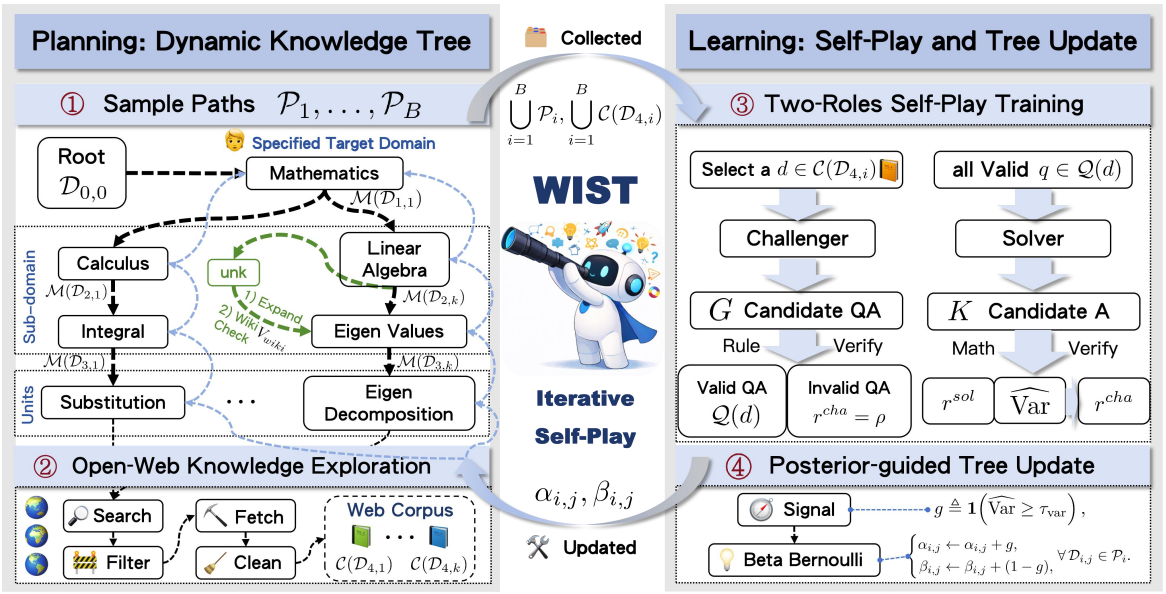


Figure 2: Overview of our proposed WIST, a web-grounded iterative self-play Tree framework for domain-targeted reasoning improvement.

uses a group-centered advantage:

$$A_i \triangleq r_i - \frac{1}{G}\sum_{j=1}^{G} r_j, \tag{3}$$

and applies a PPO-style clipped objective at the token level with a *global* normalization constant (instead of length-normalization), reducing length-related optimization bias. In our implementation, Dr. GRPO is used as the advantage computation for our training pipeline.

3 Methodology

3.1 Overview

We propose **WIST**, a web-grounded iterative self-play tree framework for domain-targeted reasoning improvement (Algorithm in Appendix A; overview in Figure 2). WIST closes the loop between *where to explore* and *what is learned* by sampling a path on a dynamically expanding domain tree, retrieving and cleaning path-aligned web documents into leaf-level corpus pools, running corpus-grounded Challenger–Solver self-play with verifiable rewards, and feeding the resulting learnability signal back to update node posteriors for subsequent exploration and curriculum budget allocation.

WIST has three coupled components: (1) a **self-expanding domain tree** for exploration planning that decomposes the target domain into leaf-level concepts; (2) an **open-web retrieval, filtering, and corpus construction pipeline** that constructs and attaches cleaned corpus pools to leaf nodes; and (3) a **tree-guided curriculum** that updates node-wise Beta posteriors and performs Thompson sampling with a sliding-window update to track non-stationary learning.

3.2 Dynamic Domain Tree Construction

To support controllable and scalable exploration in an open-web environment, we organize the target domain as a hierarchical tree and expand it incrementally during training. The tree decomposes coarse-grained topics into finer-grained concepts, so that sampled leaf nodes correspond to searchable and verifiable knowledge units for subsequent web retrieval and self-play.

Hierarchical domain tree $\mathcal{T}$. We represent the target domain as a directed tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ with maximum depth $\mathcal{L}$. Nodes are organized into layers $\{\mathcal{V}_i\}_{i=0}^{\mathcal{L}}$, where $\mathcal{V} = \bigcup_{i=0}^{\mathcal{L}} \mathcal{V}_i$. We define a virtual root node $\mathcal{D}_{0,0} \in \mathcal{V}_0$, whose child nodes correspond to targeted domain (e.g., mathematics, physics). To support continual expansion, each layer contains discovered topic nodes and an additional unknown placeholder node:

$$\mathcal{V}_i = \{\mathcal{D}_{i,1}, \ldots, \mathcal{D}_{i,N_i}\} \cup \{\mathcal{D}_{i,\text{unk}}\}. \tag{4}$$

The edge set encodes parent–child relations across layers:

$$\mathcal{E} = \{(\mathcal{D}_{i,j}, \mathcal{D}_{i+1,k}) \mid \mathcal{D}_{i+1,k} \text{ is a child of } \mathcal{D}_{i,j}, \; i \in \{0, \ldots, \mathcal{L}-1\}\}. \tag{5}$$

A root-to-leaf path is

$$\mathcal{P} = [\mathcal{D}_{0,0} \to \mathcal{D}_{1,i_1} \to \cdots \to \mathcal{D}_{\mathcal{L},i_\mathcal{L}}], \tag{6}$$

where the leaf node $\mathcal{D}_{\mathcal{L},i_\mathcal{L}}$ is treated as a minimal knowledge unit that triggers subsequent web retrieval and corpus sampling.

Node-wise learnability posterior and path sampling. The tree structure alone does not specify *where* to explore next. Inspired by multi-armed bandits (Slivkins et al., 2019; Gao et al., 2025), we maintain a Beta posterior for each node $\mathcal{D}_{i,j}$, which will be initialized to $(1, 1)$.

$$\mathcal{M}(\mathcal{D}_{i,j}) = \text{Beta}(\alpha_{i,j}, \beta_{i,j}) \tag{7}$$

where $\mathcal{M}(\mathcal{D}_{i,j})$ represents the node-wise learnability, i.e. the probability that sampling the subtree rooted at $\mathcal{D}_{i,j}$ yields a *learnable* training instance. We use Thompson sampling for path selection. At each depth, we sample a score from each candidate child's Beta posterior and choose the branch with the highest sampled score, which naturally trades off exploiting high-learnability regions and exploring uncertain ones.

Unk-triggered expansion. Since the fine-grained decomposition of a domain cannot be exhaustively enumerated in advance, we trigger incremental growth when sampling selects the unk node $\mathcal{D}_{i+1,\text{unk}}$ under a parent node $\mathcal{D}_{i,j}$. Specifically, we generate a new sibling subtopic by

$$\mathcal{D}_{i+1,\text{new}} = \text{Expand}\big(\mathcal{D}_{i,j}, \; \Psi(\mathcal{D}_{i,j})\big), \tag{8}$$

where $\Psi(\mathcal{D}_{i,j}) = \{\mathcal{D}_{i+1,1}, \ldots, \mathcal{D}_{i+1,N_{i+1}}\}$ denotes the set of existing children of $\mathcal{D}_{i,j}$ and serves as same-level context to encourage complementary, non-duplicate labels. We enforce deduplication by filtering $\mathcal{D}_{i+1,\text{new}} \notin \Psi(\mathcal{D}_{i,j})$.

Web-backed validation for non-leaf nodes. To suppress hallucinated concepts and semantic drift, we validate non-leaf expansions ($i + 1 < \mathcal{L}$) using an external function $V_{\text{wiki}}(\cdot)$. Given $\mathcal{D}_{i+1,\text{new}}$, the validator returns a set of retrieved Wikipedia titles $\hat{W}$ and computes the maximum string-match similarity:

$$s_{\max} = \max_{w \in \hat{W}} \text{sim}_{\text{str}}\big(\mathcal{D}_{i+1,\text{new}}, \; w\big). \tag{9}$$

If $s_{\max} \geq \tau_{\text{wiki}}$, we add $\mathcal{D}_{i+1,\text{new}}$ to the tree and create its next-layer unknown placeholder $\mathcal{D}_{i+2,\text{unk}}$ to enable further growth; otherwise, we reject the expansion and re-sample to prevent early erroneous concept pollution of the tree.

3.3 Open-Web Retrieval, Filtering, and Corpus Construction

While the tree $\mathcal{T}$ provides a structured concept space, turning it into a trainable environment requires continuously attaching path-consistent external corpus to leaf concepts. Therefore, we maintain an external corpus pool for each leaf node $\mathcal{D}_{\mathcal{L},k}$.

$$\mathcal{C}(\mathcal{D}_{\mathcal{L},k}) = \Big\{ F_{\text{clean}}(p) \;\Big|\; p \in \mathcal{U}_{\mathcal{L},k}, \; F_{\text{url}}(p) = 1, \; \cos\big(\phi(\mathcal{D}_{\mathcal{L},k}), \phi(H_p)\big) \geq \tau_{\text{emb}} \Big\} \tag{10}$$

where $\mathcal{U}_{\mathcal{L},k}$ is the set of URLs retrieved by querying the web with the leaf node $\mathcal{D}_{\mathcal{L},k}$ as the search keyword, H_p is the page title, $\phi(\cdot)$ is a semantic encoder, F_{url} applies URL allow/deny lists to reduce noise and mitigate benchmark leakage, and F_{clean} removes boilerplate such as navigation bars, ads, templates, and duplicate blocks. This procedure converts the open-web into a leaf-aligned, continuously refreshed corpus environment for self-play training.

3.4 Web-grounded Two Roles Self-Play Training

At each iteration, we sample B paths $\{\mathcal{P}_i\}_{i=1}^{B}$ from the tree $\mathcal{T}$. For each path $\mathcal{P}_i$, we sample a document d from the corresponding leaf corpus pool $\mathcal{C}(\mathcal{D}_{\mathcal{L},i})$, and use a single policy π_θ to play both roles: **Challenger** generates QA pairs conditioned on the visible document d, and **Solver** answers the questions generated by the Challenger without access to d. Training signals come from verifiable rewards, and policy updates are performed with Dr. GRPO.

Challenger: QA generation with verifiability filtering. For each document d, Challenger proposes G candidate QA pairs $\{(q_i, y_i^*)\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid d)$. We then apply a rule-based validator $\Gamma(\cdot)$ to filter out unverifiable or malformed instances, yielding

$$\mathcal{Q}(d) = \{(q_i, y_i^*) \mid \Gamma(q_i, y_i^*) = 1\}. \tag{11}$$

Each invalid QA will receive a penalty reward, discouraging malformed or unverifiable generations.

Solver: answer generation and solvability estimation. For a valid QA $(q_i, y_i^*) \in \mathcal{Q}(d)$, Solver answers the question q_i by sampling K independent responses: $\{\hat{y}_{i,k}\}_{k=1}^{K} \sim \pi_\theta(\cdot \mid q_i)$ and uses Math-Verify $v(\cdot,\cdot)$ to obtain correctness indicators $\ell_{i,k} = v(\hat{y}_{i,k}, y_i^*) \in \{0,1\}$. We summarize solvability by the empirical accuracy and variance:

$$\hat{p}_i = \frac{1}{K}\sum_{k=1}^{K} \ell_{i,k},\ \widehat{\mathrm{Var}}_i = \hat{p}_i(1-\hat{p}_i). \tag{12}$$

If all valid QA pairs satisfy $\widehat{\mathrm{Var}}_i = 0$, then the document-level training signal is typically too easy, too hard, or unreliable. We skip policy updates for this document.

Reward design and Dr. GRPO updates. For a valid QA (q_i, y_i^*), Solver will receive verifiable correctness rewards for each response:

$$r_{i,k}^{\mathrm{sol}} = \ell_{i,k} = v(\hat{y}_{i,k}, y_i^*), k = 1, \ldots, K \tag{13}$$

We assign a difficulty-shaped reward based on Solver's variance $\widehat{\mathrm{Var}}_i$, which peaks at moderate difficulty ($\hat{p}_i = 0.5$, i.e., $\widehat{\mathrm{Var}}_i = 0.25$) and decreases toward the extremes:

$$r_i^{\mathrm{cha}} = \begin{cases} \exp\left(-\frac{(\widehat{\mathrm{Var}}_i - 0.25)^2}{\sigma}\right), & (q_i, y_i^*) \in \mathcal{Q}(d), \\ \rho, & \text{otherwise}, \end{cases} \tag{14}$$

where σ controls the width of the medium-difficulty band and $\rho < 0$ penalizes invalid QA.

Role balancing. To keep the amount of training data aligned across the two roles, we uniformly sample one QA from the valid set, $(q^*, y^*) \sim \mathrm{Unif}(\mathcal{Q}(d))$, and use only this QA to construct Solver's grouped trajectories (i.e., K Solver responses and their rewards). Challenger, in contrast, uses all G QA pairs (valid with shaped rewards and invalid with punishment ρ) as its grouped samples.

3.5 Posterior-guided Tree Updating with Sliding Window

We convert self-play outcomes into feedback on the domain tree, closing the exploration–learning loop. For each valid QA of the sampled path $\mathcal{P}$, we define a Bernoulli learnability observation

$$g \triangleq \mathbf{1}\left(\widehat{\mathrm{Var}} \geq \tau_{\mathrm{var}}\right), \tag{15}$$

where $\tau_{\mathrm{var}} \in (0, 0.25]$ controls the width of the band around the capability boundary. Intuitively, $g = 1$ indicates that the QA is likely near the current boundary and thus training-effective, whereas $g = 0$ suggests that it is too easy, too hard, or unreliable. We attribute this feedback to all nodes on the $\mathcal{P}$ and perform Beta–Bernoulli conjugate updates:

$$\begin{cases} \alpha_{i,j} \leftarrow \alpha_{i,j} + g, \\ \beta_{i,j} \leftarrow \beta_{i,j} + (1-g), \end{cases} \forall\, \mathcal{D}_{i,j} \in \mathcal{P}. \tag{16}$$

Because learnability is non-stationary as the policy improves, accumulating statistics over the full history can bias exploration toward early observations. To mitigate this effect, we use a sliding window of the most recent μ observations per node to form effective parameters for sampling:

$$\begin{cases} \tilde{\alpha}_{i,j} = 1 + \sum_{\tau \in \mathcal{W}_{i,j}} g^{(\tau)}, \\ \tilde{\beta}_{i,j} = 1 + \sum_{\tau \in \mathcal{W}_{i,j}} \left(1 - g^{(\tau)}\right). \end{cases} \tag{17}$$

where μ is window size, $\mathcal{W}_{i,j}$ denotes the indices of the most recent μ updates of $D_{i,j}$. During path sampling, we run Thompson sampling with $\mathrm{Beta}(\tilde{\alpha}_{i,j}, \tilde{\beta}_{i,j})$, so that exploration preferences reflect the learnability distribution at the *current* capability stage, improving both adaptivity and exploration efficiency.

4 Experiments

4.1 Setup

Models and baseline. Following R-Zero and SPICE, we evaluate WIST on two model families and scales: *Qwen3-4B-Base*/*Qwen3-8B-Base* (Yang et al., 2025) and *OctoThinker-3B*/*OctoThinker-8B* (Wang et al., 2025). We compare WIST against the following baselines: (1) **Base Model**: the pretrained model without any post-training, serving as the performance floor; (2) **R-Zero**: a fully endogenous self-play method that relies only on prompting and self-generated problems, without accessing external data; (3) **SPICE**: a corpus-grounded self-play method that uses a curated high-quality pretraining corpus (Nemotron-CC-Math (Mahabadi et al., 2025)) as the environment. All baseline implementations are provided in the Appendix D.

Evaluation Benchmarks. We evaluate WIST on a broad suite of math and general reasoning benchmarks, largely following the setups in

Table 1: **Main results** on mathematical and general reasoning benchmarks across four backbones. Best and second-best results within each backbone block are marked in **bold** and underline, respectively.

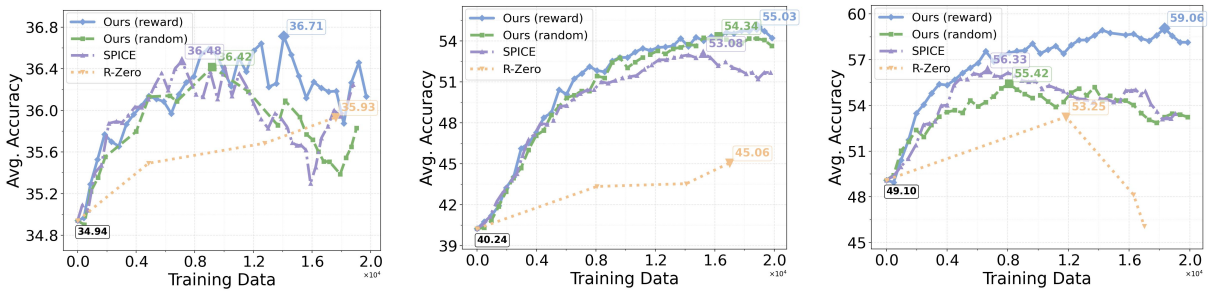
Method	Mathematical Reasoning							General Reasoning				Overall
	AMC	Minerva	MATH 500	GSM8K	Olymp.	AIME 24	AIME 25	Super-GPQA	GPQA-Diamond	MMLU-Pro	BBEH	
Qwen3-4B-Base												
Base Model	41.4	35.7	57.0	75.9	30.2	9.5	6.4	18.0	32.8	51.5	8.2	33.3
+ R-Zero	53.5	44.1	77.0	91.1	39.5	10.3	7.1	26.7	33.4	53.7	10.4	40.6
+ SPICE	50.1	**47.8**	76.2	92.5	**41.0**	**12.0**	**10.9**	27.8	35.1	54.3	**11.8**	41.8
+ WIST (ours)	**60.0**	**47.8**	**78.2**	**92.9**	40.0	11.6	9.7	**29.6**	**37.2**	**55.7**	**11.8**	**43.1**
Qwen3-8B-Base												
Base Model	57.2	43.0	73.0	91.3	40.5	11.7	11.3	28.3	34.8	58.2	9.1	42.1
+ R-Zero	61.1	48.5	80.4	92.9	45.2	14.0	12.8	31.8	**42.4**	60.4	11.2	45.5
+ SPICE	60.1	51.5	81.8	**93.9**	**45.3**	**15.4**	13.4	31.3	40.9	60.8	11.6	46.0
+ WIST (ours)	**63.4**	**53.3**	**82.6**	93.4	44.1	14.8	**13.9**	**32.5**	41.4	**61.1**	**12.9**	**46.7**
OctoThinker-3B-Hybrid-Base												
Base Model	12.5	18.5	30.6	44.9	11.0	1.7	0.6	10.4	2.0	11.1	2.3	13.2
+ R-Zero	26.2	21.8	50.4	73.5	17.2	1.8	0.4	12.6	20.9	18.7	4.4	22.5
+ SPICE	**28.3**	22.4	**50.8**	**76.7**	**17.3**	**2.7**	**0.8**	**18.4**	23.7	**31.7**	**4.8**	**25.2**
+ WIST (ours)	27.4	**22.7**	48.8	76.3	15.1	1.9	0.6	17.8	**24.1**	30.4	4.1	24.5
OctoThinker-8B-Hybrid-Base												
Base Model	20.0	26.2	42.8	82.2	17.0	2.4	1.1	16.4	12.1	25.9	5.4	22.9
+ R-Zero	25.2	31.5	58.7	86.3	**25.9**	3.5	**1.5**	24.1	27.3	42.5	9.9	30.6
+ SPICE	**33.8**	30.2	58.6	**87.6**	24.9	**4.8**	0.9	23.3	**30.8**	40.5	**10.4**	31.4
+ WIST (ours)	31.0	**36.4**	**62.0**	87.0	25.5	3.2	1.4	**25.4**	30.6	**45.9**	10.1	**32.6**

R-Zero and SPICE, and additionally include a physics benchmark to test domain-specific gains. (1) **Mathematical reasoning.** We report results on AMC, Minerva (Lewkowycz et al., 2022), MATH-500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), OlympiadBench (He et al., 2024), AIME'24, and AIME'25. We report accuracy based on greedy decoding for most evaluations, following (Ma et al., 2025). The only exceptions are AIME'24 and AIME'25, where scores are averaged across 32 sampling runs as in (Zeng et al., 2025). (2) **General-domain reasoning.** To measure generalization beyond math, we evaluate on MMLU-Pro (Wang et al., 2024), SuperGPQA (Du et al., 2025), GPQA-Diamond (Rein et al., 2024), and BBEH (Kazemi et al., 2025), following the prompts and evaluation code from (Ma et al., 2025) and reporting accuracy under greedy decoding. Detailed evaluation settings are provided in Appendix E.

Training Details. Our entire framework is implemented based on the OpenRLHF (Hu et al., 2024) and set the target domain to Mathematics, consistent with prior self-evolution studies such as R-Zero and SPICE. In each iteration, we sample $B = 128$ root-to-leaf paths from the domain tree with maximum depth $\mathcal{L} = 4$. We define the learnability event using self-consistency variance and use a threshold $\tau_{\text{var}} = 0.2$, which is consistent with the range used in prior work (Zhang et al., 2025; Huang et al., 2025; Bercovich et al., 2025), with a sliding window of size $\mu = 5$ for posterior updates. For open-web corpus acquisition, we filter retrieved pages by title semantic similarity with threshold $\tau_{\text{emb}} = 0.5$, and truncate each cleaned document to at most 5992 tokens. For self-play, Challenger and Solver both repeat sampling G=8 times; invalid QA candidates receive a fixed penalty $\rho = -0.1$. We optimize the policy with Dr.GRPO using training batch size $B_{\text{train}} = 512$. All other hyperparameters and implementation details are provided in Appendix D.

4.2 Main Results

As shown in Table 1, WIST consistently outperforms the base models across all four backbones and achieves the best overall performance in three of them. Specifically, on *Qwen3-4B-Base*, WIST improves the Overall score from 33.3 to 43.1 (+9.8), surpassing R-Zero (40.6, +7.3) and SPICE (41.8, +8.5). This indicates that, even without relying on a carefully curated corpus, WIST can continuously produce effective training signals through structured exploration and web-grounded corpus construction. On *Qwen3-8B-Base*, WIST again attains the highest Overall score (46.7), achieving a +4.6 gain over the base model and outperforming both



(a) Qwen3-4B-Base (b) Qwen3-8B-Base (c) Qwen3-14B-Base

Figure 3: Training performance of our method **WIST** in the medical domain using three Qwen3 models of different sizes.

R-Zero (45.5) and SPICE (46.0). This suggests that as model capacity increases, WIST's exploration–retrieval–self-play loop translates more reliably into cross-task generalization gains. Moreover, on *OctoThinker-8B-Hybrid-Base*, WIST raises the Overall score to 32.6 (+9.7), exceeding R-Zero (30.6) and SPICE (31.4). These results corroborate that when the underlying model has sufficient reasoning and information-integration capability, WIST's tree-structured decomposition and posterior-guided exploration can more effectively identify weaknesses, broaden long-tail concept coverage, and yield cumulative improvements on both mathematical and general reasoning benchmarks. Additional results on a larger-scale model (*Qwen3-14B-Base*) are provided in Appendix F.1.

In contrast, under *OctoThinker-3B-Hybrid-Base*, WIST achieves an Overall score of 24.5, slightly below SPICE's 25.2, while still substantially outperforming R-Zero (22.5) and the base model (13.2). This outcome is expected: WIST relies on open-web retrieval and automatic cleaning to construct its corpus environment. Although relevance filtering and controllability constraints reduce noise, the open web inevitably contains noisy, ambiguous, or weakly related content. For a smaller 3B model, such noise can more easily amplify misleading gradients and distributional drift during self-play, thereby undermining training stability. By contrast, SPICE operates in a carefully curated in-corpus environment with more controlled data quality and distribution, which mitigates the adverse impact of noise in the small-model regime. Overall, the results indicate that WIST better realizes the benefits of open-web corpus and tree-guided curricula for medium-to-large models, while in the small-model setting, the interaction between environmental noise and limited model capacity becomes a key factor shaping the attainable gains.

5 Ablation Studies

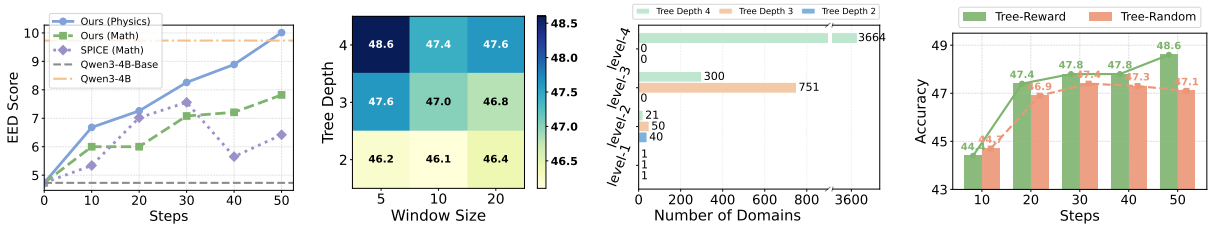
5.1 Domain Transfer Beyond Mathematics

To test whether WIST can be steered beyond mathematics, we study transfer to two scientific domains, **medicine** and **physics**. We report broader experiments in medicine on three Qwen3 base models, and a focused study on *Qwen3-4B-Base* in physics. This difference mainly reflects the SPICE setup: high-quality curated corpora are relatively abundant in medicine, but much more limited in physics.

Transfer to medicine. We steer the target domain to **medicine** and evaluate WIST on *Qwen3-4B-Base*, *Qwen3-8B-Base*, and *Qwen3-14B-Base*. For comparison, we use a PubMed-based medical corpus from prior work (Kandpal et al., 2025) as the domain corpus for SPICE. We use OpenCompass (Contributors, 2023) for evaluation and report the average accuracy on three medical benchmarks:

- **Medbullets** (Chen et al., 2025a): a clinically oriented benchmark with USMLE-style questions and expert explanations, testing grounded medical reasoning and clinical decision-making.
- **MedMCQA** (Pal et al., 2022): a large-scale multiple-choice benchmark built from real-world medical entrance examination questions.
- **MedQA** (Jin et al., 2020): an exam-based multiple-choice benchmark collected from multiple regions and accompanied by medical textbooks, suitable for evaluating professional medical knowledge and reasoning.

As shown in Figure 3, WIST consistently improves medical-domain performance across all three model sizes and outperforms both SPICE



(a) PhyBench EED scores over training steps.

(b) Math average performance across tree-depth and window-size settings.

(c) Number of nodes under different tree depths.

(d) Random vs. Reward-Guided Exploration: Math average performance over Steps.

Figure 4: Ablation results of WIST, including domain transfer, hyperparameter sensitivity, tree scaling, and exploration strategy.

and R-Zero. It raises average accuracy by **+1.77** on *Qwen3-4B-Base*, **+14.79** on *Qwen3-8B-Base*, and **+9.96** on *Qwen3-14B-Base*. We also observe larger gains on larger models: compared with the modest improvement on 4B, both 8B and 14B benefit substantially more, suggesting that higher-capacity models can make better use of retrieved medical evidence and convert it into stronger reasoning gains. The detailed results of the best checkpoints for each model are provided in the appendix F.3.

Transfer to physics. ACLWe also steer the target domain to physics and evaluate WIST on *Qwen3-4B-Base*. We still use OpenCompass (Contributors, 2023) for evaluation on **PhyBench** (Qiu et al., 2025), a physics reasoning benchmark that requires models to generate structured LaTeX expressions. We report the **EED Score** (Expression Edit Distance Score; range 0–100), which compares predicted and reference expressions through expression-tree alignment and captures structural and semantic partial correctness. For stability, each evaluation is repeated 10 times and we report the mean score. In this setting, because a comparable high-quality curated **physics corpus** is not readily available, SPICE still uses its mathematics corpus.

As shown in Figure 4a, WIST yields clear gains on PhyBench. For example, *Qwen3-4B-Base* improves from 4.73 to 10.01 after 50 training steps, outperforming both the base model and a stronger reasoning-enhanced baseline (*Qwen3-4B* in thinking mode). This indicates that WIST can acquire domain-relevant corpus from the open web and translate it into measurable gains in physics reasoning and symbolic expression generation.

5.2 Coupled Sensitivity of Tree Depth and Window Size

We next examine the interaction between two key hyperparameters, tree depth ($\mathcal{L}$) and window size (μ). Intuitively, $\mathcal{L}$ controls the granularity and branching of the exploration space, while μ controls how quickly posterior guidance adapts to policy non-stationarity. Rather than tuning them independently, we perform a grid sweep over $\mathcal{L} \in \{2, 3, 4\}$ and $\mu \in \{5, 10, 20\}$, and evaluate the average performance of mathematics after training for 50 steps on the Qwen3-4B-Base model. Results are summarized in the heatmap in Figure 4b.

We observe a clear coupling effect: shallower trees favor larger windows, whereas deeper trees favor smaller windows. This trend is consistent with how the effective search space scales with depth. As $\mathcal{L}$ increases, the number of reachable leaf concepts grows rapidly, increasing path diversity and reducing the revisit frequency of any single path. As shown in Figure 4c, the deeper the tree's depth, the more nodes explored, and the larger the search space. Under such high diversity, a large window may aggregate stale or heterogeneous feedback and introduce noise, while a smaller window better tracks local, recent learnability. Conversely, when $\mathcal{L}$ is small, paths are revisited more frequently, and larger windows provide more stable statistics for posterior estimation.

Notably, setting the tree depth to 2 is effectively equivalent to directly prompting the model to generate minimal knowledge points within the target domain. As shown in Figure 4c, this configuration yields only about 40 leaf-level concepts at the second layer. Meanwhile, Figure 4b indicates that without deeper, structured tree guidance, it is difficult for the model to reliably retrieve and organize high-value, domain-relevant knowledge from the vast and noisy open web. These results provide empirical corpus for the necessity and critical role of the proposed Tree component in open-web self-evolution.

5.3 Benefit of Reward-Guided Tree Expansion

We isolate the contribution of **reward-guided exploration** in tree construction. In this experiment, we train *Qwen3-4B-Base* for 50 steps in the **mathematics** domain and compare two variants for selecting expansion paths and sampling branches: (1) **Random exploration**, where candidate nodes are selected uniformly at random without using any feedback signal; and (2) **Reward exploration**, where sampling is driven by the posterior updated from learnability feedback. As shown in Figure 4d, which reports the average performance on the math benchmark over training steps, the two variants perform similarly at early stages but diverge as training proceeds. Reward-guided exploration yields a more stable improvement trajectory and achieves the best final performance at 50 steps, whereas random exploration remains consistently weaker. The detailed results at step 50 are provided in Appendix F.2. This suggests that unguided expansion tends to disperse effort across the enlarged search space and fails to consistently focus on high-yield regions. The same pattern is observed in the medical-domain experiments in Section 5.1, where reward exploration consistently outperforms random exploration across different model sizes, as shown in Figure 3. Together, these results indicate that the effectiveness of WIST comes not merely from broader exploration, but from using learnability signals to direct exploration toward high-yield regions of the domain tree.

6 Conclusion

We proposed **WIST**, a web-grounded iterative self-play tree framework for domain-targeted reasoning improvement. WIST closes the loop between structured exploration and learning by expanding a domain tree, retrieving path-consistent web corpus, training a Challenger–Solver self-play process with verifiable rewards, and feeding learnability signals back to guide future exploration. Across diverse backbones, WIST consistently improves over the base models and typically outperforms both purely endogenous self-evolution (R-Zero) and corpus-grounded self-play (SPICE). WIST is also inherently domain-steerable: by switching only the target domain label, it achieves strong improvements in both **medicine** and **physics**, demonstrating effective transfer beyond mathematics. Ablations further confirm the effectiveness of WIST's key components for stable open-web learning. Overall, WIST combines the flexibility and coverage of open-web grounding with structured, feedback-guided exploration, providing a practical, scalable, and domain-adaptive route to self-improvement without relying on manually curated domain corpora.

Limitations

A key limitation of the current WIST instantiation lies less in domain coverage itself than in verifier availability. WIST can transfer beyond mathematics when retrieved data can support automatically checkable self-play instances, as shown in our physics and medical experiments. However, extending this framework to open-ended expert tasks with inherently ambiguous or non-unique answers will require stronger verification mechanisms or more reliable reward modeling. In addition, WIST also relies on open-web retrieval, where corpus quality and alignment can vary despite filtering and cleaning. This variability can attenuate gains for smaller models, but we still observe consistent improvements overall.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant No. 2023YFC3305102), the National Natural Science Foundation of China (Grant No. 6250076080), and the China Postdoctoral Science Foundation (Grant No. 2025M771537). We also acknowledge support from the Shanghai Innovation Institute and the Shanghai Artificial Intelligence Laboratory.

References

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mo jtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sandra Mitts, Adithya Renduchintala, Stephen Roller, and 7 others. 2022. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378:1067 – 1074.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, and 1 others. 2025. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*.

Maosong Cao, Taolin Zhang, Mo Li, Chuyu Zhang, Yunxin Liu, Haodong Duan, Songyang Zhang, and Kai Chen. 2025. Condor: Enhance llm alignment with knowledge-driven data synthesis and refinement. *arXiv preprint arXiv:2501.12273*.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025a. Benchmarking large language models on answering and explaining challenging medical questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3563–3599.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. 2025b. Self-questioning language models. *arXiv preprint arXiv:2508.03682*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Pengyu Cheng, Yong Dai, Tianhao Hu, Han Xu, Zhisong Zhang, Lei Han, Nan Du, and Xiaolong Li. 2024. Self-playing adversarial language game enhances llm reasoning. *Advances in Neural Information Processing Systems*, 37:126515–126543.

Jeff Clune. 2019. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, and 1 others. 2025. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*.

Junqi Gao, Zhichang Guo, Dazhi Zhang, Dong Li, Runze Liu, Pengfei Li, Kai Tian, and Biqing Qi. 2025. Bohdi: Heterogeneous llm fusion with automatic data exploration. *arXiv preprint arXiv:2506.15721*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Laura Harding Graesser, Kyunghyun Cho, and Douwe Kiela. 2019. Emergent linguistic phenomena in multi-agent communication games. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3700–3710, Hong Kong, China. Association for Computational Linguistics.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. 2024. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *arXiv preprint arXiv:2009.13081*.

Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A Feder Cooper, Aviya Skowron, and 1 others. 2025. The common pile v0. 1: An 8tb dataset of public domain and openly licensed text. *arXiv preprint arXiv:2506.05209*.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, and 1 others. 2025. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26473–26501.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo

Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. 2025a. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. 2025. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*.

Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-cc-math: A 133 billion-token-scale high quality math pretraining dataset. *arXiv preprint arXiv:2508.15096*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. 2025. Self-improving language models for evolutionary program synthesis: A case study on arc-agi. *arXiv preprint arXiv:2507.14172*.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, and 1 others. 2025. Phybench: Holistic evaluation of physical perception and reasoning in large language models. *arXiv preprint arXiv:2504.16074*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Bidipta Sarkar, Warren Xia, C. Karen Liu, and Dorsa Sadigh. 2025. Training language models for social deduction with multi-agent reinforcement learning. In *Adaptive Agents and Multi-Agent Systems*.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. 2025. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*.

Aleksandrs Slivkins and 1 others. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. 2025. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*.

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Ilia Kulikov, Kyunghyun Cho, Dong Wang, Yuandong Tian, Jason E Weston, and 1 others. 2025. Naturalreasoning: Reasoning in the wild with 2.8 m challenging questions. *arXiv preprint arXiv:2502.13124*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. 2025. Consistent paths lead to truth: Self-rewarding reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.08745*.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025a. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. 2025b. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*.

A Algorithm Implementation

We presented the overall framework of WIST in Algorithm 1.

B Related Work

Reasoning-focused RL and verifiable supervision. Reinforcement learning has been widely used to align LLMs with human preferences (Jaques et al., 2019; Ouyang et al., 2022), and more recently to directly enhance reasoning by optimizing rewards that can be checked automatically. RL with verifiable rewards (RLVR) has proven effective in domains with crisp correctness signals (e.g., math or executable programs), where rule-based or deterministic verification can replace expensive human judgments (Uesato et al., 2022; Lightman et al., 2023). A practical limitation is that many RLVR pipelines still depend on pre-collected tasks, which restrict coverage and domain adaptation. *Our work complements RLVR by turning the open web into a continuously refreshable source of verifiable training instances, enabling domain-targeted improvement without relying on a pre-arranged domain corpus.*

Self-play and automatic curricula for language models. Self-play is a general mechanism for creating training curricula through interaction, and has long been central to game-playing systems (Lightman et al., 2023; Bakhtin et al., 2022). In language modeling, self-play has been explored both for alignment (e.g., models critiquing or rewarding their own outputs) and for capability gains via dual-role or adversarial setups (Chen et al., 2024; Yuan et al., 2024; Cheng et al., 2024; Liu et al., 2025a; Huang et al., 2025). However, applying multi-agent RL to full LLMs often requires simplifying assumptions or bespoke environments, and performance can hinge on how well the interaction setting controls task difficulty and data quality (Harding Graesser et al., 2019; Sarkar et al., 2025). *Like SPICE (Liu et al., 2025a), WIST falls under corpus-grounded self-play, but differs in both the environment and the training process. Rather than assuming a carefully curated domain corpus, WIST treats the open web as the self-play environment and organizes it with a dynamically expanding domain tree. Starting from only a user-specified target domain, WIST autonomously expands subtopics, retrieves path-consistent web evidence, and uses posterior-guided sampling to induce a learnability-aware curriculum during self-play.*

Endogenous self-evolution and label-free reward signals. A growing body of work seeks to reduce reliance on labeled data by deriving rewards from the model itself, such as confidence, entropy, or agreement across multiple samples (Ouyang et al., 2022; Chen et al., 2025b). These signals are often coupled with self-training loops that iteratively fine-tune on the model's own solutions (Zhao et al., 2025b; Shafayat et al., 2025). Fully endogenous variants can even generate problems from scratch, but they may suffer from drift as errors compound and the generated distribution departs from grounded knowledge. *Our approach reduces this failure mode by grounding the self-play loop in retrieved corpus and by constraining exploration through a structured decomposition of the target domain, which helps maintain coverage and training stability over iterations.*

Web/corpus mining and synthetic QA generation. Another line of research scales reasoning data by mining questions from documents or generating synthetic QA from prompts, either bootstrapping from existing datasets or harvesting from large corpora and the web (Wang et al., 2023; Ze-

```
Root    (virtual root at level 0)
  |- Mathematics    (target domain at level 1)
     |- unk              (placeholder for expansion at level 2)
     |- Calculus         (sub-domain at level 2)
         |- unk                        (placeholder for expansion at level 3)
         |- Differential Equations     (sub-domain at level 3)
             |- unk                                  (placeholder for expansion at level 4)
             |- Method of Undetermined Coefficients  (minimal knowledge point at level 4)
             |- Method of Variation of Parameters    (minimal knowledge point at level 4)
```

Figure 5: **Example of a partial mathematics domain label tree generated by WIST.** Each non-leaf level contains an unk child that indicates a possible expansion point.

likman et al., 2022; Yu et al., 2023; Mahabadi et al., 2025; Yuan et al., 2025). Most of these pipelines are offline: they produce static datasets whose distribution is fixed once collected, and they typically require extensive filtering rules to ensure quality. More interactive approaches generate questions online from document contexts to better match the learner's current capability, but often assume a curated corpus as the environment. *WIST is distinct in that it does not require a fixed in-corpus environment: it continuously acquires corpus from the open web and couples retrieval with posterior-guided exploration over a dynamically expanded domain tree, enabling domain-steerable self-evolution with minimal manual data curation.*

C Example of a Domain Label Tree

To improve readability, we provide here a concrete example of a partial domain label tree generated by WIST. Starting from a user-provided target domain label, WIST incrementally expands the tree into finer-grained subtopics, while reserving an unk child at each non-leaf level to indicate a possible expansion point.

Figure 5 shows a small real snapshot of a domain tree rooted at **Mathematics**. First, path sampling selects a root-to-leaf route, for example, *Mathematics* $\rightarrow$ *Calculus* $\rightarrow$ *Differential Equations* $\rightarrow$ *Method of Undetermined Coefficients*. Second, if sampling selects an unk node, WIST expands the tree by generating a new sibling label under the corresponding parent node. Third, after self-play on documents associated with the selected leaf, the resulting learnability signal is used to update the node statistics along the sampled path, which in turn influences future path sampling.

D Training Details

For R-Zero, we run the official released implementation with the same backbone models and compute budget whenever applicable. Since SPICE has not publicly released its code, we have reproduced the baseline based on the algorithmic process described in its paper. To reduce confounding factors, we align the optimizer, sampling strategy, and training steps with those used by WIST whenever possible, and we report ablations to quantify the effect of each additional component beyond the shared training recipe.

Baseline hyperparameters. All the overall hyperparameters of the comparison methods are shown in the Table 2.

WIST hyperparameters. Table 3 lists the main hyperparameters used in our WIST training. Unless otherwise specified, we set the target domain to Mathematics. In addition, we use a Sentence Embedding model, namely all-MiniLM-L6-v2 (Reimers and Gurevych, 2019), as our semantic encoder $\phi(\cdot)$.

E Evaluation Settings

Evaluation Protocol. We evaluate all models in a *zero-shot* setting to examine whether the reasoning abilities acquired through corpus-grounded self-play generalize to standard benchmarks without any task-specific adaptation. For most benchmarks, we use *greedy decoding* (temperature $= 0$) to maximize reproducibility and ensure consistent comparisons across models.

Mathematical Reasoning. For AIME'24 and AIME'25, we adopt a sampling-based protocol to better capture performance on challenging competition-style problems: we run 32 independent generations with temperature $= 0.6$ and report the *average accuracy*. For the remaining mathematical reasoning benchmarks, including MATH-500, OLYMPIADBENCH, MINERVA MATH, GSM8K, and AMC, we report *pass@1* under greedy decoding. Predictions are scored by *exact match* after answer extraction and normalization. To reduce false negatives caused by formatting differences,

Table 2: Training configurations for all compared methods.

Configuration	WIST	SPICE	R-Zero
Data Source			
Corpus documents	$\sim 15,000$	20,000	–
Question source	Web-grounded	Document-grounded	Self-generated
External grounding	✓	✓	×
Training Details			
Challenger training	✓	✓	✓
Challenger sampling	8	8	4
Reasoner training	✓	✓	✓
Reasoner sampling	8	8	5
Temperature	1.0	1.0	1.0
Optimizer	DrGRPO	DrGRPO	GRPO
Reward Design			
Challenger reward	Gaussian Variance (max 1.0)	Gaussian Variance (max 1.0)	$1 - 2\|p - 0.5\|$
Reasoner reward	Binary correctness	Binary correctness	Binary correctness
Invalid penalty	-0.1	-0.1	-1 (Challenger)
Performance			
Training iterations	50	50	$3^{\dagger}$

Hyperparameter	Value
Rollout batch size (paths per iteration) B	128
Training batch size B_{train}	512
Tree maximum depth $\mathcal{L}$	4
Beta prior (α_0, β_0)	$(1, 1)$
Learnability threshold (variance) τ_{var}	0.20
Sliding window size μ	5
Title semantic similarity threshold τ_{emb}	0.5
Wiki validation threshold τ_{wiki}	0.8
Challenger QA candidates G	8
Solver self-consistency samples K	8
Rule-validation penalty ρ	-0.1
Challenger shaping width σ	0.02
KL coefficient λ_{KL}	0.0
Learning rate η	1×10^{-6}
Total training updates T_{steps}	50

Table 3: Main hyperparameters for WIST.

we additionally perform equivalence checking via GPT-4.1-2025-04-14 verification.

General Reasoning. For general reasoning, we evaluate on GPQA-DIAMOND, SUPERGPQA, MMLUPRO, and BBEH. All general-reasoning evaluations use greedy decoding and are scored by *exact match* on the extracted multiple-choice option (A/B/C/D). We keep prompts consistent across models by using the same system prompt and answer extraction format as in training. Evaluation prompts instruct models to produce step-by-step reasoning before emitting a final answer, formatted as a boxed result for mathematical tasks or as a letter choice for multiple-choice questions. We will release evaluation prompts and code to support reproducibility.

F Additional Results and Analysis

This section provides complementary results and analyses that further characterize WIST. We first report full-benchmark comparisons between two exploration variants (random vs. reward). We then study cross-model transfer of the learned *domain tree* by constructing the tree with a stronger model and training a smaller model with the transferred tree. Finally, we demonstrate that WIST can be steered to a scientific domain beyond math/physics, namely medicine, and evaluate on three medical QA benchmarks.

F.1 Additional Scaling Results on larger model

In the main paper, WIST is slightly below SPICE in the small-model setting (*OctoThinker-3B-Hybrid-Base*). We attribute this mainly to two factors: (1) smaller models are less reliable in domain-tree expansion and thus more prone to noisy or less informative sub-concepts; and (2) smaller models are more sensitive to residual noise in open-web data, whereas SPICE benefits from a cleaner curated environment.

To further examine the effect of model scale, we additionally evaluate WIST on `Qwen3-14B-Base`. Table 4 reports the full results on the same set of mathematical and general reasoning benchmarks used in the main paper. We observe that WIST achieves the best overall performance, improving over base model by **+3.2** on the Overall score. This

result supports the view that tree-guided open-web grounding becomes more effective as the capability of the base model increases.

F.2 Random vs. Reward Tree Expansion

Table 5 reports the results of all benchmark tests under the same training budget and evaluation protocol for the two variants of WIST, i.e. random and reward.

F.3 Targeted Domain Steering to Medicine

We report in Table 6 the detailed benchmark results of the best-performing checkpoint obtained during medical-domain training.

F.4 Cross-model Tree Transfer: Strong Builder, Small Learner

A practical advantage of WIST is that the domain tree is an explicit, reusable artifact. We study whether a high-quality tree built by a stronger model can be transferred to improve the training of a smaller model. Specifically, we use *Qwen3-14B* to expand a mathematics domain tree (including leaf-level atomic knowledge points) under the same web filtering and validation rules, and then freeze this tree for subsequent training of *Qwen3-4B-Base*. We compare against (1) a tree built by the small model itself (Self-Tree), and (2) a tree build by the strong model (Strong-Tree), all under matched training steps and rollout budget.

From Table 7, Self-Tree and Strong-Tree both substantially improve over the base model, but **Strong-Tree** remains slightly behind **Self-Tree**. This indicates that although a stronger model can construct a reasonable and reusable tree, *tree quality alone* does not necessarily translate into better training outcomes for a weaker learner.

We attribute this behavior to two factors:

- **Learner–curriculum mismatch.**: The tree expanded by *Qwen3-14B* tends to introduce finer-grained and harder leaf concepts, which more often yield extreme self-consistency signals for *Qwen3-4B-Base* and thus produce fewer learnable self-play instances. In contrast, a self-built tree is better aligned with the learner's capability boundary.

- **Posterior dynamics and guidance quality.**: Since WIST relies on posterior-guided exploration, differences in the tree structure can change how quickly and stably node posteriors concentrate, affecting the strength of the guidance signal. Self-Tree typically yields more stable guidance for the learner, resulting in slightly better final performance.

F.5 Open-Web Data Processing and Ablation

Using the open web as a training environment introduces both opportunity and risk: while it provides abundant and continuously refreshed domain knowledge, it also contains noise, redundancy, and potential benchmark contamination. To mitigate these issues, WIST performs a multi-stage data processing pipeline before self-play. Our design is inspired by web-scale cleaning and curation practices in **NVIDIA NeMo Curator**, while being adapted to our tree-guided retrieval setting.

F.5.1 Data Processing Pipeline

Given a sampled leaf concept from the domain tree, WIST retrieves candidate web pages and processes them before adding them to the corresponding corpus pool. The pipeline consists of the following steps:

1. **URL and source filtering.** We first filter retrieved URLs using allow/deny rules to exclude low-quality or undesirable sources. This step is used to reduce obvious noise and lower the risk of benchmark leakage.

2. **Concept–page relevance filtering.** We then measure the semantic relevance between the target concept and the retrieved page (e.g., using the page title and concept representation). Only pages whose relevance exceeds a threshold are kept, ensuring that the resulting corpus remains aligned with the sampled tree node.

3. **Boilerplate removal.** For each retained page, we remove non-content HTML components such as navigation bars, advertisements, templates, and repeated layout blocks. This step extracts the main textual content that is more suitable for subsequent self-play.

4. **Normalization and deduplication.** Finally, we normalize the cleaned text and remove near-duplicate content to reduce redundancy across retrieved pages and improve the stability of the resulting corpus pool.

Together, these steps convert noisy open-web retrieval results into a more controllable and leaf-aligned corpus environment for self-play training.

Table 4: **Additional results on `Qwen3-14B-Base`** across mathematical and general reasoning benchmarks. Best and second-best results within this backbone block are marked in **bold** and underline, respectively.

Method	Mathematical Reasoning							General Reasoning				Overall
	AMC	Minerva	MATH 500	GSM8K	Olymp.	AIME 24	AIME 25	Super-GPQA	GPQA-Diamond	MMLU-Pro	BBEH	
Qwen3-14B-Base												
Base Model	62.5	50.0	80.2	93.7	43.1	13.3	11.4	35.2	40.4	63.9	14.0	46.2
+ R-Zero	60.3	52.9	81.4	94.4	45.2	14.3	11.8	37.1	44.4	65.3	14.9	47.5
+ SPICE	60.2	56.3	**83.8**	94.5	46.7	15.2	15.1	37.1	**45.5**	65.4	15.2	48.6
+ WIST (ours)	**67.5**	**56.6**	83.4	**94.7**	**46.8**	**15.4**	**15.8**	**37.4**	44.4	**66.0**	**15.3**	**49.4**

Table 5: Full-benchmark comparison between random and reward-guided tree expansion under the same training budget.

Variant	Mathematical Reasoning							General Reasoning				Overall
	AMC	Minerva	MATH 500	GSM8K	Olymp.	AIME 24	AIME 25	Super-GPQA	GPQA-Diamond	MMLU-Pro	BBEH	
Qwen3-4B-Base												
Base Model	41.4	35.7	57.0	75.9	30.2	9.5	6.4	18.0	32.8	51.5	8.2	33.3
+ WIST (Random)	51.1	46.0	78.0	92.8	**40.1**	**11.9**	**9.8**	28.0	**37.9**	53.5	11.7	41.9
+ WIST (Reward)	**60.0**	**47.8**	**78.2**	**92.9**	40.0	11.6	9.7	**29.6**	37.2	**55.7**	**11.8**	**43.1**

Table 6: **Medical-domain results** after steering the target domain to medicine. We report benchmark-level accuracy on Medbullets, MedMCQA, and MedQA, together with their average. Best and second-best results within each backbone block can be marked in **bold** and underline, respectively.

Method	Medbullets	MedMCQA	MedQA	Avg.
Qwen3-4B-Base				
Base Model	24.03	32.27	48.52	34.94
+ R-Zero	24.03	32.27	51.50	35.93
+ SPICE	**24.35**	**32.39**	52.69	36.48
+ WIST (Random)	24.19	32.30	52.76	36.42
+ WIST (Reward)	**24.35**	32.37	**53.42**	**36.71**
Qwen3-8B-Base				
Base Model	30.03	39.80	50.87	40.24
+ R-Zero	34.58	46.07	54.54	45.06
+ SPICE	45.94	54.36	**58.94**	53.08
+ WIST (Random)	**50.16**	55.63	57.22	54.34
+ WIST (Reward)	50.00	**56.35**	58.73	**55.03**
Qwen3-14B-Base				
Base Model	38.64	48.51	60.15	49.10
+ R-Zero	46.43	51.83	61.48	53.25
+ SPICE	50.16	56.87	61.96	56.33
+ WIST (Random)	50.00	53.96	62.30	55.42
+ WIST (Reward)	**54.22**	**60.29**	**62.66**	**59.06**

F.5.2 Ablation on Data Processing

To quantify the effect of data processing, we compare our default **full filtering** pipeline against a weaker variant with **HTML-only** cleaning. The latter performs only basic HTML content extraction / cleaning, without the URL/source filtering, relevance filtering, normalization and deduplication.

As shown in Table 8, full filtering and cleaning yields consistent gains across all backbones, improving the mean score by **+0.5** to **+1.7** points over HTML-only cleaning. This indicates that careful web-data curation is important for stable and reliable open-web learning. In particular, basic HTML cleaning alone is not sufficient: URL/source filtering, concept relevance filtering, and deduplication all contribute to building a higher-quality self-play environment.

G Prompt Templates

G.1 Prompt Template for Node Expansion

We present the prompt templates used for expanding the domain tree. Depending on whether the sampled *unk* node appears at a non-leaf level or at the leaf level, the model is prompted to propose either (i) a new *sub-domain* or (ii) a new *atomic knowledge point*. Both templates share a unified "No More" convention and a strict response format to reduce non-standard proposals and stabilize downstream parsing.

Placeholders. *{main_domain}* is the user-specified target domain (e.g., Mathematics, Medicine), *{path_in_str}* is the current root-to-node path, *{parent_name}* is the parent node to be expanded, *{existing_children}* lists existing children under *{parent_name}*, and *{siblings}* optionally lists sibling domains for lightweight anti-misattachment guidance.

Table 7: Tree transfer from a stronger builder model (*Qwen3-14B*) to a smaller learner (*Qwen3-4B-Base*) in mathematics.

	Mathematical Reasoning							General Reasoning				
Variant	**AMC**	**Minerva**	**MATH 500**	**GSM8K**	**Olymp.**	**AIME 24**	**AIME 25**	**Super-GPQA**	**GPQA-Diamond**	**MMLU-Pro**	**BBEH**	**Avg.**
Base Model	41.4	35.7	57.0	75.9	30.2	9.5	6.4	18.0	32.8	51.5	8.2	33.3
+ WIST (Strong-Tree)	53.2	49.3	78.2	92.5	39.7	11.5	9.6	28.5	37.4	55.4	11.7	42.5
+ WIST (Self-Tree)	60.0	47.8	78.2	92.9	40.0	11.6	9.7	29.6	37.2	55.7	11.8	43.1

Table 8: **Ablation on web data processing.** We report the performance over all mathematical and general reasoning benchmarks.

	Mathematical Reasoning							General Reasoning				
Method	**AMC**	**Minerva**	**MATH 500**	**GSM8K**	**Olymp.**	**AIME 24**	**AIME 25**	**Super-GPQA**	**GPQA-Diamond**	**MMLU-Pro**	**BBEH**	**Overall**
Qwen3-4B-Base												
Base Model	41.4	35.7	57.0	75.9	30.2	9.5	6.4	18.0	32.8	51.5	8.2	33.3
+ WIST (HTML-only)	56.6	48.6	77.0	91.8	40.0	11.3	9.5	28.6	31.8	55.1	12.1	42.0
+ WIST (full filtering)	60.0	47.8	78.2	92.9	40.0	11.6	9.7	29.6	37.2	55.7	11.8	43.1
Qwen3-8B-Base												
Base Model	57.2	43.0	73.0	91.3	40.5	11.7	11.3	28.3	34.8	58.2	9.1	42.1
+ WIST (HTML-only)	59.3	51.5	81.6	93.0	44.4	15.3	13.8	31.2	40.9	61.1	13.4	46.0
+ WIST (full filtering)	63.4	53.3	82.6	93.4	44.1	14.8	13.9	32.5	41.4	61.6	12.9	46.7
OctoThinker-3B-Hybrid-Base												
Base Model	12.5	18.5	30.6	44.9	11.0	1.7	0.6	10.4	2.0	11.1	2.3	13.2
+ WIST (HTML-only)	23.6	21.0	49.6	74.4	16.7	1.9	0.6	17.4	23.2	31.7	4.2	24.0
+ WIST (full filtering)	27.4	22.7	48.8	76.3	15.1	1.9	0.6	17.8	24.1	30.4	4.1	24.5
OctoThinker-8B-Hybrid-Base												
Base Model	20.0	26.2	42.8	82.2	17.0	2.4	1.1	16.4	12.1	25.9	5.4	22.9
+ WIST (HTML-only)	30.7	35.3	58.0	85.7	21.9	5.0	1.4	22.7	28.8	40.1	10.2	30.9
+ WIST (full filtering)	31.0	36.4	62.0	87.0	25.5	3.2	1.4	25.4	30.6	45.9	10.1	32.6

Algorithm 1 WIST: Web-grounded Iterative Self-play Tree

Require: Policy π_{θ_0}; target domain $D_{1,1}$; depth $\mathcal{L}$; window μ; iterations T; rollout batch B; group size G

Initialize domain tree $\mathcal{T}$ with virtual root $D_{0,0}$; add $D_{1,1}$ (and its "unk" child) to $\mathcal{T}$.
For each node u, keep Beta posterior (init $(1,1)$) with a sliding buffer of length μ.
for $t \leftarrow 1$ to T **do**
  Part I: Path sampling and tree expansion.
  for $b \leftarrow 1$ to B **do**
    $P_b \leftarrow [D_{0,0}]$; $u \leftarrow D_{0,0}$
    for $i \leftarrow 0$ to $\mathcal{L}-1$ **do**
      Sample $s_c \sim \mathrm{Beta}(\tilde{\alpha}(c), \tilde{\beta}(c))$ for each child $c \in \mathrm{Ch}(u)$; set $c \leftarrow \arg\max s_c$
      while $c = D_{i+1,\mathrm{unk}}$ **do** ▷ Unk-triggered expansion
        $D_{i+1,\mathrm{new}} \leftarrow \mathrm{Expand}(u, \Psi(u))$ ▷ $\Psi(u)$: existing sibling labels
        if $D_{i+1,\mathrm{new}} \in \Psi(u)$ **then**
          continue
        end if
        if $i+1 < \mathcal{L}$ **then** ▷ non-leaf validation
          $s_{\max} \leftarrow \max_{w \in V_{\mathrm{wiki}}(D_{i+1,\mathrm{new}})} \mathrm{sim}_{\mathrm{str}}(D_{i+1,\mathrm{new}}, w)$
          if $s_{\max} < \tau_{\mathrm{wiki}}$ **then**
            continue
          end if
        end if
        Add $D_{i+1,\mathrm{new}}$ (and its "unk" child if $i+1 < \mathcal{L}$) to $\mathcal{T}$; init $\mathrm{Beta}(1,1)$
        $c \leftarrow D_{i+1,\mathrm{new}}$
      end while
      Append c to P_b; $u \leftarrow c$
    end for
    $\ell_b \leftarrow u$ ▷ leaf node of P_b
    Leaf corpus: $\mathcal{C}(\ell_b) \leftarrow \mathrm{RetrieveAndClean}(\ell_b)$
  end for
  Part II: Web-grounded two-roles self-play and updates.
  for $b \leftarrow 1$ to B **do**
    Challenger: propose G QA pairs from $d \sim \mathcal{C}(\ell_b)$; invalid ones receive penalty ρ; valid set $\mathcal{Q}(d)$ is kept by $\Gamma(\cdot)$.
    if $\mathcal{Q}(d) = \emptyset$ **then**
      continue
    end if
    Solver: for each $(q, y^\star) \in \mathcal{Q}(d)$, sample G answers and compute $\widehat{\mathrm{Var}}(q, y^\star)$ via $v(\cdot,\cdot)$.
    if $\forall (q, y^\star) \in \mathcal{Q}(d),\ \widehat{\mathrm{Var}}(q, y^\star) = 0$ **then**
      continue
    end if
    Rewards: Solver uses correctness $r_S = v(\hat{y}, y^\star)$; Challenger uses $r_C = \exp\Big(-\frac{(\widehat{\mathrm{Var}}-0.25)^2}{\sigma}\Big)$ for valid QA and ρ otherwise.
    Role balancing: uniformly sample one valid QA $(q^\star, y^\star) \sim \mathrm{Unif}(\mathcal{Q}(d))$ to form the Solver training group.
    Use the Challenger group (size G) and the Solver group (size G) to separately calculate the advantage $\mathcal{A}$ of Dr. GRPO.
    Posterior feedback: $g \leftarrow \mathbf{1}\Big[\widehat{\mathrm{Var}}(q^\star, y^\star) \geq \tau_{\mathrm{var}}\Big]$.
    for all $u \in P_b$ **do**
      Append g to buffer(u) (keep last μ) and recompute $(\tilde{\alpha}(u), \tilde{\beta}(u))$.
    end for
  end for Update the $\pi_{\theta_t - 1}$ to π_{θ_t} using the advantage $\mathcal{A}$.
end for
return trained π_{θ_T} and tree $\mathcal{T}$

Sub-domain Expansion Prompt (Non-leaf Nodes)

You are helping to construct a hierarchical knowledge tree in the main domain: `{main_domain}`. The current node path is: `{path_in_str}`.

Your task: Propose **ONE NEW SUB-DOMAIN** that will become a direct child of `"{parent_name}"`.

This tree is intended for school and early-university level `{main_domain}`. It will be used to generate exam-style and competition-style problems (multiple-choice or short-answer), similar to AMC / AIME / olympiad-style, as well as standard early undergraduate courses.

Optional sibling guidance (include only if siblings are provided).
Context about siblings (soft guidance):

- Under the same parent as `"{parent_name}"`, there are already some sibling sub-domains: `{siblings}`.

- This list is provided **only** to help you avoid:
  - proposing a label that is almost the same as a sibling, or
  - proposing a topic that is obviously a subtopic of a sibling instead of `"{parent_name}"`.
- The primary objective is still to create a standard curriculum-style sub-domain for `"{parent_name}"`.

Illustrative examples of good hierarchical structure (examples only):

- `{main_domain}` $\rightarrow$ Algebra / Geometry / Number Theory / ...
- Typical sub-domains under `"Algebra"`: Equations and Inequalities / Polynomials / ...

Strict requirements for the new sub-domain.

1. **Subset relation and level of generality.**
   - The new sub-domain must be strictly more specific than `"{parent_name}"`.

   - It should look like a chapter/section title in a school/early-university textbook.
2. **Difficulty and scope constraint (primary).**
   - Focus on standard curriculum topics; avoid graduate-level or research-only topics.
3. **Existing children under this parent (local de-duplication).**
   - The new sub-domain must **not** be identical or almost identical to any existing child: `{existing_children}`.

4. **Naming style.**
   - Use clear and concise names; avoid unnatural over-specific phrasing.
5. **Anti-hallucination rule (crucial).**
   - You **must not** invent new theorem names or dubious terminology.
   - If unsure, choose a simpler, classical topic instead.
6. **Domain purity.**

   - The label must stay within the target domain and should not reference other domains.

If you believe there are **no further meaningful sub-domains** under `"{parent_name}"` that:

- are not already covered by the existing children listed above,
- fit the school / early-university scope, and
- correspond to standard, widely used topics,

then you must output exactly `"No More"`.

Strict response format.

- Propose **exactly one** new label.
- Enclose it between `[Proposition Start]` and `[Proposition End]`, e.g.:

```
[Proposition Start]Quadratic Equations[Proposition End]
```

Now, provide your proposed label.

Knowledge-Point Expansion Prompt (Leaf Nodes)

You are helping to construct a hierarchical knowledge tree in the main domain: `{main_domain}`. The current node path is: `{path_in_str}`.
Your task: propose **ONE NEW ATOMIC KNOWLEDGE POINT** that will become a direct child of `"{parent_name}"`. A knowledge point must be a minimal unit that is directly usable to construct exam/contest-style problems, such as a named theorem/lemma/proposition, a standard definition used in problems, a classical example, or a standard algorithm/construction.

`{Optional sibling guidance (lightweight, include only if siblings are provided):}`
Sibling knowledge points (soft guidance):

- Under the same parent `"{parent_name}"`, there may already be other knowledge points: `{existing_children}`
- Avoid near-duplicates; siblings are only a local de-duplication hint.

STRICT REQUIREMENTS FOR THE NEW KNOWLEDGE POINT:

1. **Scope and granularity**
   - It must be strictly narrower than `"{parent_name}"` and correspond to **EXACTLY ONE** atomic unit:
     - theorem / lemma / proposition, OR
     - standard definition, OR
     - classical configuration/example, OR
     - standard algorithm/construction.
   - It should look like a short standalone textbook entry.
2. **Difficulty and usability (primary)**
   - It should support multi-step but standard contest / early-undergrad problems.
   - Avoid research-level, overly advanced, or non-canonical topics.
3. **Existing children under this parent (local de-duplication)**
   - The new knowledge point must **NOT** be identical or almost identical to any existing child: `{existing_children}`
4. **Naming style**
   - Use clear and concise names; do **NOT** always choose `"Definition: ..."`; mix theorems/examples/algorithms when natural.
5. **Anti-hallucination rule (crucial)**
   - You **MUST NOT** invent new theorem names, lemma names, or terminology.
   - Only propose names that are standard and widely used in textbooks.
6. **Domain purity**
   - The name must stay purely in the target domain.

If you believe there are **NO** further meaningful knowledge points under `"{parent_name}"` that:

- are not already covered by the existing children listed above, **AND**
- fit the intended scope, **AND**
- correspond to standard, widely used topics,

then you must output exactly `"No More"`.
STRICT RESPONSE FORMAT:

- Propose **EXACTLY ONE** new label.
- Enclose it between `[Proposition Start]` and `[Proposition End]`, e.g.:

`[Proposition Start]Pigeonhole Principle[Proposition End]`

Now, provide your proposed label.

MCQ Question Generation Prompt (Path-Conditioned, adapted from SPICE)

Your task is to create a **CHALLENGING** question from a document by using **BOTH**:

- (1) a hierarchical **LABEL PATH** that narrows down the mathematical domain, and
- (2) background **TEXT** about the most specific knowledge point.

Label Path (Domain Hierarchy)

`[BEGINNING OF THE LABEL PATH]`

`{path}`

`[END OF THE LABEL PATH]`

The label path lists nested mathematical domains from the broadest on the left to the most specific on the right.
Example: `"Mathematics -> Algebra -> Group Theory -> Sylow's Theorems"`

- The **LEFTMOST** labels are broad fields (e.g., `"Mathematics"`, `"Algebra"`).
- The **RIGHTMOST** label is an **ATOMIC KNOWLEDGE POINT** (e.g., `"Sylow's Theorems"`).
- Your question **MUST** belong to this path:
  - It must clearly be a mathematics question.
  - It must primarily test the **RIGHTMOST** knowledge point.
  - It may use context from earlier levels in the path to add difficulty and require multi-step reasoning.
- If the text contains information that is irrelevant to this label path, **IGNORE** that information.

Text

`[BEGINNING OF THE DOCUMENT]`
`{text}`
`[END OF THE DOCUMENT]`

The text is background material (e.g., web pages) about the atomic knowledge point at the end of the label path.
You must use this text to construct a mathematically meaningful, challenging question that fits the label path.

Instructions

Step 1: Path-Guided Complex Information Extraction

****PRIORITY: Use the label path to focus on mathematically relevant, non-trivial content.****

1. Interpret the label path:
   - Identify the main domain (e.g., `"Mathematics"`).
   - Identify intermediate sub-domains (e.g., `"Representation Theory"`, `"Lie Theory"`).
   - Identify the atomic knowledge point (the last label).
2. Scan the text and identify information that:
   - Is directly about the atomic knowledge point, or
   - Naturally belongs to the specified path (e.g., theorems, constructions, examples, or techniques in that subfield).
3. Among that information, focus on content that requires connecting multiple ideas, such as:
   - Relationships between several mathematical objects (groups, modules, functors, root systems, etc.).
   - Multi-step derivations, proofs, or constructions.
   - Interactions between definitions, lemmas, and theorems.
   - Situations where properties at a higher level in the path (e.g., `"Representation Theory"`) constrain or influence the atomic concept.

****AVOID**:**

- Generic reasoning or non-mathematical content, even if it appears in the text.
- Simple, standalone definitions that require no reasoning.
- Single, directly stated facts that can be copied as-is.
- Questions that do not clearly live inside the given label path.

Your goal is to pick a relationship or conclusion that:

- Is genuinely about the atomic knowledge point AND its mathematical context.
- Requires synthesis of multiple pieces of mathematical information.

Step 2: Difficulty Enhancement Process

****EXPLICITLY STATE YOUR HARDENING PROCESS****

Before generating the question, describe your strategy to make it harder:

1. What simple version would you avoid?
2. What complexity layers will you add?
3. Which concepts will you force students to connect?
4. What common shortcuts will you block?
5. How will you ensure multi-step reasoning is required?

Document this in the output field `"hardening_process"`.

Step 3: Advanced Question Generation

For each complex relationship identified, create a question that:

- Requires applying multiple concepts from different parts of the document
- Tests understanding of relationships, not just recall of facts
- Forces reasoning through multiple steps to reach the answer
- May require comparing or contrasting different scenarios
- Could involve `"what if"` scenarios based on principles in the text
- Tests ability to apply concepts to slightly modified situations

****CRITICAL - Self-Contained Requirements**:**

- Questions must be 100% self-contained and standalone
- NEVER use phrases like: `"according to the text"`, `"in the document"`, `"as mentioned"`, `"the passage states"`, `"based on the analysis"`, etc.
- Write as if for a formal exam with no reference material
- Include all necessary context within the question itself
- Define any specialized terms if needed for clarity

Step 4: Difficulty-Driven Design

****TARGET: Generate HARD/EXTRA HARD questions by design****

- HARD: Synthesize 4+ concepts; multi-step problem solving; pattern recognition
- EXTRA HARD: Complex system analysis; counter-intuitive applications; edge cases

Design questions that **CANNOT** be answered by:

- Looking up a single fact
- Finding one sentence with the answer
- Simple keyword matching

Step 5: Knowledge Integration Requirements

Document the reasoning path that shows why this is a difficult question:

- List 3+ distinct pieces of information needed from different parts of the document
- Show the logical connections required between these pieces
- Explain why simple lookup won't work
- Include intermediate reasoning steps

Step 6: Multiple Choice Design Guidelines

Create a multiple choice question with 4 options following these STRICT rules:

- **Length Balance**: All options must be approximately equal length (+/- 20%).
- **Unit Consistency**: All numerical answers must use identical units and formatting.
- **Tone Neutrality**: Avoid overly certain language (`"definitely"`, `"always"`, `"never"`) unless justified.
- **Plausibility**: All distractors must be genuinely plausible based on partial understanding.

Format:
Question: [Complete, self-contained question with all necessary context]
A) [Balanced length option]
B) [Balanced length option]
C) [Balanced length option]
D) [Balanced length option]

****Distractor Design**:**

- Common calculation errors from the multi-step process
- Results from applying only partial reasoning
- Mixing up related concepts from the document
- Reasonable approximations that miss key factors

Step 7: Self-Testing Filter (AFTER MCQ Creation)

****SOLVE YOUR OWN MCQ AS A STUDENT WOULD****

Now test the complete multiple choice question:

1. What's the quickest path a student might try with these options?
2. Can you eliminate 2+ options without full understanding? If yes, redesign distractors.
3. Does seeing the options make the answer obvious? If yes, improve distractors.
4. Count the reasoning steps required even with options visible - if less than 3, REJECT.
5. Time estimate: Would this MCQ take <30 seconds? If yes, make it harder.
6. Could a student guess correctly by pattern matching the options? If yes, rebalance.

Document your solving process in `"self_test_solution"`.

Step 8: Final Complexity Verification

Before finalizing, verify your question is **NOT** Easy by checking:

- Can it be answered by finding one sentence? If yes, redesign.
- Does it require connecting multiple document sections? If no, add complexity.
- Would someone need to understand relationships, not just facts? If no, refocus.
- Are all MCQ options balanced and using consistent formatting? If no, revise.
- Did your self-test of the MCQ take more than 1 minute? If no, increase difficulty.

Output Format

FIRST, think step-by-step about your question design (this is your private thinking).
THEN, provide your complete analysis in a JSON object with these fields.

CRITICAL: Output **ONLY** valid JSON without any markdown formatting or code blocks.
DO **NOT** wrap your JSON in "' or `"json"` markers.
Start directly with { and end with }.
Required fields:

- `"identified_answer"`
- `"answer_quote"`
- `"hardening_process"`
- `"question"`
- `"correct_answer"`
- `"self_test_solution"`
- `"knowledge_and_reasoning_steps"`
- `"question_difficulty"`

Free-form Question Generation Prompt (Path-Conditioned, adapted from SPICE)

Your task is to create a **CHALLENGING** question from a document by using **BOTH**:

- (1) a hierarchical **LABEL PATH** that narrows down the the mathematical domain, and
- (2) background **TEXT** about the most specific knowledge point.

Label Path (Domain Hierarchy)

`[BEGINNING OF THE LABEL PATH]`

`{path}`

`[END OF THE LABEL PATH]`

The label path lists nested mathematical domains from the broadest on the left to the most specific on the right.
Example: `"Mathematics -> Algebra -> Group Theory -> Sylow's Theorems"`

- The **LEFTMOST** labels are broad fields (e.g., `"Mathematics"`, `"Algebra"`).
- The **RIGHTMOST** label is an **ATOMIC KNOWLEDGE POINT** (e.g., `"Sylow's Theorems"`).
- Your question **MUST** belong to this path:
  - It must clearly be a mathematics question.
  - It must primarily test the **RIGHTMOST** knowledge point.
  - It may use context from earlier levels in the path to add difficulty and require multi-step reasoning.
- If the text contains information that is irrelevant to this label path, **IGNORE** that information.

Text

`[BEGINNING OF THE DOCUMENT]`
`{text}`
`[END OF THE DOCUMENT]`

The text is background material (e.g., web pages) about the atomic knowledge point at the end of the label path.
You must use this text to construct a mathematically meaningful, challenging question that fits the label path.

Instructions

Step 1: Path-Guided Complex Information Extraction

****PRIORITY: Use the label path to focus on mathematically relevant, non-trivial content.****
`... (same as above) ...`

Step 3: Advanced Question Generation (Free-Form Answer)

For each complex relationship identified, create a question that:

- Requires applying multiple concepts from different parts of the document
- Tests understanding of relationships, not just recall of facts
- Forces reasoning through multiple steps to reach the answer
- May require comparing or contrasting different scenarios

The answer must be a typed free-form answer extracted or computed from the document:

- A numeric value (integer or real number)
- A symbolic or algebraic expression
- A short string (name, label, or concept) that appears in or is uniquely determined by the document

Step 4–8

`(Identical self-contained requirements, difficulty checks, and self-testing as in the MCQ prompt, except that the output is a single free-form answer rather than an option letter.)`

Output Format

Output **ONLY** valid JSON with required fields:

- `"identified_answer"`
- `"answer_quote"`
- `"hardening_process"`
- `"question"`
- `"correct_answer"`
- `"answer_type"`
- `"self_test_solution"`
- `"knowledge_and_reasoning_steps"`
- `"question_difficulty"`