Interactive Evolution: A Neural-Symbolic Self-Training Framework For Large Language Models

Fangzhi Xu[1,2,5]* **Qiushi Sun**[3] **Kanzhi Cheng**[4] **Jun Liu**[1,5,6]† **Yu Qiao**[2] **Zhiyong Wu**[2]†
[1]School of Computer Science and Technology, Xi'an Jiaotong University
[2]Shanghai AI Lab [3]The University of Hong Kong [4]Nanjing University
[5]Ministry of Education Key Laboratory of Intelligent Networks and Network Security
[6]Shaanxi Province Key Laboratory of Big Data Knowledge Engineering
{fangzhixu98, whucs2013wzy}@gmail.com liukeen@xjtu.edu.cn

Abstract

One of the primary driving forces contributing to the superior performance of Large Language Models (LLMs) is the extensive availability of human-annotated natural language data, which is used for alignment fine-tuning. This inspired researchers to investigate self-training methods to mitigate the extensive reliance on human annotations. However, the current success of self-training has been primarily observed in natural language scenarios, rather than in the increasingly important neural-symbolic scenarios. To this end, we propose an environment-guided neural-symbolic self-training framework named *ENVISIONS*. It aims to overcome two main challenges: (1) the scarcity of symbolic data, and (2) the limited proficiency of LLMs in processing symbolic language. Extensive evaluations conducted on three distinct domains demonstrate the effectiveness of our approach. Additionally, we have conducted a comprehensive analysis to uncover the factors contributing to *ENVISIONS*'s success, thereby offering valuable insights for future research in this area.

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Team et al., 2023) have undergone extensive training using massive data, enabling them to possess remarkable capabilities across diverse domains. One of the main recipes of LLMs' success is the post-pretraining effort to achieve alignment with downstream tasks (Taori et al., 2023; Yin et al., 2023). The effective alignment primarily relies on *the accessibility of a substantial volume of expensive human-annotated data*, employing techniques such as Supervised Fine-Tuning (SFT) (Ivison et al., 2023) or Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022). Recently, there has been a growing interest in developing self-training methods that enable fine-tuning of LLMs without human annotations, thereby reducing cost and streamlining the training process (Yuan et al., 2024).

Notable progress has been made in self-training techniques for natural language (NL) scenarios (Chen et al., 2024; Rosset et al., 2024), where researchers focus on improving LLMs by synthesizing their own natural language input-output pairs. However, in recent years, there has been a growing emphasis on delegating tasks to external tools/environments to expand the capability boundaries of LLMs. The shift in focus necessitates the generation of a symbolic intermediate representation a that can be executed in the environment to faithfully produce the desired output y. This neural-symbolic framework (Xu et al., 2024) has achieved significant success in complex planning (Liu et al., 2023a), mathematical reasoning (Gou et al., 2023), robotic planning (Hu et al., 2023), and agentic tasks (Zheng et al., 2023; Wu et al., 2024). In contrast to the abundance of NL annotation data (x-y), curating symbolic annotation (x-a-y) is significantly more challenging and costly due to the scarcity and inherent complexity of symbolic language (SL). In this paper, we delve into the exploration of effective self-training methods for LLMs within complex neural-symbolic scenarios, all without human-annotated symbolic data.

Current self-training approaches in empowering LLMs in SL-centric scenarios fall into two categories, each with its own drawbacks. *Distill-then-Finetune* (Ivison et al., 2023; Xu et al., 2023a), shown in Fig. 1(a), entails fine-tuning a less powerful LLM using distilled data obtained from a teacher LLM, such as GPT-4 (Achiam et al., 2023).

*Work done during internship at Shanghai AI Lab.
†Corresponding Author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12975–12993
July 27 - August 1, 2025

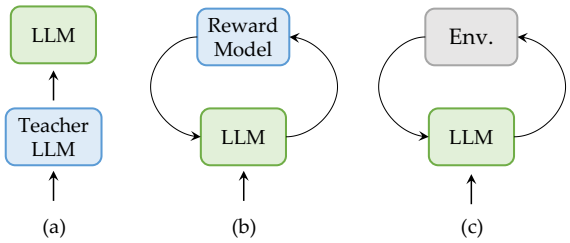


Figure 1: Weak-to-strong paradigms. (a) Distill-then-Finetune paradigm. (b) Reinforced Self-Training methods. (c) Environment-guided Self-Training paradigm.

Although this method is simple yet effective, its application is constrained by the requirement of an already existing stronger LLM and the associated high costs. Furthermore, the performance of the student LLM is upper-bounded by the capabilities of the teacher LLM. *Reinforced Self-Training* (Gulcehre et al., 2023; Singh et al., 2023), as shown in Fig. 1 (b), iteratively improves a weak LLM by leveraging reinforcement learning algorithms (Rafailov et al., 2024), guided by customized reward models. Nevertheless, reinforced methods are constrained by their inefficiency in training and/or reliance on human annotations for reward model training.

To address the limitations of previous approaches, this work focuses on two key challenges: enhancing the proficiency of LLMs in processing SL and eliminating the requirement for human-annotated data. Illustrated in Figure 1 (c), the proposed approach, called *Environment-guided (Env-guided) self-training*, involves iterative training of LLMs through interactions with an embodied environment. Built upon the approach, we propose an **ENV**-gu**I**ded **S**elf-tra**I**ning framework f**O**r **N**eural **S**ymbolic scenarios, named *ENVISIONS*. As an example, consider the training of LLMs for web browsing, i.e., training a web agent. Given a web manipulation task x, the agent generates multiple candidate actions $a \in \mathcal{A}$ and executes these actions within the web browser, resulting in both correct and incorrect outcomes. A self-rewarding algorithm is designed to post-process the agent's trajectories and create contrastive training pairs. These correct-incorrect trajectory pairs, along with a self-refining loss, are utilized to empower the LLMs to self-improve.

Through the Env-guided self-training approach, the LLMs leverage the interactive nature of the embodied environment to generate trajectories and learn symbolic language processing abilities, mitigating the need for human annotations. Through extensive evaluation, we found that *ENVISIONS* can consistently convert an existing LLM to a stronger one without reliance on existing stronger models or reward models. It's also worth noting that *ENVISIONS* and previous methods are not mutually exclusive, but we leave it as a future work to explore their synergy.

We highlight our contributions as follows:

(1) A neural-symbolic self-training framework: We propose a novel framework *ENVISIONS* for neural-symbolic self-training. The proposed framework can eliminate the need for human annotation or a stronger teacher model during self-training.

(2) Comprehensive evaluations and analysis: We extensively evaluate *ENVISIONS* across three domains to showcase its superiority over existing self-training methods. Our thorough analysis uncovers the reasons behind *ENVISIONS*'s exceptional performance, highlighting its potential as a new paradigm for neural-symbolic self-training.

(3) Insights on Env-guided neural-symbolic self-training: Our research provides valuable insights, supported by evidence, into the training process of Env-guided neural-symbolic self-training. These findings pave the way for future researches.

2 Related Work

Self-Training Methods. Self-training (Tao et al., 2024; Cao et al., 2024), offers a promising avenue for models to learn from their own outputs, reducing reliance on extensive human annotations. Recent advances (Gulcehre et al., 2023) leverage well-trained reward models to filter better training samples, and optimize the policy via reinforced self-training (Singh et al., 2023; Liu et al., 2023b). However, these approaches heavily rely on a strong reward model, which limits its applicability and training efficiency. Following the success of DPO (Rafailov et al., 2024), self-play frameworks have emerged as a new path that implicitly models the preferences among unlabeled rationales in iterative DPO styles (Chen et al., 2024; Rosset et al., 2024; Yuan et al., 2024). Nevertheless, these RL methods still face efficiency issues (Wang et al., 2023a). Beyond RL, previous works (Zelikman et al., 2022; Ni et al., 2022) optimize policy models within iterative SFT frameworks, yet neglecting the value of negative samples. Notably, previous efforts merely focus on the NL scenarios, but fail to apply in neural-symbolic settings.

Data Synthesis with LLMs. Obtaining high-quality reasoning traces to optimize LLMs has been a long-standing challenge (Mukherjee et al., 2023). Beyond well-established approaches utilizing data augmentation strategies to obtain diversified training data (Deng et al., 2023; Lee et al., 2024; Huang et al., 2025a). Recent efforts (Yue et al., 2023; Zeng et al., 2023; Cheng et al., 2024) primarily distill strong LLMs (Achiam et al., 2023; Anil et al., 2023) to generate novel samples in the given format. They either generate more diverse samples from seed data through self-instruct (Wang et al., 2023b) or enhance diversity through sample rewriting (Wei et al., 2023; Xu et al., 2025). However, current works mainly employ proprietary LLMs for data synthesis, which is a cost.

Neural-Symbolic Integration for LLMs. Neural-symbolic methods synergize the powerful generation capacity of LLMs with the reliability and interpretability of symbolic systems. Typically, PAL/PoT (Gao et al., 2023; Chen et al., 2023) synthesize executable programs as intermediate reasoning steps to solve numerical problems. This strategy of delegating problems to external solvers (e.g., Python interpreter), has gained significant traction (Xu et al., 2024; Sun et al., 2024; Huang et al., 2025b). For instance, (Gou et al., 2023) and (Pan et al., 2023) apply neural code generation and symbolic execution on math and logical reasoning respectively. Beyond reasoning, recent endeavors have extended the application into agent scenarios (Xu et al., 2023b; Qin et al., 2023) and leverage external feedback from the environment (Zheng et al., 2023; Yang et al., 2024) for refinement. However, these approaches mainly optimize LLM usage rather than providing autonomous self-improvement.

3 Methodology

3.1 Preliminaries

In neural-symbolic scenarios, based on the NL input x, LLMs are required to produce symbolic solution a to obtain the desired output y through the execution in the environment **ENV**. To adapt the weak LLMs to such complex settings and curate extensive (x, a, y) pairs, we propose to iteratively interact with **ENV** for self-improving LLMs. For each iteration i, the LLM π_{θ_i} will be provided with the task data set $\{(x^{(i)}, y^{(i)})\}$, with J input-output pairs. Without loss of generality, we assume the samples keep static between iterations.

Our framework *ENVISIONS*, presented in Fig. 2, is specifically designed to address two key challenges: (1) the scarcity of SL data and (2) the limited proficiency of LLMs in SL. Data scarcity limitation is addressed by the online exploration stage (Step ①-⑦). To convert LLMs from weak to strong in addressing SL, we employ LLM training using a carefully designed loss function and filtered data (Step ⑧-⑩). To simplify the expression, we omit the indicator of iteration i in the symbols. The overall procedure of *ENVISIONS* is also concluded in the pseudocode of Appendix D.

3.2 Online Exploration for SL Scarcity

Given the limited annotated SL data, *ENVISIONS* enables the policy LLM to autonomously generate symbolic solutions by interacting with the environment **ENV**. This process is named *Online Exploration*, which includes three main aspects 1) self-exploration (Step ①-③); 2) self-refinement (Step ④-⑥); and 3) self-rewarding (Step ⑦).

Self-exploration. Given the NL input x, the policy model π_θ first generates K diverse symbolic outputs (Step ①), formulated as $\{a_k\}_{k=1}^K \sim \pi_\theta(\cdot|x)$. These intermediate outputs will be executed in **ENV** (Step ②) to obtain the binary feedback $\{b_k\}_{k=1}^K$ based on y (Step ③). This procedure allows π_θ to explore the environment autonomously and search for diverse symbolic solutions.

Self-refinement. Considering the complexity of SL, solutions generated by the LLM may contain mistakes in symbolic format, which significantly impair the efficiency of exploration. To address this, we utilize the above self-explored solutions $\{a_k\}_{k=1}^K$ as references to regenerate new refined symbolic solutions (Step ④), formulated as $\{\widetilde{a}_k\}_{k=1}^K \sim \pi_\theta(\cdot|x; a_k)$. Similarly, these outputs will be executed in **ENV** (Step ⑤) and receive the corresponding binary reward $\{\widetilde{b}_k\}_{k=1}^K$ (Step ⑥).

Self-rewarding. Feedback from **ENV** merely gives the binary rewards. However, it remains challenging to discern preferences among various positive solutions or obtain valuable feedback from negative solutions. Motivated by it, we propose a soft reward score through sequence output probabilities with the following calculation:

$$r = \overline{p}_\theta(a|x) = \frac{1}{||a||} \sum_t \log p_\theta(a_t|x; a_{<t}), \quad (1)$$

where $||a||$ is the length of the symbolic solution a. Based on this definition, the soft self-rewards

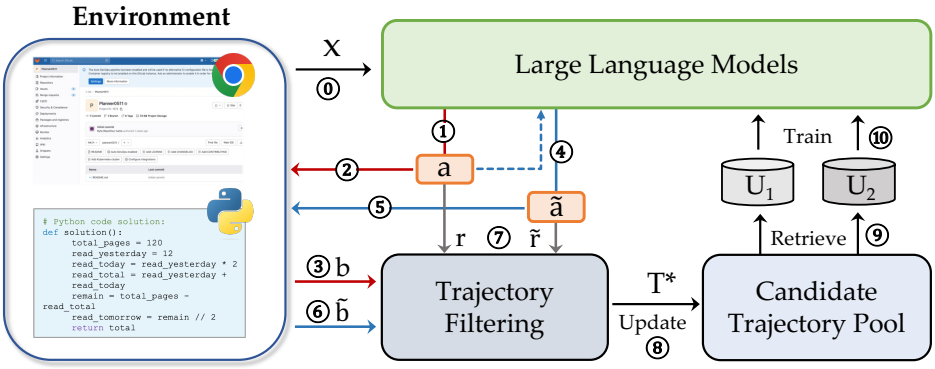


Figure 2: The neural-symbolic self-training framework *ENVISIONS*. → denotes *self-exploration* process (Step ①-③), → indicates *self-refinement* (Step ④-⑥), and → is *self-rewarding* (Step ⑦). a is the solution, and b is the binary feedback from the environment based on the execution of a.

of a_k and $\widetilde{a}_k$ are derived respectively as r_k and $\widetilde{r}_k$. Considering that no extra reward model is involved, we name it *self-rewarding*[1] (Step ⑦).

3.3 Data Selection and Training Strategies

After the online exploration stage, the candidate trajectories are constructed as $T_k = (x, y, a_k, b_k, r_k)$ and $\widetilde{T}_k = (x, y, \widetilde{a}_k, \widetilde{b}_k, \widetilde{r}_k)$, which are sourced from *self-exploration* and *self-refinement* respectively. Next, we select premium trajectories for training the LLM π_{θ_i}.

Trajectory filtering and candidate pool updating. To control the candidate number and maintain high-quality trajectories. we select the superior one from T_k and $\widetilde{T}_k$ to update the candidate pool (Step ⑧). To facilitate automatic selection, we incorporate binary rewards and self-rewards for assessment. Following the principle of prioritizing execution correctness, we derive the filtered trajectory T_k^*:

$$T_k^* = (x, y, a_k^*, b_k^*, r_k^*) = \begin{cases} (x, y, a_k, b_k, r_k), & \text{if } b_k = 1 \text{ and } \widetilde{b}_k = 0, \\ (x, y, a_k, b_k, r_k), & \text{if } b_k = \widetilde{b}_k \text{ and } r_k > \widetilde{r}_k, \\ (x, y, \widetilde{a}_k, \widetilde{b}_k, \widetilde{r}_k), & \text{otherwise.} \end{cases} \tag{2}$$

Notably, our filter strategy still maintains some trajectories with incorrect solutions but relatively higher rewards. These trajectories will serve as hard negative samples for the subsequent steps.

Supervised fine-tuning on positive solutions. As we have explored diverse trajectories in ENV, an intuitive way to bootstrap the performance of LLMs is fine-tuning with the positive solutions. Therefore, for each input x, we can retrieve the positive trajectories (i.e., $b = 1$) from the candidate pool. Giving priority to more valuable solutions, we rank the trajectories in descending order based on self-rewards, resulting in the positive set S^+. To mitigate overfitting, we enforce a maximum of N_1 positive-only solutions sampled for each input x:

$$U_1 = \{(x, a_m^+) \mid m \le \min(N_1, |S^+|) \text{ and } T_m^+ \in S^+\} \tag{3}$$

where $m \in \mathbb{Z}^+$ means the index in the ranked set and $|\cdot|$ returns the number of trajectories in the given set. $T_m^+ = (x, y, a_m^+, b_m^+ r_m^+)$ denotes the trajectories in S^+. Following the principle of MLE, the optimized loss function can be written as:

$$\mathcal{L}_1 = -\sum_{(x,a^+)\sim U_1} \log p_\theta(a^+|x) \tag{4}$$

RL-free loss to learn from mistakes. Under the neural-symbolic setting, negative solutions may comprise a substantial portion of exploration trajectories, while also offering valuable insights for model enhancement. *ENVISIONS* explores motivating the policy LLM to learn from mistakes during the weak-to-strong process. We can obtain the ranked negative set S^- from the candidate pool. For each input x, at most N_2 positive-negative pairs will be constructed from S^+ and S^-:

$$U_2 = \{(x, a_m^+, a_m^-) \mid T_{m+|U_1|}^+ \in S^+ \text{ and } T_m^- \in S^- \text{ and } m \le \min(N_2, |S^+| - N_1, |S^-|)\}, \tag{5}$$

where $T_m^- = (x, y, a_m^-, b_m^- r_m^-)$ denotes the trajectories in S^-. Limited by the difficulty and complexity of optimizing models in an RL manner (e.g., DPO (Rafailov et al., 2024)), it is challenging for reinforced-based methods (Chen et al., 2024; Rosset et al., 2024) to quickly adapt to the SL scenarios.

[1] *Self-rewarding* step in *ENVISIONS* is different from (Yuan et al., 2024), though they share the same name.

Therefore, we design the following contrastive RL-free loss function:

$$\mathcal{L}_2 = - \sum_{(x,a^+,a^-)\sim U_2} \log p_\theta(a^+|x;a^-) \quad (6)$$

It brings two main advantages: (1) the ability of self-refinement is acquired, which benefits the scalability to complex cases; (2) compared to reinforced losses, superior training efficiency is achieved. Finally, the overall loss function of each iteration is simply designed as $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$.

4 Experiments

4.1 Datasets

We evaluate the proposed framework on three distinct domains, each with its own environment: web agents (Chrome browser), math reasoning (Python compiler), and logical reasoning (Pyke engine). For agentic tasks, we select the widely-used web navigation benchmark MiniWob++ (Liu et al., 2018). For the math reasoning domain, we include: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), GSM-Hard (Gao et al., 2023), SVAMP (Patel et al., 2021), and AsDiv (Miao et al., 2020). For logical reasoning tasks, we include ProofWriter (Tafjord et al., 2021) and RuleTaker (Clark et al., 2021). To evaluate the generalization capability of our method, we reserve some datasets for out-of-distribution evaluation as shown in Table 1. Refer to Appendix A.2 for details.

4.2 Baselines and Training Details

Following the categorization of Figure 1, we consider the respective three lines of baselines. All baselines are reproduced under the same codebase for a fair comparison.

Distill-then-Finetune. GPT-4 and Claude-2 are selected as strong teacher LLMs in this approach. By prompting teacher LLMs, we obtain the symbolic trajectories with correct answers to fine-tune LLMs. Due to the high time and financial cost of calling API, each input will be prompted only once.

Reinforced Self-Training. We implement two RL-based self-training baselines: *Self-Rewarding* (Yuan et al., 2024) and *iterative SFT+DPO*. For the former, we follow the official implementation to first warm up the weak LLM using human annotation from OpenAssistant (Köpf et al., 2024). The latter is a variation of *ENVISIONS* that mainly separates the training into two stages, with positive solutions for SFT and positive-negative pairs for DPO.

Env-guided Self-Training. Since there is no existing baseline for this approach, we consider extending the NL-centric self-training method STaR (Zelikman et al., 2022) to support neural-symbolic scenarios. It is worth noting that STaR only uses positive samples for behavior cloning. For the methods under this paradigm (including *ENVISIONS*) we optimize LLM from scratch in each iteration with the updated training samples.

Except for *Distill-then-Finetune* baselines, all other methods utilize few-shot prompting to acquire training samples as a cold start. The few-shot numbers for the web agent, math, and logic domains are set to 1, 3, and 1 respectively. We also include few-shot results on weak LLM for comparison. For a fair evaluation, all baselines are optimized to generate symbolic outputs (e.g., Python code) rather than natural language outputs, following PoT style (Chen et al., 2023). Please refer to Appendix A for other details.

We use LLaMA2-Chat 7B/13B models for the evaluation. At each generation step (i.e., Step ①,④), the candidate size K is set to 5. The total iteration number for web agent, math, and logic tasks is set to 5, 10, and 8 respectively, unless otherwise stated. For each input, N_1 and N_2 are fixed to 10 and 2 respectively. All the self-training experiments are implemented on 8*A100 of 80GB VRAM. Please refer to Appendix A.1 for other details.

4.3 Main Results

Table 2 presents the evaluation results. For supplementary experiments on other backbone LLMs, please refer to Section 4.5 and Appendix C.3.

***ENVISIONS* presents consistent superiority over strong baselines.** Evolving from LLaMA2-Chat, *ENVISIONS* notably boosts average performance by 30.00% and 24.95% for the 7B and 13B variants, respectively. Compared with *Distill-then-Finetune* methods, 5.66%-7.13% gains are obtained. Apart from its superior performance, *ENVISIONS* presents scalability without the associated costs of using strong LLMs. It exhibits clear advantages over *Reinforced Self-Training* and other *Env-guided Self-Training* methods, delivering average gains of 2.78%-14.47%. The competitive performances, with the training efficiency, makes *ENVISIONS* stand out among these strong baselines.

Domains	Held-in Tasks	Held-out Tasks	#Samples	Static ?	Env.
Web Agent	MiniWob++	-	2,200	No	Chrome browser
Math Reasoning	GSM8K, MATH	GSM-H, SVAMP, AsDiv	13,492	Yes	Python compiler
Logic Reasoning	ProofWriter	RuleTaker	3,600	Yes	Pyke engine

Table 1: Details and statistics of evaluated domains. *#Samples* denotes the number of input samples per iteration. *Static?* indicates whether the input data remains the same across all iterations.

Models	Agent	Math Reasoning					Logical Reasoning		Avg.
	MiniWob++	GSM8K	MATH	GSM-H	SVAMP	ASDiv	ProofWriter	RuleTaker	
Is Held-out ?	✗	✗	✗	✓	✓	✓	✗	✓	
LLaMA2-Chat (7B)									
LLaMA2-Chat (few-shot)	51.14	12.21	1.32	10.69	22.00	25.86	34.83	47.44	25.69
Distill-then-Finetune									
GPT-4 + LLaMA2-Chat	81.14	53.07	18.84	47.84	66.80	68.75	34.33	48.88	52.46
Claude-2 + LLaMA2-Chat	82.80	52.69	18.17	44.88	70.50	69.85	36.17	49.17	53.03
Reinforced Self-Training									
Self-Rewarding	69.39	40.03	10.70	31.69	58.20	61.55	32.17	50.04	44.22
Iterative SFT+DPO	77.05	54.81	14.75	47.08	70.10	66.22	49.00	58.82	54.73
Env-guided Self-Training									
STaR + Env.	83.71	58.23	15.97	46.63	67.50	68.46	50.17	58.60	55.91
ENVISIONS	**85.38**	**58.98**	**19.00**	**48.52**	**72.40**	**69.80**	**52.83**	**62.63**	**58.69**
LLaMA2-Chat (13B)									
LLaMA2-Chat (few-shot)	60.00	34.87	6.07	28.96	45.00	46.61	35.83	51.50	38.61
Distill-then-Finetune									
GPT-4 + LLaMA2-Chat	80.15	62.85	23.64	53.98	73.00	73.52	34.17	50.61	56.49
Claude-2 + LLaMA2-Chat	84.77	62.24	23.47	52.08	76.30	74.05	36.00	48.45	57.17
Reinforced Self-Training									
Self-Rewarding	74.55	50.80	13.97	41.24	74.10	71.37	37.33	56.66	52.50
Iterative SFT+DPO	82.73	63.84	22.32	50.57	77.30	70.94	51.00	59.47	59.77
Env-guided Self-Training									
STaR + Env.	85.15	63.61	20.57	53.37	74.70	74.76	52.33	60.33	60.60
ENVISIONS	**87.12**	**68.31**	**26.04**	**57.54**	**78.30**	**75.52**	**54.83**	**60.84**	**63.56**

Table 2: Main Results on Agent, Math Reasoning and Logical Reasoning domain. Notably, we report the average performance across extensive tasks in MiniWob++ benchmark (refer to Appendix C.8 for details). *Is Held-out?* row distinguishes the held-in and held-out tasks. *Avg.* column reports the averaged performances on all tasks.

Env-guided Self-Training exhibits strong scalability to neural-symbolic scenarios. Compared to the other two approaches, *Env-guided Self-Training* is more applicable to complex neural-symbolic scenarios, especially in agentic tasks where NL-centric methods inherently exhibit limitations. Besides the great performances of *ENVISIONS*, previous methods *STaR* can also benefit from the supervision signals acquired in **Env**, which helps the evolution progress.

4.4 Evolution Progress

In Figure 3, we present the iterative evolution curves of the self-training frameworks with LLaMA2-Chat (13B) as the LLM, which clearly shows the procedure of weak-to-strong transformation. We leave the discussion on the evolution of both performance and explored sample numbers with the 7B version in Appendix C.1.

***ENVISIONS* combines high evolutionary efficiency and sustainability.** In the initial iterations, *ENVISIONS* demonstrates swift adaptability to different scenarios. This indicates that exceptional performance can be achieved with minimal time for data collection in *ENVISIONS*. Additionally, *ENVISIONS* stands out as a more sustainable option when compared to other baselines. For instance, in math reasoning tasks of Fig. 3(b), all baseline methods achieve saturated performance levels by 6^{th} iteration. However, our framework continues to exhibit evolutionary progress.

Reinforced baselines are largely flawed during iterations. The incorporation of reinforced loss (e.g., DPO) brings difficulty in optimization and greatly restricts the evolutionary scales of the LLM to adapt to the neural-symbolic scenarios. *Self-Rewarding* exhibits largely reduced benefits during iterations, in contrast to its impressive performance in NL-centric tasks. For *Iterative SFT+DPO*, the

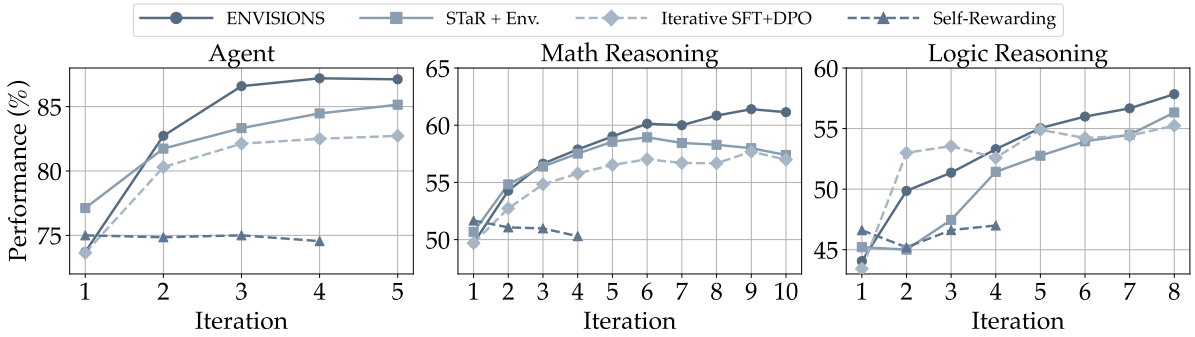


Figure 3: Performance evolution of self-training methods on LLaMA2-Chat 13B model. *Reinforced Self-Training* approaches are represented by dashed lines, while *Env-guided* ones are in solid lines.

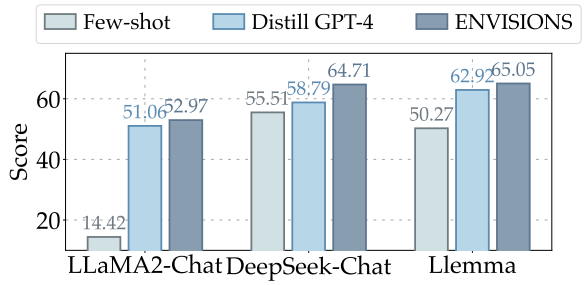


Figure 4: Generalization to different LLMs. The performances on math reasoning tasks are reported.

SFT stage boosts the ability in effective exploration. However, the subsequent DPO stage imposes a slight improvement while significantly reduce the training efficiency.

4.5 Generalization to Various Backbones

To demonstrate the generalizability, we apply *ENVISIONS* to enhance two additional base LLMs on mathematical reasoning tasks: (1) DeepSeek-Chat (DeepSeek-AI, 2024) model of 7B size, which is a foundational LLM and (2) Llemma (Azerbayev et al., 2023), a competent domain-specific LLM optimized for math reasoning. Figure 4 shows the comparisons with *Few-shot Prompting* and *Distill GPT4-then-Finetune*. It is observed that our framework still works for strong foundation LLMs, with 9.20% and 14.78% performance boosts for DeepSeek-Chat and Llemma respectively. This demonstrates that our framework can not only convert LLMs from weak to strong, but also elevate LLMs from strong to stronger.

5 Analysis

This section will make an in-depth analysis of the underlying reason behind *ENVISIONS*'s success.

5.1 What is the Impact of Key Components?

Some key components are ablated independently to verify their effectiveness in Table 3. *w/o self-refine* ablates both the self-refinement process (i.e., Step ④-⑥) and $\mathcal{L}_2$. *w/o self-rewards* replaces the trajectory ranking on the self-rewarding strategy with random sampling. *w/o long-term memory* only utilizes the generated trajectories from the current iteration for training. *w/o $\mathcal{L}_2$ loss* ablates the optimization with positive-negative pairs.

Of these components, self-refine-oriented optimizations (i.e., self-refinement and $\mathcal{L}_2$ loss) play key roles in boosting the performances. As one of the key contributions, the design of $\mathcal{L}_2$ loss leads to 3.10%-4.57% improvements in *ENVISIONS*. It makes full use of negative trajectories while maintaining training efficiency in an RL-free style. Especially in agent tasks, *ENVISIONS* benefits a lot from $\mathcal{L}_2$ loss, with 3.49%-5.53% gains.

5.2 What is Behind the Superiority?

We provide in-depth evidence and analysis on the superiority of *ENVISIONS* from three distinctive views: (1) exploratory ability and stability; (2) log probability margin between positive and negative solutions; and (3) diversity of synthetic samples. The analysis is on LLaMA2-Chat 7B and we leave the discussion of 13B in Appendix C.6.

Balanced exploratory ability and stability are key to success in weak-to-strong. To effectively navigate the environment and acquire new skills autonomously, two factors are crucial: 1) promptly resolving extensive samples to collect correct trajectories, and 2) minimizing the potential loss of knowledge from previously solved samples. We employ two metrics *exploratory ability* and *stability* to evaluate the LLM (both of them are the higher, the better). Refer to Appendix B for definition de-

Models	Agent	Math Reasoning					Logical Reasoning		Avg.
	MiniWob++	GSM8K	MATH	GSM-H	SVAMP	ASDiv	ProofWriter	RuleTaker	
LLaMA-2-Chat (7B)									
ENVISIONS	**85.38**	**58.98**	**19.00**	**48.52**	**72.40**	**69.80**	**52.83**	**62.63**	**58.69**
w/o self-refine	84.92	56.86	18.20	48.14	68.70	67.89	42.00	58.60	55.66
w/o self-reward	84.47	58.61	18.75	47.92	71.10	68.46	47.33	59.61	57.03
w/o candidate pool	83.86	57.77	17.55	47.16	70.90	68.03	49.17	59.18	56.70
w/o $\mathcal{L}_2$ loss	81.89	55.88	18.90	47.16	67.60	67.75	47.67	57.88	55.59
LLaMA-2-Chat (13B)									
ENVISIONS	**87.12**	**68.31**	**26.04**	**57.54**	**78.30**	**75.52**	**54.83**	**60.84**	**63.56**
w/o self-refine	84.24	65.96	24.95	55.34	77.70	73.90	51.00	57.59	61.34
w/o self-reward	85.45	67.02	25.59	55.57	77.80	74.05	51.50	60.69	62.21
w/o candidate pool	85.61	66.89	24.19	53.07	77.20	72.90	51.33	58.96	61.27
w/o $\mathcal{L}_2$ loss	81.59	63.08	20.00	51.18	74.30	71.23	50.33	60.19	58.99

Table 3: Ablation studies on key components.

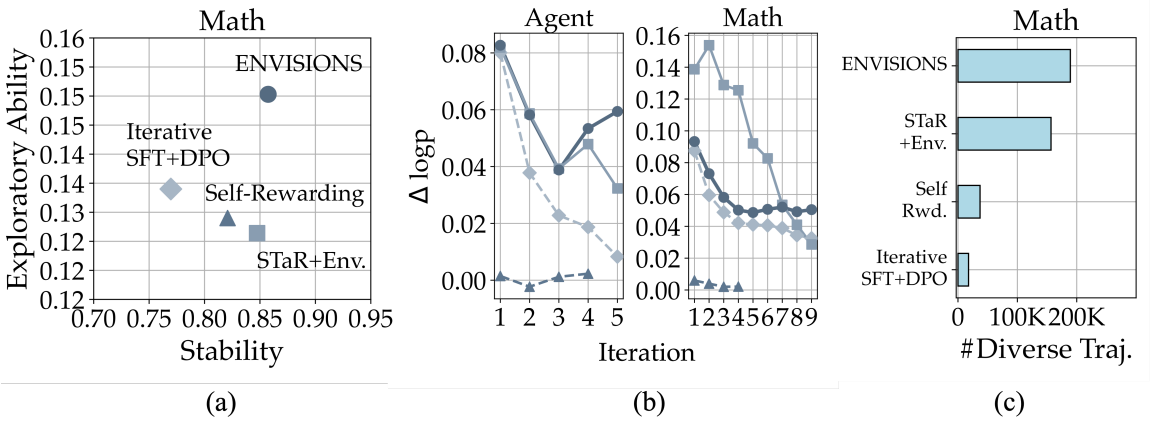


Figure 5: In-depth analysis from three perspectives. Plots in fig.(b) correspond to the methods represented by the same colors in fig.(a).

tails. In Figure 5(a), *ENVISIONS* demonstrates remarkable performance in achieving a balance between exploratory ability and stability. By leveraging the candidate pool and self-rewards, *ENVISIONS* effectively retains high-quality positive solutions during training, significantly mitigating the issue of forgetting previous trajectories. Additionally, the RL-free loss $\mathcal{L}_2$ enables flexible updates of the LLM, enhancing its exploration capabilities.

Clearly distinguishing positive and negative solutions can help the LLM optimization. During the optimization process, it is inevitable for the log probability of both positive and negative trajectories to increase simultaneously (Hong et al., 2024). However, clearly keeping the probability margins (Δlogp) between positive-negative pairs is crucial to facilitate the optimization. Fig. 5(b) shows the analysis of Δlogp during iterations. It is observed *ENVISIONS* keeps the margin within a reasonable range, while reinforced methods exhibit a rapid decrease in Δlogp. It indicates the unsuitablity of DPO to the exploration setting and the importance of feedback from **ENV**. Notably, *STaR+Env.* fails to keep the stable margins in the math domain, since it merely utilizes positive data for training, which fails to distinguish negative ones and leads to overfiting on the limited number of solutions. Such finding corresponds to the lack of exploratory ability in Fig. 5(a).

Diverse trajectories are what you need for self-training. In Fig. 5(c), we compare the number of correct and unique trajectories by the last iteration. It demonstrates the huge strengths of *ENVISIONS* in synthesizing diverse trajectories. It largely surpasses *Reinforced Self-Training* approaches, which is one of the underlying reasons for our superiority. In fact, the LLM updates in RL methods are restricted by *KL* constraints, which ultimately impact the diversity of the generated trajectories. Moreover, *Distill GPT-4* and *Distill Claude2* lead to 10,831 and 8,561 diverse trajectories with one iteration. Since repeatedly calling strong LLMs involves extremely high cost and cumbersome prompt optimizations, they are far from sustainable compared with *ENVISIONS*.

6 Conclusion

This paper focuses on converting LLMs from weak to strong in increasingly promising neural-symbolic scenarios, without human-annotated symbolic training data. In view of two key challenges, i.e., 1) the scarcity of symbolic training data, and 2) the inherent weakness of LLMs in addressing SL, we conclude the env-guided self-training approach. Built on it, we propose a novel neural-symbolic self-training framework *ENVISIONS*. Extensive experiments across three domains verify the remarkable performances. In-depth analysis on the superiority of *ENVISIONS* from three distinctive views provide novel insights for future researches.

Acknowledgement

This work was supported by National Key Research and Development Program of China (2022YFC3303600), National Natural Science Foundation of China (No. 62137002, 62293550, 62293553, 62293554, 62437002, 62477036, 62176209, 62176207), "LENOVO-XJTU" Intelligent Industry Joint Laboratory Project, Shaanxi Undergraduate and Higher Education Teaching Reform Research Program (No. 23BY195), and Xi'an Jiaotong University City College Research Project (No. 2024Y01), Project of China Knowledge Centre for Engineering Science and Technology, the Youth AI Talents Fund of China Association of Automation (Grant No.HBRC-JKYZD-2024-311).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*.

Boxi Cao, Keming Lu, Xinyu Lu, Jiawei Chen, Mengjie Ren, Hao Xiang, Peilin Liu, Yaojie Lu, Ben He, Xianpei Han, et al. 2024. Towards scalable automated alignment of llms: A survey. *arXiv preprint arXiv:2406.01252*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

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*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3882–3890.

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

DeepSeek-AI. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*.

Hanxu Hu, Hongyuan Lu, Huajian Zhang, Yun-Ze Song, Wai Lam, and Yue Zhang. 2023. Chain-of-symbol prompting elicits planning in large langauge models. *arXiv preprint arXiv:2305.10276*.

Muye Huang, Han Lai, Xinyu Zhang, Wenjun Wu, Jie Ma, Lingling Zhang, and Jun Liu. 2025a. Evochart: A benchmark and a self-training approach towards real-world chart understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 3680–3688.

Muye Huang, Lingling Zhang, Han Lai, Wenjun Wu, Xinyu Zhang, and Jun Liu. 2025b. Vprochart: Answering chart question through visual perception alignment agent and programmatic solution reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 3689–3696.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipali, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. 2024. Llm2llm: Boosting llms with novel iterative data enhancement. *arXiv preprint arXiv:2403.15042*.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2023b. Statistical rejection sampling improves preference optimization. In *The Twelfth International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984.

Subhabrata (Subho) Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv: Computation and Language*.

Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Alex Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. 2022. Learning math reasoning from self-sampled correct and partially-correct solutions. In *The Eleventh International Conference on Learning Representations*.

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

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacroce, Ahmed Awadallah, and Tengyang Xie. 2024. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, et al.

2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand. Association for Computational Linguistics.

Qiushi Sun, Zhirui Chen, Fangzhi Xu, Kanzhi Cheng, Chang Ma, Zhangyue Yin, Jianing Wang, Chengcheng Han, Renyu Zhu, Shuai Yuan, et al. 2024. A survey of neural code intelligence: Paradigms, advances and beyond. *arXiv preprint arXiv:2403.14734*.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Jianing Wang, Qiushi Sun, Nuo Chen, Chengyu Wang, Jun Huang, Ming Gao, and Xiang Li. 2023a. Uncertainty-aware parameter-efficient self-training for semi-supervised language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7873–7884, Singapore. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. 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, Toronto, Canada. Association for Computational Linguistics.

Jerry Wei, Le Hou, Andrew Lampinen, Xiangning Chen, Da Huang, Yi Tay, Xinyun Chen, Yifeng Lu, Denny Zhou, Tengyu Ma, and Quoc Le. 2023. Symbol tuning improves in-context learning in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 968–979, Singapore. Association for Computational Linguistics.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. Os-copilot: Towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2025. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge and Data Engineering*.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. 2024. Symbol-llm: Towards foundational symbol-centric interface for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13091–13116.

Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023b. On the tool manipulation capability of open-sourced large language models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. 2024. React meets actre: Autonomous annotations of agent trajectories for contrastive self-training. *arXiv preprint arXiv:2403.14589*.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

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

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms.

Longtao Zheng, Rundong Wang, and Bo An. 2023. Synapse: Leveraging few-shot exemplars for human-level computer control. *arXiv preprint arXiv:2306.07863*.

A Implementation Details

In this section, we provide some details of the implementation.

A.1 Training Details

The SFT training in both our framework and baselines is conducted on 8*A100 with a maximum length of 2,048. They are optimized and accelerated with Deepspeed Zero3 and FlashAttention2. The AdamW optimizer (Loshchilov and Hutter, 2017) is leveraged with a *Linear* learning rate of 2e-5. The SFT training epoch number of each iteration is set to 2, 1, 2 for agent, math reasoning, and logic reasoning tasks respectively.

For the DPO training stage in baseline methods, it is also conducted on 8*A100 with a maximum length of 2,048. The *Linear* learning rate is 5e-7 with a warm-up ratio of 0.1. The epoch number for each domain is the same as the SFT stage.

A.2 Test Tasks and Benchmark

The experiments in the main paper primarily cover three domains: web agent, math reasoning, and logic reasoning. We have concluded some key details in Table 1. In Table 4, we attach extra information on the test tasks and benchmark.

Unless otherwise stated, all these test tasks are evaluated under the zero-shot setting. For MiniWob++ benchmark, we select 44 tasks for the test (Cheng et al., 2024), each with 30 randomly generated samples. All the above settings are consistent among all baseline methods.

B Definition of *Exploratory Ability* and *Stability*

(1) Whether the policy LLM can rapidly explore large amounts of correct samples, and (2) whether it can mitigate the issue of forgetting previously-solved samples are two key factors to evaluate LLMs in interacting with the environment. We define *Exploratory Ability* (EA) and *Stability* (STB) respectively as the metrics. The calculation of the metrics is defined as follows:

Suppose that we have the input set M. In the i^{th} iteration, the solved sample (with correct trajectories) constitute of set M_i. $\bigcup_{j=1}^{i-1} M_j$ contains all the previously-solved samples from the iteration 1 to $i-1$. And $M_i \cup \bigcup_{j=1}^{i-1} M_j$ comprises the overlapped successful samples between the current iteration and preceding iterations. $M_i \setminus \bigcup_{j=1}^{i-1} M_j$ denotes the sample set that are newly solved in the current iteration i. Based on the definition, we accumulate to obtain the overall EA and STB of the entire process:

$$\begin{aligned} \text{EA} &= \sum_{i=2}^{T} \frac{|M_i \setminus \bigcup_{j=1}^{i-1} M_j|}{|\bigcup_{j=1}^{i-1} M_j|}, \\ \text{STB} &= \sum_{i=2}^{T} \frac{|M_i \cap \bigcup_{j=1}^{i-1} M_j|}{|\bigcup_{j=1}^{i-1} M_j|} \end{aligned} \tag{7}$$

where $|\cdot|$ is the number of samples in the given set. T is the total number of iterations.

Take the process of 2 iterations as an example, suppose the iteration 1 explores 1,000 correct samples. Iteration 2 obtains 1200 correct samples, including 800 previously-solved samples and 400 newly-solved samples. Then, EA $= 400/1000$ and STB $= 800/1000$.

C Supplementary Results

C.1 Evolution Progress

Apart from the performance evolution curves with the LLaMA2-Chat 13B model presented in Figure 3, we expand the discussion on the 7B version. In Figure 6, we visualize the evolution progress of self-training methods on both the model performance and the number of explored samples. The explored sample denotes that one input x is solved by at least one generated symbolic solution a_k (i.e., $b_k = 1$). We count the number of explored samples at each iteration to make the figure.

From the results, the performances of the frameworks are positively correlated with the ability to continuously explore correct trajectories. *ENVISIONS* presents great superiority, especially in the logic reasoning tasks. Compared with our proposed *Env-guided Self-Training* approach, *Reinforced Self-Training* approach appears to be weaker at exploring new samples. This finding is consistent with Figure 5 in the main paper.

C.2 Scaling of K

The hyper-parameter K controls the number of generated candidate symbolic solutions at each generation step. In the main results, we only implement $K = 5$ for illustration.

In Table 5, we present performances under various choices of K, including 2, 5, 10, and 15. Considering the training cost, we only include LLaMA2-Chat (7B) as the base LLM. From the results, we conclude the following takeaways:

Domains	Task name	Is Held-out?	#Test Samples	Beam Size	Max. Length	Sources
Web Agent	MiniWob++		30 (×44 tasks)	1	2,048	Liu et al. (2018)
Math Reasoning	GSM8K		1,319	2	2,048	Cobbe et al. (2021)
	MATH		4,001	2	2,048	Hendrycks et al. (2021)
	GSM-Hard	✓	1,319	2	2,048	Gao et al. (2023)
	SVAMP	✓	1,000	2	2,048	Patel et al. (2021)
	AsDiv	✓	2,096	2	2,048	Miao et al. (2020)
Logic Reasoning	ProofWriter		600	1	4,096	Tafjord et al. (2021)
	RuleTaker	✓	1,389	1	4,096	Clark et al. (2021)

Table 4: Details of test tasks and benchmarks.

Models	Agent	Math Reasoning					Logical Reasoning		Avg.
	MiniWob++	GSM8K	MATH	GSM-H	SVAMP	ASDiv	ProofWriter	RuleTaker	
K=2	78.56	53.60	17.37	44.96	67.20	66.84	35.17	49.82	51.69
K=5	**85.38**	**58.98**	19.00	**48.52**	**72.40**	69.80	52.83	**62.63**	**58.69**
K=10	79.24	58.30	21.89	48.29	67.90	69.75	53.50	61.99	57.61
K=15	79.55	57.47	**23.72**	46.63	69.80	**70.28**	**54.83**	59.54	57.73

Table 5: Scaling of K with LLaMA2-Chat (7B) as the base LLM. In the main results, we implement $K = 5$ for illustration.

Moderate value of K leads to the optimal performances. When $K = 5$, *ENVISIONS* demonstrates superior performances, especially on agentic tasks (i.e., MiniWob++ benchmark). However, when reducing the value of K (i.e., $K = 2$), the overall performances of *ENVISIONS* drop a lot. It indicates that keeping a moderate number of candidate solutions in each generation step benefits the self-training process.

Scaling of K does not bring significant improvements. Scaling K from 5 to 10 and 15 does bring improvements on some challenging tasks (e.g., MATH). However, this observation is not consistent across various tasks. Generally, the average performances remain stable with K increasing.

C.3 Generalization to Other Backbones

In Section 4.5 of the main paper, we have presented the generalization of *ENVISIONS* to various backbones. The promising results in mathematical domains demonstrate that *ENVISIONS* is compatible with a wide range of LLMs (from weak LLMs to stronger ones).

To further support our claims, we supplement the experiments on another popular LLM backbone Mistral-Instruct-v0.2 (7B). Limited by self-training time cost, we only implement it in the agentic domain. We include the strong baselines of *STaR+Env.* and *iterative SFT+DPO* for comparisons. Table 6 presents the experimental results.

It is observed that the superiority of *ENVISIONS* with the Mistral backbone is also obvious. The generalization capability is further verified. Compared with the previous SOTA method *STaR+Env.*, *ENVISIONS* achieves 12.95% superiority over it. And it also outperforms reinforced self-training baseline *iterative SFT+DPO* baseline by 4.77%.

Methods	MiniWob++	Δ
Few-shot Prompting	51.44	+26.51
iterative SFT+DPO	73.18	+4.77
STaR+Env.	65.00	+12.95
ENVISIONS	77.95	-

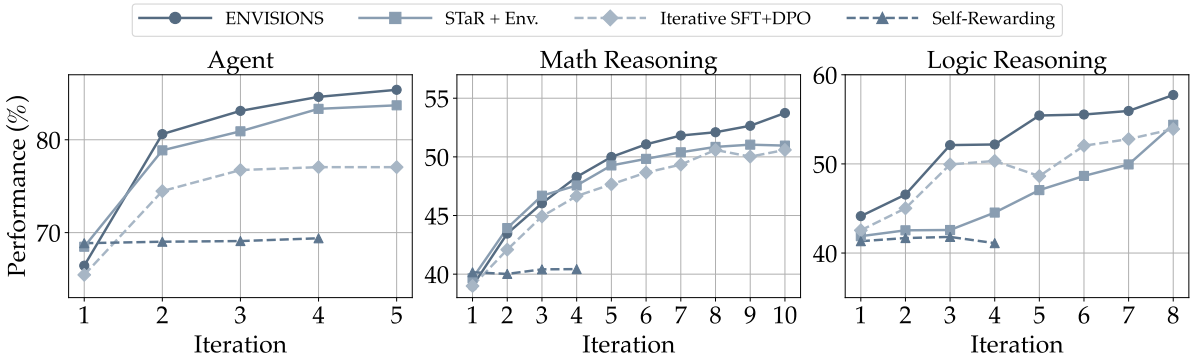
Table 6: Averaged performances on MiniWob++ benchmark. All these methods are based on Mistral-Instruct-v0.2 (7B) model.

C.4 Results on latest powerful LLM

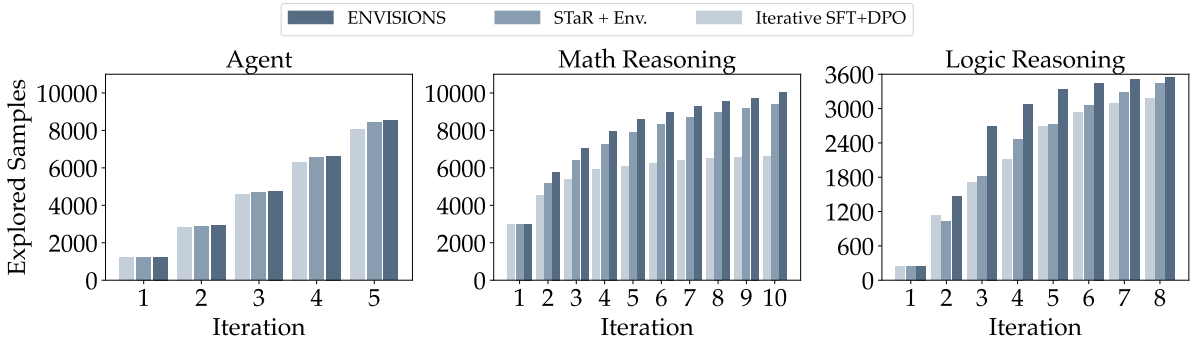
To verify the effectiveness and generalization capability of ENVISIONS, we supplement the implementation of ENVISIONS on the latest open-source LLM - LLaMA3.1 in Table 7.

From the results, **ENVISIONS still works for the powerful LLaMA3.1-8B** model in the mathematical reasoning tasks. It improves performances by large margins and presents superiority over strong baselines.

It is observed that some previous SOTA methods fail to generalize well to LLaMA3.1. It is worth noting that current trending LLMs (e.g., LLaMA3.1) may be widely contaminated by the



(a) Evolution of performance.



(b) Evolution of explored sample numbers.

Figure 6: Evolution curves on LLaMA2-Chat 7B version across agent, math, and logic reasoning domains. (a) is the performance evolution progress. (b) denotes the evolution of explored sample numbers.

	GSM8K	MATH	GSM-H	Avg.
Is Held-out ?	✗	✗	✓	-
LLaMA3.1-Chat (8B)				
Few-shot	60.27	39.57	50.95	50.26
Distill GPT-4	63.08	38.47	53.07	51.54
ETO	66.64	39.29	59.06	55.00
STaR + Env.	69.60	36.69	66.11	57.47
ENVISIONS	**83.32**	**42.13**	**69.14**	**64.86**

Table 7: Performances on LLaMA3.1.

	GSM8K	MATH	GSM-H	SVAMP	AsDiv
iterative SFT+DPO	54.81	14.75	47.08	70.10	66.22
+Self-refine	55.11	14.82	47.38	71.10	66.36
STaR+Env.	58.23	18.82	48.45	67.50	68.46
+Self-refine	58.30	18.87	48.52	67.60	68.51
ENVISIONS	58.98	19.00	48.52	72.40	69.80
+Self-refine	**60.65**	**19.70**	**49.81**	**73.60**	**70.61**

Table 8: Inference-time optimization. We apply self-refinement strategy to the self-training methods and report the performances on five mathematical datasets.

training corpus (e.g., GSM8K) or have been exposed to similar training corpus. That is why we chose to experiment on the Llemma or Mistral base model to evaluate its generalization capability in the original manuscript.

C.5 Inference-Time Optimization

One of the unique advantages of RL-free loss is the inference-time self-refinement, which can be obtained with traditional contrastive learning works. Table 8 presents the results.

The experimental results show that ENVISIONS can benefit a lot from the optimization of self-refinement (RL-free loss), while other baselines can hardly conduct effective self-refinement.

C.6 Detailed Analysis From Three Views

In the section 5.2 of the main paper, we make a analysis on *What is behind the superiority of ENVISIONS*. We present the analysis with LLaMA2-Chat (7B) model as backbone from three distinctive views: (1) exploratory ability and stability; (2) log probability margin between positive and negative solutions; and (3) diversity of synthetic samples. Here, we supplement the results on the LLaMA2-Chat (13B) model. The main findings are consistent with the 7B model:

Balanced exploratory ability and stability are key to success in weak-to-strong. We employ

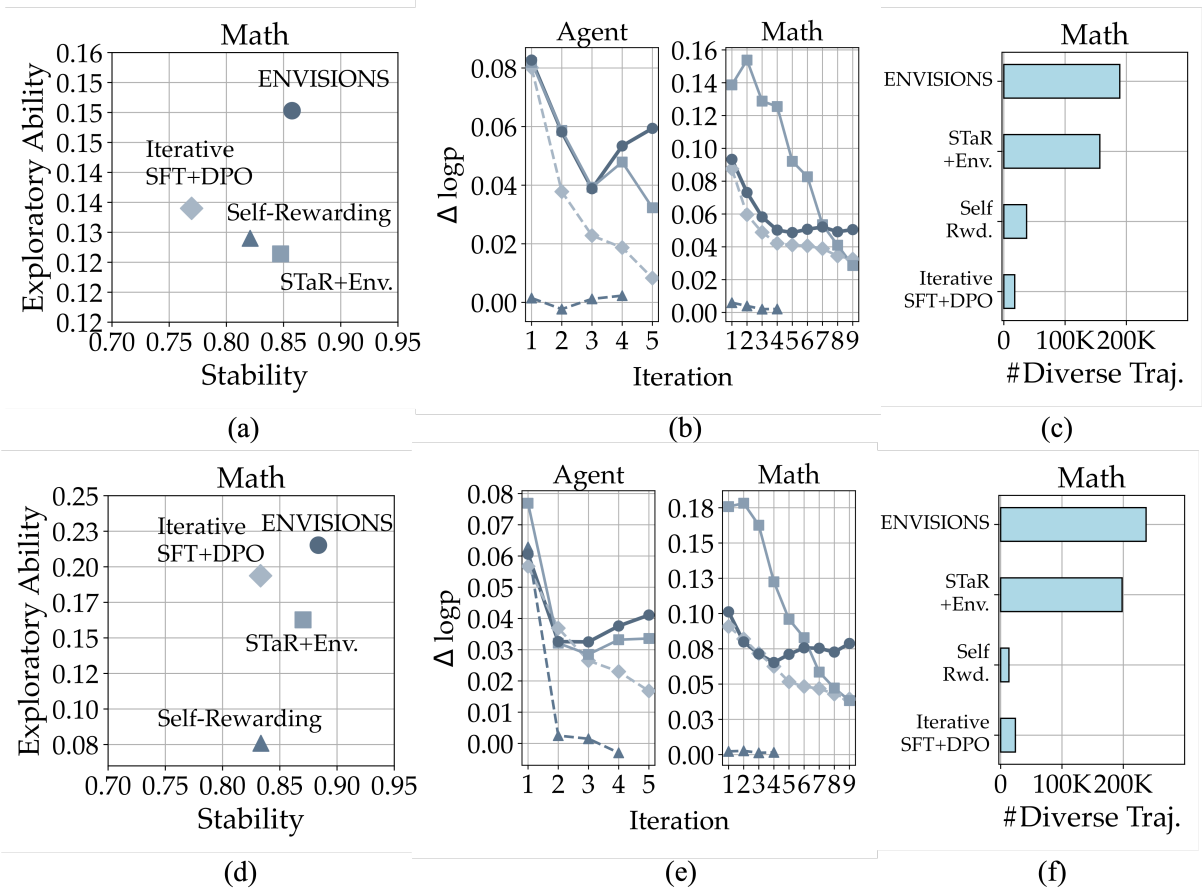


Figure 7: In-depth analysis from three perspectives. The first row (i.e., (a),(b),(c)) and the second row (i.e., (d),(e),(f)) represent the results on LLaMA2-Chat (7B) and LLaMA2-Chat (13B) respectively. Plots in fig.(b),(e) correspond to the methods represented by the same colors in fig.(a),(d).

two metrics *exploratory ability* and *stability* to evaluate the LLM (both of them are the higher, the better). Appendix B gives definition details. In both Fig. 7(a) and (d), *ENVISIONS* demonstrates remarkable performance in achieving a balance between exploratory ability and stability. By leveraging the candidate pool and self-rewards, *ENVISIONS* effectively retains high-quality positive solutions during training, significantly mitigating the issue of forgetting previous trajectories. Notably, reinforced self-training methods consistently exhibit unstable performance.

Clearly distinguishing positive and negative solutions can help the LLM optimization. Clearly keeping the probability margins (Δlogp) between positive-negative pairs is crucial to facilitate the optimization. Fig. 7 (b) and (e) shows the analysis of Δlogp during iterations. It is observed *ENVISIONS* keeps the margin within a reasonable range, while reinforced methods exhibit a rapid decrease in Δlogp. It indicates the unsuitablity of DPO to the exploration setting and the importance of feedback from **Env**. Such finding corresponds to the lack of exploratory ability in Fig. 7(a) and (d).

Diverse trajectories are what you need for self-training. In Fig. 7 (c) and (f), we compare the number of correct and unique trajectories by the last iteration. It demonstrates the huge strengths of *ENVISIONS* in synthesizing diverse trajectories. It largely surpasses *Reinforced Self-Training* approaches. Notably, LLM updates in RL methods are restricted by *KL* constraints, which ultimately impact the diversity of the generated trajectories. Moreover, *Distill GPT-4* and *Distill Claude2* lead to 10,831 and 8,561 diverse trajectories. Since repeatedly calling strong LLMs involves extremely high costs, they are far from sustainable compared with *ENVISIONS*.

C.7 How does the Training Recipe Matter in Iterative Self-Exploration?

In each iteration of *ENVISIONS*, we optimize the policy LLM from scratch (e.g., LLaMA2-Chat) with the updated training trajectories. Such a training recipe is expected to bring stability to the training process, compared with the strategy of contin-

Tasks	Cont.	*ENVISIONS*	Δ
LLaMA-2-Chat (7B)			
Agent	78.18	85.38	+7.20
Math Reasoning	51.20	53.74	+2.54
Logic Reasoning	46.20	57.73	+11.53
Average	53.32	58.69	+5.37

Table 9: Comparisons between training strategies. *Cont.* column denotes the performances of *ENVISIONS* under the continual training setting.

uous training based on previous checkpoints. Table. 9 presents the performance comparisons. Obvious superiority of *ENVISIONS* is observed across these three domains, with an average improvement of 5.37%. Training from previous checkpoints does affect the exploration. For the RL-based self-training method, the training of the policy LLM is constrained within the range of the reference model by the KL term. In order to enable continuous evolution, the policy LLM is required to be updated from the checkpoint of the previous iteration. It is also one of the main causes of their sub-optimal performances.

C.8 MiniWob++ Results Per Tasks

Table 11 shows the performance of *ENVISIONS* on each of the 44 MiniWob++ tasks.

C.9 Comparison with more SOTA baselines

To better verify the effectiveness of ENVISIONS, we supplement one more SOTA baseline ETO (Song et al., 2024)for comparison. The results show consistent superiority in math reasoning tasks.

Tasks	GSM8K	GSM-H	MATH	SVAMP	AsDiv	Avg.
LLaMA-2-Chat (7B)						
ETO	50.04	15.75	45.49	68.10	65.36	48.95
ENVISIONS	58.98	19.00	48.52	72.40	69.80	53.74

Table 10: Comparisons with another SOTA baseline ETO on math reasoning tasks.

D Pseudocode of *ENVISIONS*

The self-training framework *ENVISIONS* can be expressed in Algorithm 1.

E Prompt of Self-Refinement

We provide the prompt for the self-refinement. Below is an example of the math reasoning task.

[INPUT]
You are provided with a Python code to solve the given problem. You can either repair and refine it, or simply return the original solution.
The question is:
<question>
The current Python code is:
<negative solution>
The solution code is:
[OUTPUT]
<positive solution>

	1-shot	Distill GPT4	Distill Claude2	STAR+Env.	Self-Rewarding	iter. SFT+DPO	Ours
LLaMA-2-Chat (7B)							
choose-date	0.00	0.00	0.00	0.00	0.00	0.00	0.00
choose-list	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-button	0.00	100.00	100.00	100.00	100.00	96.67	100.00
click-button-sequence	100.00	100.00	100.00	100.00	100.00	100.00	96.67
click-checkboxes	20.00	100.00	96.67	100.00	100.00	100.00	100.00
click-checkboxes-large	20.00	86.67	96.67	86.67	66.67	100.00	100.00
click-checkboxes-soft	0.00	6.67	30.00	50.00	0.00	63.33	76.67
click-checkboxes-transfer	56.67	100.00	100.00	100.00	100.00	100.00	100.00
click-collapsible	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-color	53.33	100.00	100.00	100.00	100.00	100.00	100.00
click-dialog	100.00	100.00	100.00	100.00	0.00	100.00	100.00
click-dialog-2	0.00	26.67	73.33	100.00	73.33	100.00	100.00
click-link	73.33	93.33	93.33	93.33	93.33	93.33	93.33
click-option	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-scroll-list	56.67	100.00	100.00	100.00	96.67	100.00	100.00
click-shades	93.33	100.00	100.00	100.00	100.00	100.00	100.00
click-shape	0.00	70.00	53.33	63.33	16.67	50.00	70.00
click-tab	100.00	100.00	100.00	96.67	26.67	56.67	100.00
click-test	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-test-2	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-widget	96.67	100.00	100.00	100.00	100.00	100.00	100.00
copy-paste	100.00	100.00	100.00	100.00	100.00	100.00	100.00
copy-paste-2	100.00	100.00	100.00	100.00	100.00	100.00	100.00
enter-date	3.33	100.00	100.00	100.00	100.00	100.00	100.00
enter-password	96.67	100.00	100.00	100.00	100.00	100.00	100.00
enter-text	100.00	100.00	100.00	100.00	100.00	100.00	100.00
enter-text-dynamic	100.00	100.00	100.00	100.00	100.00	100.00	100.00
enter-time	0.00	30.00	0.00	0.00	0.00	0.00	43.33
focus-text	100.00	100.00	100.00	100.00	100.00	100.00	100.00
focus-text-2	33.33	100.00	100.00	100.00	100.00	100.00	100.00
guess-number	6.67	0.00	6.67	10.00	6.67	10.00	10.00
identify-shape	0.00	56.67	80.00	100.00	56.67	50.00	100.00
multi-layouts	3.33	96.67	86.67	100.00	76.67	96.67	100.00
multi-orderings	0.00	93.33	100.00	100.00	80.00	100.00	100.00
navigate-tree	60.00	60.00	60.00	60.00	60.00	60.00	60.00
read-table	70.00	100.00	100.00	100.00	100.00	100.00	100.00
search-engine	3.33	100.00	100.00	100.00	43.33	0.00	100.00
simple-algebra	6.67	50.00	63.33	80.00	6.67	3.33	73.33
simple-arithmetic	0.00	86.67	90.00	100.00	40.00	73.33	96.67
social-media-all	30.00	100.00	100.00	30.00	0.00	30.00	30.00
text-transform	66.67	100.00	100.00	100.00	100.00	100.00	100.00
unicode-test	100.00	100.00	100.00	100.00	100.00	100.00	100.00
use-slider	0.00	6.67	6.67	6.67	6.67	6.67	6.67
use-spinner	0.00	6.67	6.67	6.67	6.67	0.00	0.00
Average	51.14	81.14	82.80	83.71	69.47	77.05	**85.38**
LLaMA-2-Chat (13B)							
choose-date	0.00	0.00	0.00	0.00	0.00	0.00	0.00
choose-list	96.67	100.00	100.00	100.00	100.00	100.00	100.00
click-button	96.67	100.00	100.00	100.00	100.00	96.67	100.00
click-button-sequence	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-checkboxes	30.00	100.00	100.00	100.00	100.00	100.00	100.00
click-checkboxes-large	26.67	86.67	96.67	90.00	43.33	90.00	93.33
click-checkboxes-soft	0.00	3.33	46.67	90.00	20.00	60.00	90.00
click-checkboxes-transfer	10.00	100.00	96.67	100.00	100.00	100.00	100.00
click-collapsible	100.00	100.00	96.67	100.00	100.00	100.00	100.00
click-color	56.67	100.00	100.00	100.00	100.00	100.00	100.00
click-dialog	100.00	100.00	100.00	100.00	0.00	100.00	100.00
click-dialog-2	0.00	26.67	73.33	100.00	73.33	100.00	100.00
click-link	70.00	93.33	93.33	93.33	93.33	93.33	93.33
click-option	0.00	100.00	100.00	100.00	100.00	100.00	100.00
click-scroll-list	63.33	100.00	100.00	100.00	100.00	100.00	100.00
click-shades	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-shape	10.00	76.67	56.67	86.67	10.00	66.67	73.33
click-tab	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-test	100.00	100.00	96.67	100.00	100.00	100.00	100.00
click-test-2	100.00	100.00	100.00	100.00	100.00	100.00	100.00
click-widget	96.67	100.00	100.00	100.00	100.00	100.00	100.00
copy-paste	100.00	100.00	100.00	100.00	100.00	100.00	100.00
copy-paste-2	100.00	100.00	100.00	100.00	100.00	100.00	96.67
enter-date	100.00	100.00	100.00	100.00	100.00	100.00	100.00
enter-password	100.00	96.67	100.00	100.00	100.00	100.00	100.00
enter-text	100.00	100.00	100.00	100.00	100.00	100.00	100.00
enter-text-dynamic	100.00	100.00	100.00	100.00	100.00	100.00	100.00
enter-time	0.00	0.00	23.33	0.00	0.00	0.00	93.33
focus-text	0.00	100.00	100.00	100.00	96.67	100.00	100.00
focus-text-2	63.33	100.00	100.00	100.00	100.00	100.00	100.00
guess-number	6.67	0.00	6.67	3.33	6.67	10.00	6.67
identify-shape	0.00	10.00	90.00	100.00	20.00	90.00	100.00
multi-layouts	66.67	100.00	100.00	100.00	86.67	96.67	100.00
multi-orderings	56.67	100.00	100.00	100.00	100.00	100.00	100.00
navigate-tree	60.00	60.00	60.00	60.00	60.00	60.00	56.67
read-table	76.67	100.00	100.00	100.00	100.00	100.00	100.00
search-engine	90.00	100.00	100.00	100.00	100.00	100.00	100.00
simple-algebra	23.33	76.67	80.00	80.00	43.33	36.67	83.33
simple-arithmetic	56.67	100.00	100.00	100.00	100.00	96.67	100.00
social-media-all	93.33	100.00	100.00	30.00	0.00	30.00	30.00
text-transform	96.67	83.33	100.00	100.00	100.00	100.00	100.00
unicode-test	93.33	100.00	100.00	100.00	100.00	100.00	100.00
use-slider	0.00	6.67	6.67	6.67	6.67	6.67	10.00
use-spinner	0.00	6.67	6.67	6.67	6.67	6.67	6.67
Average	60.00	80.15	84.77	85.15	74.24	82.73	**87.12**

Table 11: Detailed performances on 44 MiniWob++ tasks.

Algorithm 1: A Neural-Symbolic Self-Training Framework *ENVISIONS*

Input: Data pair $\{(x, y)\}$, environment **ENV**, candidate trajectory pool POOL, weak LLM π_{θ_0}, number of generated samples K, number of iteration T.

Output: Strong LLM π_θ^*.

```
// Initialize
```

$\pi_\theta \leftarrow \pi_{\theta_0}$

```
// Start the Loop
```

for $i = 1$ **to** T **do**

  for *each* x *in the input* **do**

    `// 1-Online Exploration`

    Generate K symbolic solutions with self-rewards: $\{a_k\}_{k=1}^K, \{r_k\}_{k=1}^K \sim \pi_\theta(\cdot|x)$.

    Get binary rewards by executing in **ENV**: $\{b_k\}_{k=1}^K \longleftarrow \mathbb{I}[\textbf{ENV}(a_k) == y]$.

    Generate self-refined solutions with self-rewards: $\{\widetilde{a}_k\}_{k=1}^K, \{\widetilde{r}_k\}_{k=1}^K \sim \pi_\theta(\cdot|x; a_k)$.

    Get binary rewards by executing in **ENV**: $\{\widetilde{b}_k\}_{k=1}^K \longleftarrow \mathbb{I}[\textbf{ENV}(\widetilde{a}_k) == y]$.

    Let $T_k = (x, y, a_k, b_k, r_k), \widetilde{T}_k = (x, y, \widetilde{a}_k, \widetilde{b}_k, \widetilde{r}_k)$ denote the collected trajectories.

    `// 2-Traj. Filtering and Candidate Pool Updating`

    Filter the superior trajectory T_k^* from T_k and $\widetilde{T}_k$ with binary rewards and self-rewards.

    Update the candidate pool with T_k^*.

  end

  `// 3-Training`

  Rank and retrieve positive-only training set U_1 and positive-negative pairs U_2 from POOL.

  Optimize π_{θ_0} to π_θ^* with $\mathcal{L} = -\sum_{(x,a^+)\sim U_1} \log p_{\theta_0}(a^+|x) - \sum_{(x,a^+,a^-)\sim U_2} \log p_{\theta_0}(a^+|x; a^-)$.

  Update the policy LLM for the next iteration: $\pi_\theta \leftarrow \pi_\theta^*$

end

```
// Output the enhanced LLM
```

Return π_θ^*;