Genius: A Generalizable and Purely Unsupervised Self-Training Framework For Advanced Reasoning

Fangzhi Xu[1,2*] **Hang Yan**[2] **Chang Ma**[3] **Haiteng Zhao**[4] **Qiushi Sun**[1,3]
Kanzhi Cheng[1] **Junxian He**[5] **Jun Liu**[2†] **Zhiyong Wu**[1†]
[1]Shanghai AI Lab [2]Xi'an Jiaotong University [3]The University of Hong Kong
[4]Peking University [5]Hong Kong University of Science and Technology
`fangzhixu98@gmail.com`

Abstract

Advancing LLM reasoning skills has captivated wide interest. However, current post-training techniques rely heavily on supervisory signals, such as outcome supervision or auxiliary reward models, which face the problem of scalability and high annotation costs. This motivates us to *enhance LLM reasoning without the need for external supervision*. We introduce a generalizable and purely unsupervised self-training framework, named *Genius*. Without external auxiliary, *Genius* requires to seek the optimal response sequence in a stepwise manner and optimize the LLM. To explore the potential steps and exploit the optimal ones, *Genius* introduces a stepwise foresight re-sampling strategy to sample and estimate the step value by simulating future outcomes. Further, we recognize that the unsupervised setting inevitably induces the intrinsic noise and uncertainty. To provide a robust optimization, we propose an advantage-calibrated optimization (ACO) loss function to mitigate estimation inconsistencies. Combining these techniques together, *Genius* provides an advanced initial step towards self-improve LLM reasoning with general queries and without supervision, revolutionizing reasoning scaling laws given the vast availability of general queries.

1 Introduction

Reasoning skills are crucial for Large Language Models (LLMs) to achieve human-level intelligence (Achiam et al., 2023; Team et al., 2023). Recent efforts (Qwen, 2024; Guo et al., 2025) focus on enhancing reasoning capabilities during the post-training phase. Typically, existing methods rely on supervision, which can be divided into two main approaches. One stream is supervised fine-

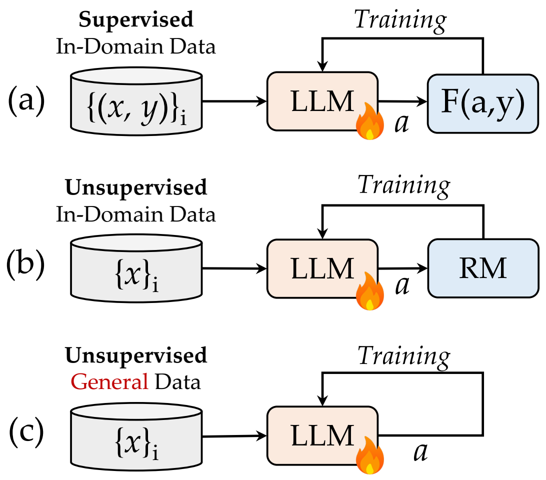


Figure 1: Typical reinforce-like approaches to boost LLM reasoning. (a) and (b) abstract two types of reinforce-like methods, which require final answer and verification respectively. (c) depicts the ultimate goal of our research. x denotes the input query, a denotes the LLM-generated responses that contain multiple steps, and y is the final answer (if exists).

tuning (SFT) that requires the well-annotated response (a) paired with the query (x). The other line of work is reinforce-like approaches that require either ground-truth answer or verification. The former needs rule-based matching of the answer, as depicted in Fig. 1(a). Though they are effective in specific domains like math and coding, many other problems lack clear solutions or explicit ground truth, presenting challenges for generalization to broader reasoning tasks. The latter utilizes the external reward model for verification. However, the training of a generalized reward model relies on expensive annotation (Lightman et al., 2023) and it may induce the reward hacking problem (Amodei et al., 2016; Gao et al., 2023). Considering these limitations of scalability, it would revolutionize reasoning scaling law if we could achieve effective and efficient post-training simply from *general queries*.

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

Therefore, it motivates us to raise a research question of ***How to advance LLM reasoning ability without any external supervision ?*** (Fig. 1(c)).

This work tackles this problem by proposing a generalizable self-training framework, named *Genius*. As depicted in Fig. 1(c), *Genius* only requires the policy LLM itself with a set of unsupervised queries. Without the external auxiliary, it builds upon the self-training paradigm that the LLM first generates response a given the input query x, and selects the optimal ones for training. As an initial attempt, *Genius* paves the way to self-improve LLM reasoning with unsupervised general queries.

To generate training data for self-training, a crucial challenge is to determine *how to collect and self-reward LLM responses without relying on external auxiliary resources.* One intuitive solution is to have the LLM generate the entire response and then evaluate the response based on its sequence confidence (Xu et al., 2024a), re-prompt the model to assign scores (Yuan et al., 2024), or apply self-consistency methods (Prasad et al., 2024). However, these attempts primarily focus on response-level rewards, while step-wise supervision could provide more stable and fine-grained training (Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2024a). Recent efforts like (Zhang et al., 2024) employ step-level sampling, but they naturally inherit the short-sighted limitation of auto-regressive generation, lacking global adherence to overarching goals (Ma et al., 2024). Search-based rewarding (e.g., MCTS-style methods) provides more global-awareness but requires intricate backtracking processes, which are notoriously time-consuming.

To this end, *Genius* seeks the optimal response sequence via stepwise sampling and collects high-quality preference pairs with rewards. Without reliance on external supervision to obtain the global-awareness, *Genius* adopts a coarse step estimation by simply rolling out future steps (referred to as *foresight*) and using their uncertainty as the foresight score to evaluate candidate steps. Based on this technique, we propose the stepwise foresight re-sampling approach: using foresight scores to approximate the distribution that is sampled to determine the next step (for exploration) and re-sampled to create step-level preference pairs (for exploitation).

Although the above approach offers a quality-efficiency balanced solution for superior *sample-and-reward*, calculating the distribution of foresight scores based on a few rollouts may results in a biased estimation of step values, inevitably inducing noise to the self-supervision labels. However, previous self-training methods simply employ either supervised fine-tuning (Zelikman et al., 2022) or reinforced learning strategies (Yuan et al., 2024) for optimization, neglecting the uncertainty of self-supervision (Liang et al., 2024). We tackles this second challenge – *improving the robustness of self-training optimization* – by introducing an advantage-calibrated loss function (ACO), which penalizes inconsistent estimation between foresight score and step advantage. We find that ACO, compared to SFT (Zelikman et al., 2022), and DPO (Rafailov et al., 2024) improves training stability and boosts performance.

Genius offers a unique perspective on post-training: LLMs can self-improve their general reasoning abilities using general queries without any form of external supervision. With merely 25K unsupervised general queries, *Genius* surprisingly improves the average performance across diverse reasoning benchmarks by >7%. We also demonstrate that the scaling law on general queries consistently improves with more training steps (see § 4.2). Given the abundance of general data available, this scaling can significantly enhance reasoning capabilities and further pushes the boundary of reasoning scaling law (Guo et al., 2025).

2 Methodology

2.1 Preliminary

One major advantage of *Genius* is that it only requires unsupervised natural language (NL) queries as inputs. Under the self-training setting, the LLM π_θ generates responses given the query, and then select the optimal ones to optimize itself. The main objectives of *Genius* are divided into two parts: (1) synthesizing and rewarding the responses (§ 2.2); (2) optimizing the LLM with responses (§ 2.3). Figure 2 presents the overall framework of *Genius*.

To achieve the first objective, *Genius* models the response as a sequence of steps. The globally optimal response is derived by stepwise sampling and rewarding. At each timestamp, *Genius* first rollouts a set of candidate steps and self-rewards them via simulating future steps (Fig. 2(a)). Then, we select the optimal next step (Fig. 2(b)) and collect preference pairs for training (Fig. 2(c)). As for the second objective, *Genius* derives the step advantages during the sampling (Fig. 2(d)), and

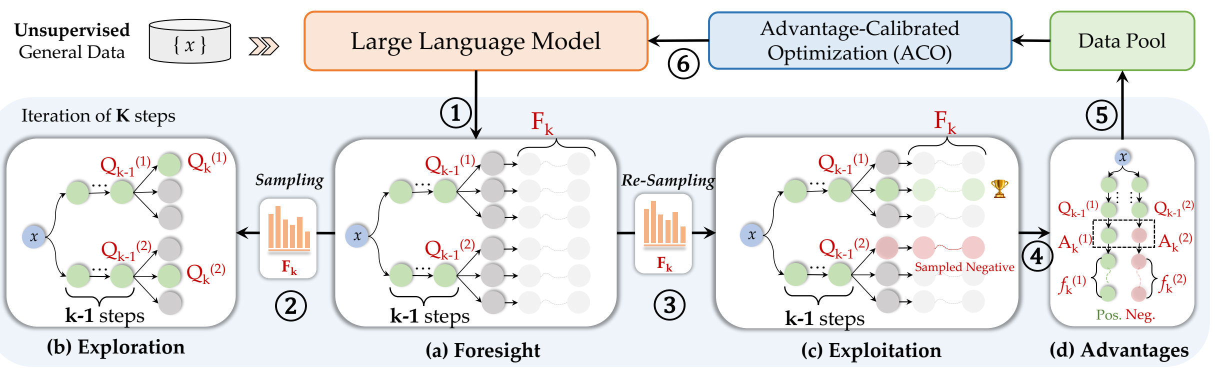


Figure 2: The overall framework of *Genius*. It only receives the unsupervised NL queries as inputs. To complete goal of self-improving, the policy LLM goes through K steps of sampling and rewarding for each query (Step 1-4), collects the high-quality response sequence as the training data (Step 5), and trains itself with the advantage calibrated optimization loss (Step 6).

adopts them to optimize the LLM with designed reinforced loss.

For clearer illustration, we introduce the following symbols at the timestamp k: a_k denotes the current step with the step value of Q_k. $\mathbf{a}_{<k}$ represents the preceding steps while $\mathbf{a'}_{>k}$ is the simulated future steps. $T_k = (\mathbf{a}_{<k}, a_k, \mathbf{a'}_{>k})$ denotes the curated response for training. For simplicity, we omit the symbol index in some descriptions.

2.2 Exploration and Exploitation via Foresight Re-Sampling

To ensure the diversity, we use beam search strategy (Freitag and Al-Onaizan, 2017) during the step-wise sampling. We define the step beam size as M, as illustrated in Fig. 2 where we plot a simple case of $M = 2$.

Step Rollouts with Foresight. At Step $k-1$, *Genius* keeps M preceding paths $\mathbf{a}_{<k}$, each consisting of $k-1$ steps. The value of the last step in the path is defined as $Q_{k-1}^{(m)}$, where $m \in [1, M]$. For each beam m, *Genius* first rollouts N candidate steps a_k, leading to $M * N$ candidate steps totally.

To address the limitation of auto-regressive generation and construct the globally-aware response, *Genius* performs the simulation of future steps based on each candidate step a_k. We refer to this process as *foresight* (Fig. 2(a)). This allows us to derive the response sequence along with their respective foresight scores, calculated using the averaged log probability of the remaining steps:

$$\mathbf{a}'_{>k}, f_k \sim \pi_\theta(\cdot|\mathbf{a}_{<k}; a_k). \tag{1}$$

With these simulated future steps $\mathbf{a}'_{>k}$, the completed response can be constructed, written as $T = (\mathbf{a}_{<k}, a_k, \mathbf{a}'_{>k})$. Here we obtain $M * N$ foresight scores f_k. They can be utilized to approximate a distribution $\mathbf{F}_k$. After normalization, the elements of $\mathbf{F}_k$ are given by:

$$\mathbf{F}_k(i) = \exp(f_k^{(i)}/\tau)/\sum_j \exp(f_k^{(j)}/\tau) \tag{2}$$

In this expression, $\mathbf{F}_k(i)$ denotes the foresight score at index i, where $i \in [1, M*N]$. τ is the temperature parameter used to control the sharpness of the distribution.

Re-Sampling for Exploration and Exploitation. Based on the foresight technique, *Genius* further selects the steps $a_k^{(m)}$ for current timestamp k via sampling on the distribution of $\mathbf{F}_k$ (Fig. 2(b)):

$$\{a_k^{(m)}\}_{m=1}^{M} \sim \text{Categorical}(\mathbf{F}_k). \tag{3}$$

In this way, we can keep M beams for exploration in the next step. Here, we define the Q value of each selected step $a_k^{(m)}$ with the foresight score:

$$Q_k^{(m)} := f_k^{(m)} \tag{4}$$

Besides the exploration, *Genius* also exploits the entire response sequence $T_k = (\mathbf{a}_{<k}, a_k, \mathbf{a}'_{>k})$ at each timestamp k for optimization (Fig. 2(c)). To encourage diversity and avoid overfitting on similar responses, we introduce the re-sampling strategy based on the distribution $\mathbf{F}_k$. The response with the highest foresight score f_k^w is chosen as the positive one, written as T_k^w. The negative response is re-sampled from $\mathbf{F}^{(k)}$:

$$T_k^l \sim \text{Categorical}(\mathbf{F}_k /\ f_k^w) \tag{5}$$

The corresponding foresight score of the negative path is f_k^l. With such a re-sampling strategy, the balance between *exploration* and *exploitation* can be achieved.

Advantages and Data Construction. Since the reasoning sequences are completed from different beams, it is insufficient to simply evaluate each step with the foresight score f_k. Therefore, *Genius* derives the advantage value A_k for both positive and negative response sequences:

$$A_k^w = f_k^w - Q_{k-1}^w, \quad A_k^l = f_k^l - Q_{k-1}^l \tag{6}$$

From the equation, the foresight score is calibrated with the Q value of the previous step.

In this way, the training preference pair obtained from each step k is constructed in the quintuple format, i.e., $(x, T_k^w, A_k^w, T_k^l, A_k^l)$.

2.3 Advantage-Calibrated Optimization

Given the constructed preference pairs, we can optimize the LLMs through reinforced learning. There remains two critical steps unaddressed: (i) formulating the self-rewards for preference optimization; and (ii) deriving the optimization objective.

Formulating Self-Rewards as Preferences. Building upon Bradley-Terry model (Bradley and Terry, 1952), the measurement of the preferences can be formulated as:

$$\begin{aligned} p^*(T^w \succ T^l|x) &= \frac{\exp(r^*(x,T^w))}{\exp(r^*(x,T^l)) + \exp(r^*(x,T^l))} \\ &= \sigma(r^*(x,T^w) - r^*(x,T^l)) \end{aligned} \tag{7}$$

where $r^*(T|x)$ represents the optimal reward function. $\sigma(\cdot) = 1/(1+\exp(-x))$ denotes the sigmoid function. Based on this modeling, the well-trained reward model r_ψ is required to further optimize the LLM policy via RL fine-tuning. However, under our unsupervised setting, training the reward model becomes impractical.

In the context of DPO (Rafailov et al., 2024), the policy LLM π_θ is leveraged as the implicit reward model. The self-reward function ϕ is modeled as:

$$\begin{aligned} \phi(x,T) &= \beta \log \frac{\pi_\theta(T|x)}{\pi_{\text{ref}}(T|x)} + \beta \log Z(x) \\ &\propto \beta \log \frac{\pi_\theta(T|x)}{\pi_{\text{ref}}(T|x)} \end{aligned} \tag{8}$$

where π_{ref} denotes the reference model and $Z(x) = \sum_T \pi_{\text{ref}}(T|x)\exp(\phi(x,T)/\beta)$ represents the partition function. With this approximation, the standard rewards for both the positive and negative response sequences can be derived:

$$\phi_w = \beta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)}, \ \phi_l = \beta \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)} \tag{9}$$

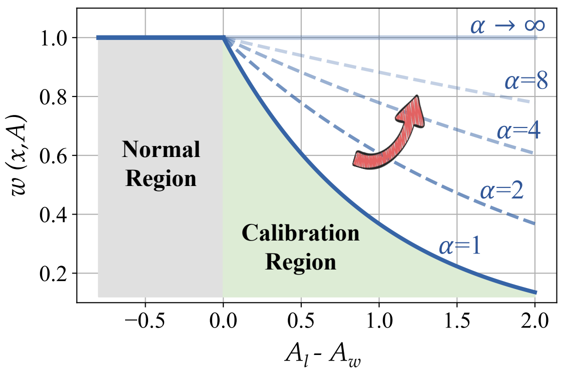


Figure 3: Visualization of the calibration function. The x-axis denotes the the differences between A_l and A_w, while the y-axis is the value of the calibration term. By adjusting α, we can control the decay rate of the curve.

ACO Loss Function. Under the unsupervised setting, the training pairs are sampled based on the distribution of foresight score. It would induce the noise during the optimization. The above formulation of self-rewards treats each preference pair with equal scale, making it difficult to detect the exception and improve the robustness. Therefore, we propose to employ the calculated advantage value A to calibrate the self-reward function ϕ. In detail, we add the relaxation term $w(x, A)$ for the self-rewards of the negative response sequence:

$$\phi_l(x, T^l) = \beta w(x, A) \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}, \tag{10}$$

$$w(x, A) = \text{clip}\left(\exp \frac{-(A^l - A^w)}{\alpha}, 1\right) \tag{11}$$

where $A_l - A_w$ denotes the differences in the advantages brought by the negative steps and positive steps. α is the hyper-parameter to control the scale of the relaxation term.

For better understanding, we visualize $w(x, A)$ in Figure 3. This function can be categorized into two distinct regions: the *Normal Region* and the *Calibration Region*. In the *Normal Region*, where $A_l - A_w \leq 0$, the negative response sequence is distinguishable from the positive one. Conversely, when $A_l - A_w > 0$, the *Calibration Region* is engaged, offering increased relaxation to the negative sample. Through the adjustment of the hyperparameter α, we can regulate the extent of this flexibility. In short, if the negative response sequence provides more actual advantages than the positive one, then it will be less punished (with smaller weight in the self-reward calculation).

Substituting the self-reward function ϕ_w and ϕ_l into Eq. 7 and optimizing it using the form of negative log-likelihood, the ACO loss is derived:

$$\mathcal{L}_{\text{ACO}} = -\mathbb{E}_{(x,T_w,T_l)\sim\mathcal{D}} \log\sigma \Bigg[\beta \log \frac{\pi_\theta(T_w|x)}{\pi_{\text{ref}}(T_w|x)} - \beta\text{clip}\left(\exp\frac{-(A_l - A_w)}{\alpha}, 1\right) \log \frac{\pi_\theta(T_l|x)}{\pi_{\text{ref}}(T_l|x)}\Bigg] \tag{12}$$

We include the gradient analysis of ACO loss function and its relations to other robust preference optimization strategies in Appendix A.

3 Experiments

3.1 Implementation Details

Training Corpora The training queries are respectively sourced from two general corpora, Magpie (Xu et al., 2024b) and OpenHermes-2.5 (Teknium, 2023). Considering the computational cost, we opt to randomly select 25K queries from Magpie and 32K queries from OpenHermes-2.5. They are utilized respectively as the sources for self-training.

Evaluation Tasks To comprehensively evaluate the basic reasoning abilities of the LLMs, we incorporate the following benchmarks: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and GPQA (Rein et al., 2023) for math reasoning, ReClor (Yu et al., 2020) and LogiQA (Liu et al., 2021) for logical reasoning, StrategyQA (Geva et al., 2021) and ARC-Challenge (Clark et al., 2018) for general reasoning. Moreover, we also include the some general benchmarks to verify the performance stability on the general domain: AlpacaEval (Li et al., 2023), WildBench (Lin et al., 2024), and ArenaHard (Li et al., 2024) for subjective evaluation, WikiBench (Kuo et al., 2024), MMLU (Hendrycks et al., 2020), and MMLU-Pro (Wang et al., 2024b) for objective evaluation. Also, the competition-level benchmark AIME2024 (Veeraboina, 2023) is included to prove the scalability.

Baselines One line of baselines requires supervision (labeled response) beyond the training queries, including *SFT* and *SPIN* (Chen et al., 2024). Another line of methods only needs unsupervised queries as the input, covering *STaR* (Zelikman et al., 2022), *CoH* (Liu et al., 2024), *Self-Rewarding* (Yuan et al., 2024) and *ScPO* (Prasad et al., 2024). Details refer to Appendix B.1.

Base LLMs In the main experiments, we utilize LLaMA3.1-8B-Instruct (Dubey et al., 2024) as the backbone. To verify the generalization capability, we also apply the self-training approaches on Qwen2.5-Instruct series models (Yang et al., 2024), including 3B and 7B variants.

Training and Inference Setup For the foresight sampling configuration, we set M=2, N=4, and K=4. Based on it, the total number of training pairs is 100K and 128K for Magpie and OpenHermes2.5 respectively. The inference process is accelerated by the vLLM engine. Setup details refer to Appendix B.2.

3.2 Main Results

In Table 1, we present the evaluation results. In addition to presenting individual results for each dataset, the average performances are also reported in the last column.

***Genius* significantly boosts the reasoning abilities of LLaMA3.1, surpassing all strong baselines.** With merely 25K unsupervised training queries (i.e., self-training from Magpie), *Genius* demonstrates a notable enhancement in the average CoT reasoning performance of LLaMA3.1-8B-Instruct by 7.43%. This improvement is further underscored when using OpenHermes as the training corpus. In comparison to strong baselines, *Genius* consistently exhibits state-of-the-art performance, showcasing an average advantage of >2%. Among them, *Genius* presents obvious more advantages in challenging tasks (e.g., MATH), outperforming *Self-Rewarding* by >4%. Moreover, the superiority of *Genius* is consistent across all the evaluation benchmarks, while other baselines (e.g., CoH and SPIN) show deviations in performances.

Self-training with RL optimization is more effective in improving reasoning with general data. Among the baselines, RL-based self-training methods (e.g., *CoH*, *Self-Rewarding* and *ScPO*), exhibit consistent advantages over supervised fine-tuning baselines (e.g., SFT and STaR). Since cur-

Models	GSM8K	MATH	ReClor	LogiQA	StrategyQA	GPQA	ARC-c	Avg.
LLaMA3.1-8B-Instruct (CoT)	70.28	30.52	49.40	33.33	58.91	26.56	78.33	49.65
Self-Training from **Magpie (25K)**								
w/ Supervision								
SFT	71.72	26.27	52.80	37.78	57.34	26.79	74.06	49.54
SPIN (Chen et al., 2024)	74.91	31.49	<u>57.40</u>	40.09	<u>71.35</u>	<u>29.91</u>	<u>83.96</u>	<u>55.59</u>
w/o Supervision								
STaR (Zelikman et al., 2022)	72.86	29.32	46.40	35.94	33.36	20.31	67.24	43.63
CoH (Liu et al., 2024)	74.37	<u>32.29</u>	56.20	38.56	69.08	28.13	82.51	54.45
Self-Rewarding (Yuan et al., 2024)	<u>76.04</u>	30.19	55.80	37.94	70.48	28.35	82.17	54.42
ScPO (Prasad et al., 2024)	71.11	30.99	55.00	<u>40.40</u>	59.87	28.57	78.92	52.12
Genius	**78.32**	**34.64**	**58.80**	**40.86**	**72.53**	**30.35**	**84.04**	**57.08**
Self-Training from **OpenHermes2.5 (32K)**								
w/ Supervision								
SFT	63.68	21.64	45.00	29.03	48.47	23.44	69.37	42.95
SPIN (Chen et al., 2024)	63.61	24.74	54.00	35.33	59.00	28.57	71.76	48.14
w/o Supervision								
STaR (Zelikman et al., 2022)	<u>75.51</u>	29.47	43.60	34.87	19.34	22.99	68.43	42.03
CoH (Liu et al., 2024)	74.29	31.22	54.80	38.40	<u>69.91</u>	29.69	81.48	<u>54.26</u>
Self-Rewarding (Yuan et al., 2024)	73.92	29.99	<u>56.00</u>	39.78	67.55	<u>30.13</u>	<u>81.66</u>	54.15
ScPO (Prasad et al., 2024)	73.54	<u>31.27</u>	54.80	<u>41.01</u>	58.65	28.79	79.52	52.51
Genius	**75.82**	**34.42**	**57.60**	**41.63**	**70.79**	**34.82**	**83.19**	**56.90**

Table 1: Main Results. The self-training performances from Mappie and OpenHermes2.5 corpora are reported independently. The optimal results are in bold and the suboptimal ones are underlined.

rent LLMs (e.g., LLaMA3.1) have been well optimized to perform chain-of-thought reasoning, it requires RL to cultivate a broader generalization capacity, rather than relying on SFT to inject reasoning patterns. Moreover, it is observed that the supervision of ground-truth responses does not yield enhancements, whereas self-generated responses can serve as an effective source to achieve performance boosts.

Performance Consistency on General Tasks. Beyond the reasoning-intensive tasks, it is also non-trivial to maintain the performances on the general benchmarks after post-training. Supported by the evaluation suite of OpenCompass (Contributors, 2023; Cao et al., 2024), we experiment on 6 widely-used general benchmarks, which are categorized into *subjective* and *objective* evaluation. We report the evaluation results in Table 2.

Overall, *Genius* keeps the stability of the performances on the general domain, with slight improvements in most cases. Specifically, self-training with *Genius* also achieves huge performance gains in Arena-Hard benchmark, which reflects the superior alignment with human preference. On the knowledge-intensive benchmarks (i.e., WikiBench, MMLU, and MMLU-Pro), *Genius* can maintain the performances of the base LLMs, avoiding catastrophic forgetting after post-training.

3.3 Generalization and Adaptation

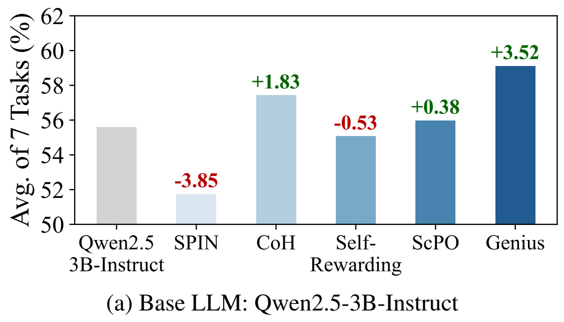
In this section, we analyze the generalization and adaptation of *Genius* to (i) different backbone LLMs, and (ii) on the competition-level task.

Generalization to other backbone LLMs. Besides the experiments on LLaMA3.1, we also supplement the evaluations on Qwen2.5-series. Fig. 4a and 4b present the average performance across 7 benchmarks on Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct respectively. Compared with all the strong baselines, *Genius* shows the highest performance gains over the base LLM. Notably, self-training on Qwen2.5 series models does not yield larger benefits than on LLaMA3.1-8B-Instruct. Some of the baselines even fail in some cases. One hypothesis is that Qwen2.5-Instruct has conducted comprehensive post-training, which makes it challenging for further advancement. It does not conflict with our key contribution that *Genius* serves as a versatile post-training technique, as it can function both as an ongoing self-training method for post-trained LLMs and as an alternative post-training strategy for the model itself.

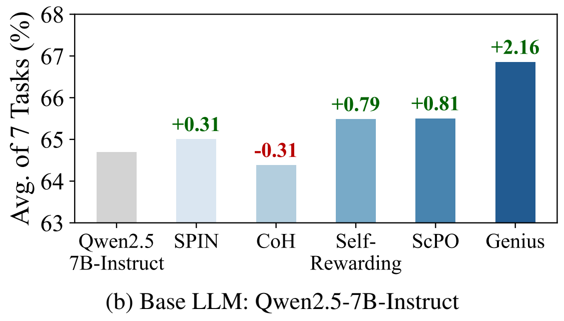
Adaptation to Challenging Task. Although *Genius* is not targeted for training large reasoning model like DeepSeek-R1, it is interesting to evaluate the challenging problems and uncover the po-

Models	Subjective Benchmarks			Objective Benchmarks		
	AlpacaEval	WildBench	Arena-H	WikiBench	MMLU	MMLU-Pro
LLaMA3.1-8B-Instruct	24.60	-1.11	30.31	27.65	71.14	48.62
Genius [From Magpie (25K)]	**26.96**	**2.68**	**50.00**	**28.75**	71.86	48.44
Genius [From OpenHermes (32K)]	25.47	1.44	**50.00**	27.00	**72.21**	**49.19**

Table 2: Performances on benchmarks in the general domain.



(a) Base LLM: Qwen2.5-3B-Instruct



(b) Base LLM: Qwen2.5-7B-Instruct

Figure 4: Generalization to Qwen2.5 series models. All methods are trained on the OpenHermes2.5 split. The numbers above the bars represent the average performance gain relative to the base model.

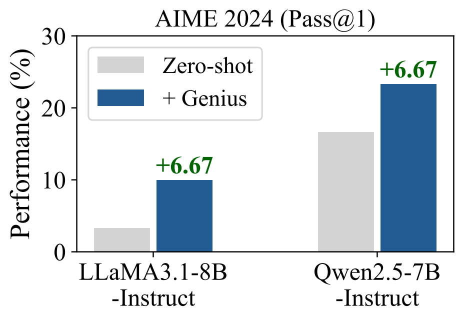


Figure 5: Results on AIME 2024.

tential. Figure 5 supplements the experiments on the competition-level task AIME 2024. We evaluate the model trained on OpenHermes2.5 split from LLaMA3.1-8B-Instruct and Qwen2.5-7B-Instruct respectively. It is observed that *Genius* boosts the performances by 6.67%. It verifies that *Genius* not only improves upon the fundamental LLM reasoning capabilities but also expands the boundaries of LLMs to address more intricate scenarios.

4 Analysis

4.1 Ablation Studies

To uncover the effectiveness of the major contributions of *Genius*, *sample-and-reward* strategy and *optimization* objectives are ablated respectively.

Ablation of Sampling Strategy. Table 3 presents the ablation results. Firstly, ablating the foresight module results in 3.17%-3.25% average performance drops. It illustrates that the foresight sampling strategy alleviates the short-sightedness of language model generation, and the employment of foresight score optimizes the self-rewarding of the step value. Secondly, replacing the *sampling* with the greedy selection also leads to significant drops. It verifies that our *re-sampling* strategy strikes the balance between exploration and exploitation.

Ablation of Optimization Methods. We present the comparison between various optimization approaches in Table 4, covering DPO, SimPO, IPO, ROPO and SFT. Among these popular approaches, our ACO loss function stands out, showcasing significant advantages with an average performance improvement across 7 reasoning benchmarks. Compared to the robustness optimization strategy ROPO, ACO is more suitable for self-training scenarios.

4.2 Post-Training Scaling Law

Limited by computational resources, we only provide the 10K-level training attempts, presented in the main table. To give the direction of future post-training scaling, Figure 6 shows the down-sampled scaling curves. We have the following findings.

***Genius* holds great potential for further scalability.** From Figure 6, it is observed that *Genius* can rapidly self-improve with limited training steps. The evolution progress proceeds smoothly with the training steps increasing. This curve suggests that self-training with *Genius* is far from saturation and still has room for improvement, whereas other

Variants	GSM8K	MATH	ReClor	LogiQA	StrategyQA	GPQA	ARC-c	Avg.	Δ
Self-Training from **Magpie (25K)**									
Genius	**78.32**	**34.64**	**58.80**	**40.86**	**72.53**	**30.35**	**84.04**	**57.08**	-
w/o *foresight*	71.65	31.07	57.60	39.17	66.20	30.13	81.57	53.91	+3.17
w/o *sampling*	69.29	31.37	57.60	39.17	68.03	24.33	81.06	52.98	+4.10
Self-Training from **OpenHermes2.5 (32K)**									
Genius	**75.82**	**34.42**	**57.60**	**41.63**	**70.79**	**34.82**	**83.19**	**56.90**	-
w/o *foresight*	73.01	30.74	57.60	35.94	67.25	29.46	81.57	53.65	+3.25
w/o *sampling*	72.93	30.59	56.00	41.17	65.76	30.13	80.03	53.80	+3.10

Table 3: Ablation studies on the sampling strategy. *w/o foresight* ablates the look-ahead process and simply samples from the distribution formed by step uncertainty. *w/o sampling* indicates that we adopt a greedy approach by selecting the two steps with the highest foresight score for *exploration*, while the step with the lowest foresight score serves as the negative response for *exploitation*.

Models	Average Performances	
	Magpie	OpenHermes
LLaMA3.1-8B-Instruct (CoT)	49.65	49.65
w/ Foresight Sampling		
+ **ACO**	**57.08**	**56.90**
+ DPO (Rafailov et al., 2024)	55.51	55.73
+ SimPO (Meng et al., 2025)	50.42	50.87
+ IPO (Azar et al., 2024)	52.31	52.20
+ ROPO (Liang et al., 2024)	55.30	55.25
+ SFT	44.63	49.70

Table 4: Comparison of different reinforced loss. In the experiments, we only replace the optimization methods while keep the foresight sampling strategy unchanged.

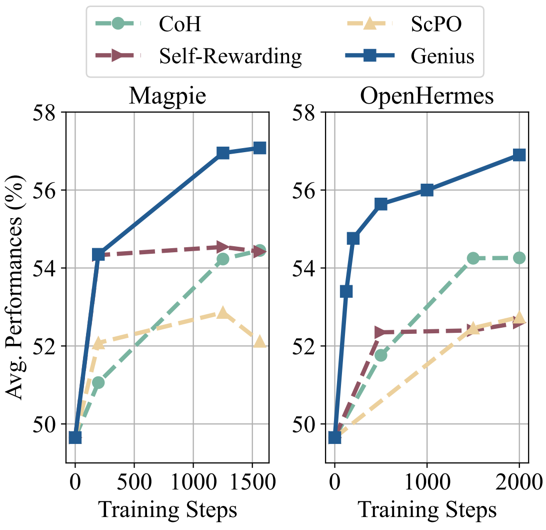


Figure 6: Post-training scaling law with LLaMA3.1-8B-Instruct as the base LLM.

baseline methods appear to face challenges when expanded in scale.

5 Related Works

Post-Training for LLM Reasoning. Boosting LLM reasoning by post-training has been a hot topic recently. Some efforts perform large-scale imitation learning on well-curated reasoning data (Yue et al., 2024; Toshniwal et al., 2024; Xu et al., 2025) to build large reasoning models for the specific domain (e.g., math, code). Considering the huge annotation costs, some recent endeavors have turned to self-training techniques to synthesize the reasoning trajectories more efficiently. However, these efforts still rely on supervision signals and in-domain training corpus, either with explicit outcome supervision (Zelikman et al., 2022; Xu et al., 2024a; Trung et al., 2024; Cheng et al., 2024) or auxiliary reward models (Zeng et al., 2024; Jiao et al., 2024). Conversely, we embark on a new path towards self-improving the general reasoning ability of LLMs without any form of supervision.

Optimization Techniques. Previous works (Yue et al., 2024; Hu et al., 2023; Wang et al., 2025) often rely on supervised fine-tuning to inject new reasoning patterns (e.g., reflection, refinement) into the foundational LLMs. With the development of stronger backbones, some efforts (Prasad et al., 2024; Cui et al., 2025; Guo et al., 2025) focus on offering the RL-based solution (Rafailov et al., 2024; Shao et al., 2024) to improve LLM reasoning, proving the RL scaling effect with outcome rewards. In our work, we propose a novel reinforced learning objective to perform robust optimization under the unsupervised setting. We uncover the promising scaling law of self-training with RL optimization.

6 Conclusion

This paper focuses on tackling the challenging and critical task of enhancing LLM reasoning, without relying on any external supervision. A generalizable and purely unsupervised self-training framework, named *Genius*, is proposed to address several key technical challenges on how to: (1) sample responses; (2) self-reward responses without external auxiliary; (3) robustly optimize with the self-curated data. Extensive experiments on 7 reasoning benchmarks, 7 general benchmarks, and 1 competition-level math task are included to comprehensively evaluate the LLM performance after post-training. The analysis of the scaling law curve uncovers the huge potential on further scalability.

Limitations

Limited by the computational resources, we only evaluate on 3B, 7B and 8B-sized LLMs, lacking the exploration of larger backbone LLMs. Also, the number of training queries is 25K and 32K respectively. Given that the scaling curve continues to show a trend of improvement, scaling up to larger input data sizes becomes an intriguing prospect.

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*.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs the method of paired comparisons. *Biometrika*, 39:324–345.

Maosong Cao, Alexander Lam, Haodong Duan, Hongwei Liu, Songyang Zhang, and Kai Chen. 2024. Compassjudger-1: All-in-one judge model helps model evaluation and evolution. *arXiv preprint arXiv:2410.16256*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. In *Forty-first International Conference on Machine Learning*.

Kanzhi Cheng, Yantao Li, Fangzhi Xu, Jianbing Zhang, Hao Zhou, and Yang Liu. 2024. Vision-language models can self-improve reasoning via reflection. *arXiv preprint arXiv:2411.00855*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. *GitHub repository*.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. 2025. Process reinforcement through implicit rewards.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*.

Hiroki Furuta, Kuang-Huei Lee, Shixiang Shane Gu, Yutaka Matsuo, Aleksandra Faust, Heiga Zen, and Izzeddin Gur. 2025. Geometric-averaged preference optimization for soft preference labels. *Advances in Neural Information Processing Systems*, 37:57076–57114.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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)*.

Qisheng Hu, Kaixin Li, Xu Zhao, Yuxi Xie, Tiedong Liu, Hui Chen, Qizhe Xie, and Junxian He. 2023. Instructcoder: Empowering language models for code editing. *arXiv preprint arXiv:2310.20329*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy F Chen, and Shafiq Joty. 2024. Learning planning-based reasoning by trajectories collection and process reward synthesizing. *arXiv preprint arXiv:2402.00658*.

Tzu-Sheng Kuo, Aaron Lee Halfaker, Zirui Cheng, Jiwoo Kim, Meng-Hsin Wu, Tongshuang Wu, Kenneth Holstein, and Haiyi Zhu. 2024. Wikibench: Community-driven data curation for ai evaluation on wikipedia. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–24.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Xize Liang, Chao Chen, Jie Wang, Yue Wu, Zhihang Fu, Zhihao Shi, Feng Wu, and Jieping Ye. 2024. Robust preference optimization with provable noise tolerance for llms. *arXiv preprint arXiv:2404.04102*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. 2024. Chain of hindsight aligns language models with feedback. In *The Twelfth International Conference on Learning Representations*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3622–3628.

Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. 2024. Non-myopic generation of language models for reasoning and planning. *arXiv preprint arXiv:2410.17195*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2025. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. 2024. Self-consistency preference optimization. *arXiv preprint arXiv:2411.04109*.

Team Qwen. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

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.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

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

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*.

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

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614.

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

Hemish Veeraboina. 2023. Aime problem set 1983-2024.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for ComputatDo NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMsional Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Yubo Wang, Xiang Yue, and Wenhu Chen. 2025. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*.

Fangzhi Xu, Qiushi Sun, Kanzhi Cheng, Jun Liu, Yu Qiao, and Zhiyong Wu. 2024a. Interactive evolution: A neural-symbolic self-training framework for large language models. *arXiv preprint arXiv:2406.11736*.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. 2023. Symbol-llm: Towards foundational symbol-centric interface for large language models. *arXiv preprint arXiv:2311.09278*.

Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, et al. 2025. Redstar: Does scaling long-cot data unlock better slow-reasoning systems? *arXiv preprint arXiv:2501.11284*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024b. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

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*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. Mammoth: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations*.

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, Lulu Zhao, Yijun Wang, Zifei Shan, and Junxian He. 2024. B-star: Monitoring and balancing exploration and exploitation in self-taught reasoners. *arXiv preprint arXiv:2412.17256*.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *arXiv preprint arXiv:2406.09136*.

A Gradient Analysis of ACO

Derivation of the Gradient. The complete form of ACO loss function is written as:

$$\mathcal{L}_{\text{ACO}} = -\mathbb{E}_{(x,T^w,T^l)\sim\mathcal{D}} \log\sigma \Bigg[\beta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)} - \beta \text{clip}\left(\exp \frac{-(A_l - A_w)}{\alpha}, 1\right) \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}\Bigg]$$

To simplify the formulation, we denote the advantage calibration term as $w(x, A)$:

$$w(x, A) = \text{clip}\left(\exp \frac{-(A_l - A_w)}{\alpha}, 1\right)$$

Our objective is to derive the gradient $\nabla_\theta \mathcal{L}_{\text{ACO}}$. At the beginning, we define:

$$z = \beta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)} - \beta w(x, A) \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}$$

The gradient of the ACO loss can be represented in the form of:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{ACO}} &= -\mathbb{E}_{(x,T^w,T^l)} \nabla_\theta \log \sigma(z) \\ &= -\mathbb{E}_{(x,T^w,T^l)} \frac{\sigma'(z)}{\sigma(z)} \nabla_\theta z\end{aligned}$$

Using the unique characteristics of sigmoid function: $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ and $\sigma(-x) = 1 - \sigma(x)$, the gradient of ACO becomes:

$$\nabla_\theta \mathcal{L}_{\text{ACO}} = -\mathbb{E}_{(x,T^w,T^l)} (1 - \sigma(z)) \nabla_\theta z \tag{13}$$

Now we need to compute $\nabla_\theta z$. Since $w(x, A)$ is independent of θ, the gradient of z reduces to:

$$\begin{aligned}\nabla_\theta z = \beta \nabla_\theta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)} \\ - \beta w(x, A) \nabla_\theta \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}\end{aligned}$$

The gradients of the log probability terms are:

$$\nabla_\theta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)} = \nabla_\theta \log \pi_\theta(T^w|x)$$

$$\nabla_\theta \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)} = \nabla_\theta \log \pi_\theta(T^l|x)$$

Replacing them into the gradient of z, we have:

$$\begin{aligned}\nabla_\theta z = \beta \nabla_\theta \log \pi_\theta(T^w|x) \\ - \beta w(x, A) \nabla_\theta \log \pi_\theta(T^l|x)\end{aligned} \tag{14}$$

Substituting Eq. 14 into Eq. 13, the gradient of ACO is derived as:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{ACO}} = -\mathbb{E}_{(x,T^w,T^l)} \underbrace{(1 - \sigma(z))}_{\text{scale}} \cdot \\ \underbrace{[\beta \nabla_\theta \log \pi_\theta(T^w|x) - \beta w(x, A) \nabla_\theta \log \pi_\theta(T^l|x)]}_{\text{postive and negative gradients}}\end{aligned} \tag{15}$$

In this equation, $1 - \sigma(z)$ controls the scale of the gradients. Putting z in the formula, then we can obtain the following format:

$$\begin{aligned}1 - \sigma(z) = \beta w(x, A) \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)} \\ - \beta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)}\end{aligned} \tag{16}$$

When the negative trajectory T^l brings more advantages (i.e., higher A-value), $w(x, A)$ would decrease in value according to the illustration in Fig. 3. Then, the gradient scale $1 - \sigma(z)$ drops, offering less optimization to the corresponding training data pairs. It aligns with our initial motivation of calibrating with advantage values.

Relationship with Other Reinforced Loss Functions. With the derived gradient formulation, we discuss the relationship between ACO and other representative reinforced loss function (Furuta et al., 2025). DPO, c-DPO, and ROPO are included, where the latter two losses are specifically designed for the robust optimization.

DPO. The formulation of DPO loss is:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\beta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)} - \beta \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}\right)$$

The gradient of DPO $\nabla_\theta \mathcal{L}_{\text{DPO}}$ can be represented as the same formulation of Eq. 15 and 16, where the gradient scale is $1 - \sigma(z)$. The DPO loss can be regarded as the special case of ACO loss, with $\alpha \to +\infty$.

c-DPO. The c-DPO loss is designed to apply the label smoothing technique to alleviate the noise. Its formulation is:

$$\begin{aligned}\mathcal{L}_{\text{c-DPO}} = \\ -\epsilon \log \sigma \left(\beta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)} - \beta \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}\right) \\ -(1 - \epsilon) \log \sigma \left(\beta \log \frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)} - \beta \log \frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}\right)\end{aligned}$$

The gradient of c-DPO $\nabla_\theta \mathcal{L}_{\text{c-DPO}}$ has the scale of $(1-\epsilon)-\sigma(z)$ with $\alpha \to +\infty$. Compared with our ACO loss, c-DPO offers static and equal calibration for each data pair with the use of ϵ, while ACO loss provides an adaptive solution.

ROPO. The ROPO loss specially proposed for the robust optimization:

$$\mathcal{L}_{\text{ROPO}} = -\gamma\sigma\left(\beta\frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}\sigma\frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)}\right) - \eta\log\sigma\left(\beta\frac{\pi_\theta(T^w|x)}{\pi_{\text{ref}}(T^w|x)}\sigma\frac{\pi_\theta(T^l|x)}{\pi_{\text{ref}}(T^l|x)}\right)$$

It has the similar gradient formulation with ACO loss when $\alpha \to +\infty$. Its gradient scale can be derived as $(\gamma - \eta\sigma(z))(1-\sigma(z))$. The main difference is ROPO provides the robust calibration with the positive and negative gradients, while ACO loss obtains the advantage during the sampling process.

B Implementation Details

B.1 Baselines

SFT We finetune the LLM given the input query (x) and the labeled response (a).

SPIN (Chen et al., 2024) It iteratively refines the model-generated response against the labeled ones with an objective similar to DPO.

STaR (Zelikman et al., 2022) It continuously bootstraps from the self-constructed response through finetuning.

CoH (Liu et al., 2024) It obtains both positive and negative responses via self-prompting and optimizes LLM with DPO loss function.

Self-Rewarding (Yuan et al., 2024) It leverages the LLM itself as a judge to label the self-generated responses (1-5 scores), then the LLM is optimized with DPO loss on the constructed preference pairs.

ScPO (Prasad et al., 2024) This approach generates multiple trajectories and labels the preference with self-consistency. To address the open-ended generation scenarios, we modify the implementation of self-consistency with the cluster strategy.

B.2 Setup

Sampling. The foresight sampling process is supported by 32*A100 GPUs of 80GB VRAM, and it is accelerated by the vLLM engine. To ensure the diversity of sampling, we set the generation temperature to 0.6. The step beam size M is set to 2, the rollout number on each beam N is set to 4, and the number of foresight steps K is 4.

Training. The optimization of LLM is implemented with 8*A100 GPUs of 80GB VRAM, supported by Deepspeed Zero3 and FlashAttention2. The total training batch size is set to 128, and the learning rate is 5e-7. The hyper-parameter α in the ACO loss is set to 1. Based on the configuration of sampling, we keep 4 training pairs for each query (collect one at each timestamp). Therefore, the size of the training datasets is 100,000 and 128,000 for Magpie and OpenHermes-2.5 respectively.

Inference. The inference is supported by the vLLM engine. We keep the default configuration of vLLM with a temperature of 1.0. For GSM8K and MATH benchmarks, we leverage the widely-used few-shot examples, utilizing 4-shot for GSM8K and 8-shot for MATH. For other benchmarks, we evaluate under the zero-shot setting.

C Analysis of The Training Corpus

We visualize the data distribution difference between the training corpus and the evaluation tasks in Fig. 7. In the implementation, we utilize the sentence-embedding model (`multi-qa-mpnet-base-dot-v1`) to acquire the high-dimensional embeddings of the queries. Subsequently, we employ the t-SNE algorithm to visually represent these embeddings in a lower-dimensional space. A set of 500 samples are randomly selected in each dataset.

To differentiate the data domains, we employ the following color scheme: **red** denotes the general training corpus, **purple** denotes mathematical datasets, **yellow** denotes logical datasets, and **green** is used for other reasoning datasets. It is observed that the data distribution of the general training corpus is distinct and independent from that of other evaluation domains. It supports our conclusion that *Genius* paves the way to self-improve LLM reasoning from unsupervised queries in the general domain.

Following the similarity analysis proposed in (Xu et al., 2023), we also report the intra- and inter-class sample distances in Table 5.

The average similarity between the training corpus and the domain-specific downstream tasks, as depicted in the table, is notably low. Among the tasks evaluated, the logical reasoning benchmarks (i.e., ReClor and LogiQA) exhibit the least resem-

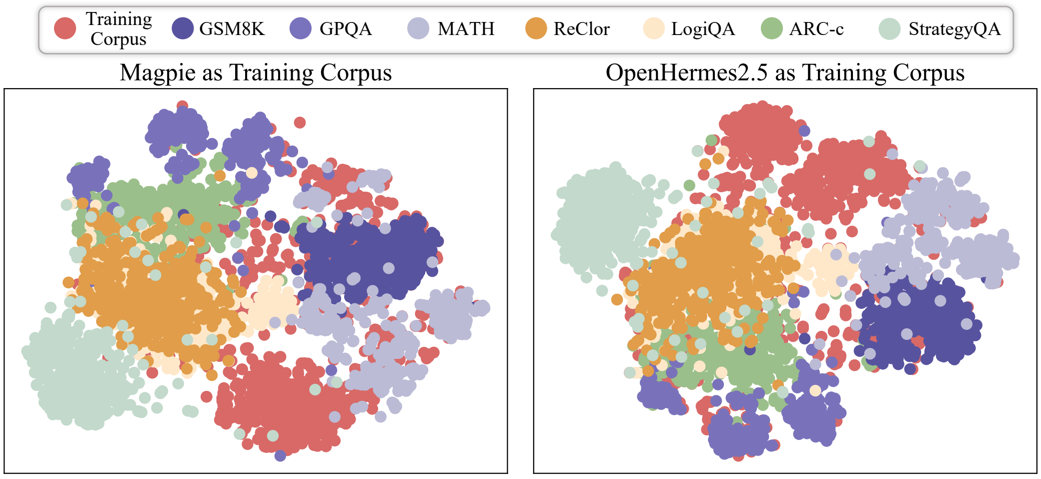


Figure 7: Visualization of data distribution. The training corpus is marked in red color.

Task	w/ Self	w/ Magpie	w/ OpenHermes
GSM8K	0.3967	0.1343	0.1256
MATH	0.1641	0.0908	0.0719
GPQA	0.1907	0.0566	0.0602
ReClor	0.2609	0.0552	0.0792
LogiQA	0.2330	0.0732	0.0930
StrategyQA	0.1186	-0.0045	0.0025
ARC-c	0.2276	0.0673	0.0843

Table 5: Comparison between intra- and inter-class distance. *w/ Self* denotes the distance within the task queries. *w/ Magpie* means the distance between the target task with Magpie training corpus, while *w/ OpenHermes* denotes the distance between the target task with OpenHermes2.5 training corpus.

blance to the training corpus, making them an ideal evaluation scenario for our approach.

D More Experiments on Coding Tasks

Besides the natural language reasoning and understanding tasks, it is also interesting to present the potential of *Genius* on coding-related benchmarks, which is one of the key abilities of LLMs (Sun et al., 2024). Table 6 reports the performances on MBPP (Austin et al., 2021) and LiveCodeBench (Jain et al., 2024).

Models	MBPP	LiveCodeBench
LLaMA3.1-8B-Instruct	69.65	19.50
Genius [From Magpie]	71.60	19.75
Genius [From OpenHermes]	71.98	21.25

Table 6: Experiments on coding tasks.

It is observed that self-training with *Genius* on the general data would also benefit the LLM coding abilities, which involve strict formats and structured representations.