Learning to Reason via Self-Iterative Process Feedback for Small Language Models

Kaiyuan Chen, Jin Wang* and Xuejie Zhang
School of Information Science and Engineering
Yunnan University
Kunming, China
chenkaiyuan@stu.ynu.edu.cn, {wangjin, xjzhang}@ynu.edu.cn

Abstract

Small language models (SLMs) are more efficient, cost-effective, and customizable than large language models (LLMs), though they often underperform in specific areas like reasoning. Past methods for enhancing SLMs' reasoning, such as supervised fine-tuning and distillation, often depend on costly external signals, resulting in SLMs being overly confident with limited supervision signals, thus limiting their abilities. Therefore, this study enables SLMs to learn to reason from self-iterative feedback. By combining odds ratio preference optimization (ORPO), we fine-tune and align SLMs using positive and negative signals generated by themselves. Additionally, we introduce process supervision for rewards in preference alignment by sampling-based inference simulation and process reward models. Compared to Supervised Fine-Tuning (SFT), our method improves the performance of Gemma-2B by 12.43 (Acc) on GSM8K and 3.95 (Pass@1) on MBPP. Furthermore, the proposed method also demonstrated superior out-of-domain generalization capabilities on MMLU_Math and HumanEval.

1 Introduction

Reasoning is a popular area of research in natural language processing. Recent studies (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023b) show that closed-source LLMs with Chain-of-Thought (CoT) demonstrated excellent performance in various reasoning tasks, even without additional fine-tuning on specialized supervised datasets (Cobbe et al., 2021). However, due to the limited parameters, open-source SLMs ($\leq$7B) haven't fully demonstrated this capability before despite being more cost-effective than their larger counterparts (Magister et al., 2023; Ho et al., 2023). Therefore, enhancing the incremental reasoning abilities of SLMs has recently become a significant research focus.

*Corresponding author

Previous works (Cobbe et al., 2021; Hendrycks et al., 2021b; Magister et al., 2023; Ho et al., 2023; Hsieh et al., 2023; Chen et al., 2024a) have shown that fine-tuning SLMs on meticulously designed large-scale supervised datasets effectively narrows the reasoning performance gap between open-source SLMs and closed-source LLMs. However, constructing such datasets requires extensive supervision from humans or advanced LLMs. Additionally, due to limited parameters and datasets, SLMs tend to become overly confident with limited supervision signals, which means that SLMs are prone to lacking generalization (Yuan et al., 2023) or being misled by incorrect reasoning paths (Wang et al., 2023a; Bentham et al., 2024), particularly when reasoning by CoT, as shown in Figure 1 (above).

Self-taught methods like STaR (Zelikman et al., 2022) and RFT (Yuan et al., 2023) make it possible to enhance language models' reasoning abilities without external annotation signals, where the model learns from self-generated reasoning paths that lead to correct final answers. By incorporating the Direct Preference Optimization (DPO) (Rafailov et al., 2023), self-refine methods also aim to enhance LLMs' reasoning abilities by preference alignment on self-generated positive and negative samples, where each sample is labeled as positive or negative either by advanced LLMs (like GPT-4) (Lee et al., 2024) or automatically based on whether the final answer is correct (Pang et al., 2024). However, these methods depend only on result-based binary rewards, which means they overlook a significant amount of detailed step-by-step feedback, as shown in Figure 1 (below). Please refer to Appendix D for more details of related works.

To combine the advantages of self-iterative learning and fine-grained process feedback, we propose fine-tuning SLMs on self-generated positive and negative samples through self-iterative process feedback (SIPF). A process reward model (PRM)

Proceedings of the 31st International Conference on Computational Linguistics, pages 3027–3042
January 19–24, 2025.

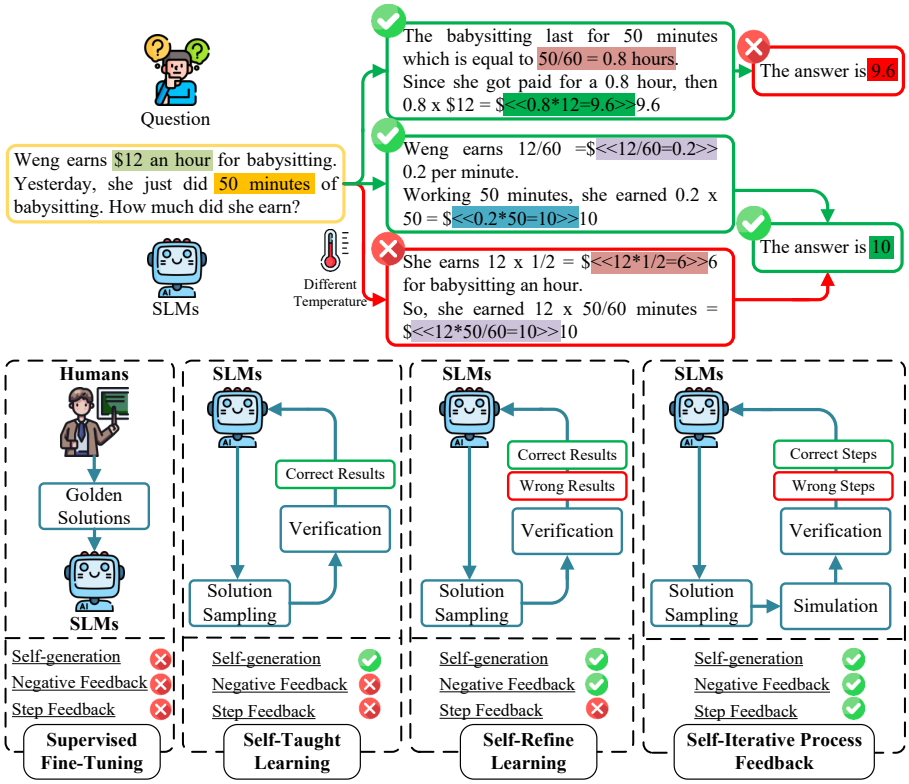


Figure 1: The conceptual diagram of the proposed self-iterative process feedback method against several previous methods (below). Compared to prior approaches, SIPF emphasizes on the correctness of reasoning steps. This means that SIPF can distinguish between correct reasoning with incorrect results and incorrect reasoning with correct results (above).

(Lightman et al., 2024) produces the process feedback signals without human annotations.

The proposed self-iterative method involves these steps: First, using fine-tuned SLMs for sampling reasoning paths (CoT); Second, applying sampling-based inference simulation to label the correctness of steps in some examples, which are then used for training the verifier (or PRM); Next, scoring all sampled reasoning paths with the verifier and constructing preference datasets; Finally, performing odds ratio preference optimization (ORPO) (Hong et al., 2024) to align SLMs on preference datasets.

Unlike recent process feedback-based reasoning optimization methods (Jiao et al., 2024; Wang et al., 2024), our approach focuses on gradually improving the SLMs' reasoning abilities through self-iteration. Overall, the main contributions of this work can be summarized as follows:

1) We propose a self-iterative process feedback optimization method to gradually improve reasoning in open-source SLMs without additional human-annotated signals. It also does not need process supervision signals from humans when combined with sampling-based inference simulation.

2) We propose improving self-refine methods by incorporating fine-grained process feedback, with process feedback rewards assigned by a PRM. Comprehensive and rigorous analysis results indicate that incorporating process feedback leads to more robust improvements in self-refine methods.

3) We conduct experiments on various types and scales of SLMs, including TinyLlama-v1.1, Phi-1.5, and Gemma-2B, to validate the universality of the proposed method. Extensive experiments show that our method outperforms supervised fine-tuning, self-taught, and self-refine methods on multi-step reasoning tasks like GSM8K and MBPP. It also achieves superior out-of-domain generalization on MMLU_Math and HumanEval. Additional experimental analyses also strongly support the reliability of the proposed approach.

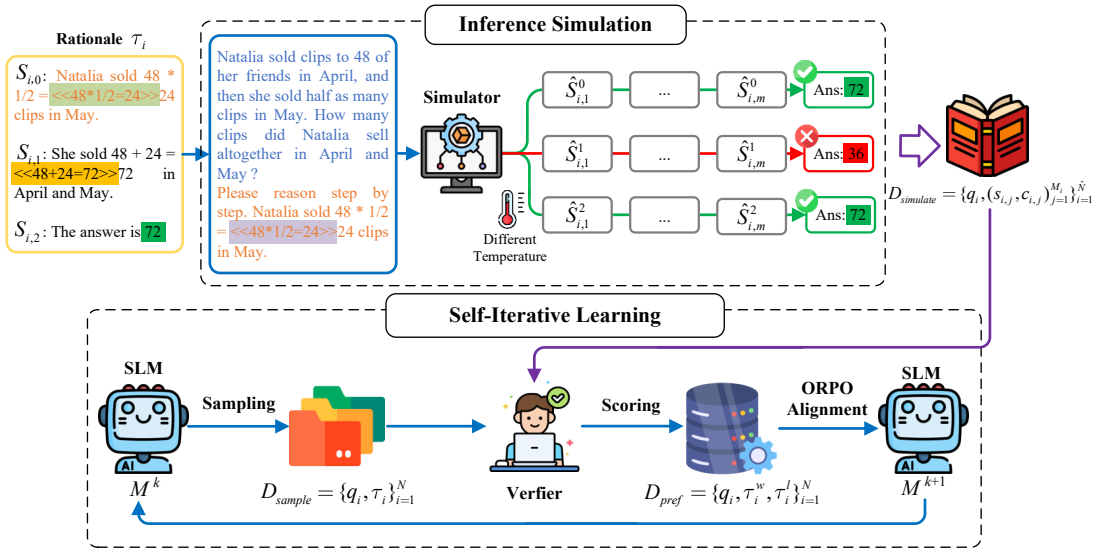


Figure 2: Overall framework of the proposed learning to reason from self-iterative process feedback. A single iteration of the online learning process includes sampling, collecting, inference simulation, fine-tuning verifier, scoring to construct the preference dataset, and RL alignment by ORPO.

2 Self-Iterative Process Feedback

Instead of observing the correctness of the final answer, the proposed method enhances the reasoning ability of the SLMs M with parameters θ by observing the correctness of each reasoning step.

We introduce a simulator S and a verifier V, both well-trained LLMs on the specific task to achieve this goal. Specifically, a continuous value from 0 and to 1 represents the correctness of intermediate reasoning steps.

The simulator S uses the SLM's intermediate steps as input to perform multiple reasoning simulations, determining the correctness of each step based on the final results of multiple simulations. The verifier V rates the correctness scores of each step generated by the SLM and generator. Figure 2 shows the detailed architecture of the proposed SIPF method.

2.1 Reasoning Paths Sampling

The fine-tuned M with parameters θ can sample a set of different reasoning paths, i.e. $\{\tau_0, \tau_1, ..., \tau_n\}$, by high temperature T for a given question q. The process can be formally described as follows:

$$R = \{\tau_i | \tau_i \sim M(q_i, T, \theta)\}_{i=1}^{N} \quad (1)$$

where R represents possible reasoning paths generated by M given the question q_i and temperature T. The initial model M_0 is obtained by fine-tuning the original model on the original dataset D_0, and the k-th iteration model M_k is obtained by aligning it on the k-th iteration dataset D_k via ORPO. Additionally, $R = \{\tau_0, \tau_1, ..., \tau_n\}$ will be deduplicated and diversified based on reasoning paths (for math) or edit-distance (for code).

2.2 Process Reward Estimation

Compared to outcome-based rewards, which solely rely on the correctness of the final answer a of reasoning. Process supervision rewards assess the correctness of each intermediate step s_j in the reasoning path $\tau = (s_0, s_1, ..., s_m, a)$, where $j \in [0, m]$ and a is the final result.

Inspired by Monte Carlo Tree Search (MCTS), sampling-based inference simulation provides a feasible approach for determining the correctness of steps of the reasoning path without cumbersome human annotations (Lightman et al., 2024). The basic idea is that an intermediate step will likely be correct if it frequently reaches the correct result.

As shown in Figure 2, the simulator performs multiple samplings from the reasoning step $s_{i,j}$ in τ_i, obtaining K sampled paths: $\{\hat{\tau}_{i,j}^t\}_{t=1}^{K} = \{(s_{i,0}, ..., s_{i,j}, \hat{s}_{i,j+1}^t, \hat{s}_{i,j+2}^t, ...\hat{s}_{i,m}^t, \hat{a}_i^t)\}_{t=1}^{K}$. The correctness $c_{i,j}$ of step $s_{i,j}$ can be defined as follows:

$$c_{i,j} = \begin{cases} 1, & \text{if } \sum \mathbb{I}(\hat{a}_i^t = a_i^*) > \delta \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where δ is threshold that denotes whether a step is considered correct only if the number of simulated correct results exceeds a specific count, $\mathbb{I}$

represents the $0/1$ indicator function and a_i^* is the gold answer.

2.3 Process Reward Model

The verifier can be used to assign fine-grained step-level scores to reasoning paths. Given a dataset annotated through inference simulation:

$$\begin{aligned} D_{simulate} =& \{q_i, \tau_i, c_i\}_{i=1}^{\hat{N}} \\ =& \{q_i, (s_{i,j}, c_{i,j})_{j=1}^{M_i}\}_{i=1}^{\hat{N}} \end{aligned} \quad (3)$$

where M_i represents the total number of steps in τ_i and $\hat{N}$ represents the number of samples less than N. The corresponding loss function can be formalized as follows:

$$\begin{aligned} \mathcal{L}_{PRM} = -\sum_{i=0}^{N}\sum_{j=0}^{M_i} & c_{i,j} \log \hat{c}_{i,j} \\ & + (1 - c_{i,j}) \log(1 - \hat{c}_{i,j}) \end{aligned} \quad (4)$$

where $\hat{c}_{ij}$ is the probability assigned by the verifier that step $s_{i,j}$ is correct, which is ranges from $[0, 1]$. Therefore, the reward r of an reasoning path $\tau = (s_0, s_1, ..., s_m, a)$ of the i-th question q_i can be estimated by the fine-tuned verifier as follows:

$$r(\tau) = \frac{\sum_{j=0}^{m} \hat{c}_{i,m}}{m} \quad (5)$$

where m is the total number of reasoning steps. Since the performance of verification is related to language modeling capabilities, the verifier is usually an open-source model with strong reasoning performance.

2.4 RL Alignment via ORPO

Based on the PRM, we can assign different reward r to various reasoning paths τ, thereby constructing the preference dataset as follows:

$$D_{pref} = \{q_i, {\tau_i}^w, {\tau_i}^l \,\big|\, r({\tau_i}^w) - r({\tau_i}^l) \geq \eta\} \quad (6)$$

where τ^w is the chosen path while τ^l is the reject one and η is the confidence margin. Benefiting from the efficiency of ORPO, supervised fine-tuning and preference alignment can be incorporate into a single process. It allows the model to learn more diverse reasoning from the chosen reasoning paths and avoid erroneous patterns through preference alignment. The learning objectives corresponding to this process are denoted as:

Algorithm 1 Iterative Training Procedure

Initialize: pretrained SLM M; fine-tuned verifier V; dataset $D_0 = \{q_i, \tau_i, a_i\}_{i=1}^{N}$;
$M_0 = \text{SFT}(M, D_0)$
for $k = 1$ to N **do**
  $D_{\text{sample}} = \text{Sample}(M_{k-1}, D_{k-1})$
  $D_k = D_{k-1} \cup D_{\text{sample}}$
  $D_{\text{pref}} = \text{Score}(V, D_k)$
  $M_k = \text{ORPO}(M, D_{\text{pref}})$
end for

$$\begin{aligned} \mathcal{L}_{ORPO} = -\mathbb{E}_{(q,\tau^w,\tau^l)} \Bigg[& \mathcal{L}_{SFT}(q, \tau^w) \\ & + \beta \left(\log \sigma \left(\frac{\text{odds}(\tau^w \mid q)}{\text{odds}(\tau^l \mid q)} \right) \right) \Bigg] \end{aligned} \quad (7)$$

$$\text{odds}(\tau \,|q\,) = \frac{P(\tau \,|q\,)}{1 - P(\tau \,|q\,)} \quad (8)$$

where $\mathcal{L}_{SFT}(q, \tau^w)$ is is the negative log-likelihood loss on τ^w, σ is log sigmoid function, β is weight parameters, and $\text{odds}(\tau \,|q\,)$ represents the ratio between the probability of obtaining reasoning path τ given question q and the probability of not generating it.

2.5 Self-Iterative Process Feedback

Through self-iteration, SLMs and datasets can be updated iteratively, overcoming the limitations of finite datasets and reducing bias with continuous online feedback. This process involves a series of iterative models $M_0, M_1, ..., M_k$ and their corresponding datasets $D_0, D_1, ..., D_k$. Algorithm 1 shows the details of self-iterative process feedback.

3 Experiments

3.1 Datasets

The experiments involve two challenging types of multi-step reasoning tasks: mathematical reasoning (GSM8K (Cobbe et al., 2021), MMLU_Math (Hendrycks et al., 2021a)) and code generation (MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021)).

GSM8K involves solving math word problems in grade school. MBPP tasks involve basic programming knowledge, requiring multi-step reasoning and passing unit tests. Both datasets offer publicly available training and testing sets.

Model	TinyLlama-v1.1		Gemma-2B	
Method\Task	GSM8K	MMLU_Math (OOD)	GSM8K	MMLU_Math (OOD)
CoT (8-shot)	1.14	5.08	14.40	13.28
SFT	13.75	6.21	31.54	24.01
RFT	14.40	6.21	34.80	24.58
pRFT	14.56	7.34	34.57	28.25
SRF	7.43	2.26	27.98	23.73
pSRF	9.63	3.95	28.35	23.73
RPO	14.25	6.21	37.91	27.12
STaR	14.63	5.65	34.72	24.58
SIPF-Iter1	16.22	7.34	38.15	25.98
SIPF-Iter2	17.44	7.91	42.00	**29.10**
SIPF-Iter3	**18.57**	**9.60**	**43.97**	28.25

Table 1: Comparison of different methods on the mathematical reasoning tasks GSM8K (in-domain) and MMLU_Math (out-of-domain).

To assess the models' out-of-domain generalization, we will train them on GSM8K and MBPP, and then evaluate them on MMLU_Math and HumanEval respectively. MMLU_Math comprises college, high school, and elementary school mathematical problems from MMLU. Please refer to Appendix A.1 for more details of datasets.

3.2 Experiment Setup

The experiments consider three types and scales of pre-trained SLMs, including TinyLlama-v1.1 (Zhang et al., 2024b), Phi-1.5 (Li et al., 2023), and Gemma-2B (Team et al., 2024). All models in the experiments are fine-tuned by QLoRA (Dettmers et al., 2023) and employ Unsloth[1] for inference acceleration.

The deepseek-math-7b-instruct and deepseek-math-7b-rl (Shao et al., 2024) are employed as the simulator and reward model for mathematical reasoning, respectively, and deepseek-coder-6.7b-instruct (Guo et al., 2024) as both the simulator and reward model for code generation.

In each iteration, we collect sufficient positive and negative samples through temperature sampling and ensure sample diversity by deduplicating based on reasoning paths (for math) or edit-distance (for code). Inference simulation is performed by sampling eight times for all tasks, with temperatures set to 1 for math tasks and 0.7 for code tasks. Please refer to Appendix A.2 for more setup details.

[1] https://github.com/unslothai/unsloth

3.3 Baseline

In addition to CoT and supervised fine-tuning, we consider the following existing methods as strong baselines:

Self-Taught Methods: STaR (Zelikman et al., 2022) and RFT (Yuan et al., 2023) are Self-Taught methods based on outcome feedback, where the models only learn from the reasoning paths that lead to correct results. RFT boosts diversity in self-generated reasoning with high-temperature sampling, while STaR uses greedy decoding and iteratively improves performance.

Self-Refine Methods: Similar to the approach used by Yuan et al. (2024); Lee et al. (2024), we consider using SFT and DPO to align the model with self-sampled positive and negative examples. However, labels of examples are not assigned by LLMs but are instead automatically assigned based on the correctness of the final results. In subsequent experiments, this baseline will be referred to as SRF. Additionally, RPO (Pang et al., 2024) optimizes reasoning by aligning self-generated positive and negative outcome feedback. We compare RPO's performance after one iteration in the experiments.

For a strict comparison, we use PRMs described in Section 2.3 to introduce process feedback for RFT and SRF. The improved methods are referred to as pRFT and pSRF, respectively. Please refer to Appendix A.3 for more baselines details.

3.4 Evaluation

We evaluate the model's accuracy for mathematical reasoning tasks, while for code generation, Pass@1

Model	Phi-1.5		Gemma-2B	
Method\Task	MBPP	HumanEval (OOD)	MBPP	HumanEval (OOD)
CoT (0-shot)	34.87	33.54	28.29	20.12
SFT	35.20	32.32	29.75	17.68
RFT	34.60	26.22	27.49	14.02
pRFT	36.60	28.05	28.56	15.24
SRF	22.38	17.07	16.47	12.80
pSRF	25.92	19.51	19.17	11.59
RPO	28.52	33.54	31.53	20.73
STaR	**38.44**	29.88	28.56	15.85
SIPF-Iter1	36.87	**34.76**	31.83	20.12
SIPF-Iter2	38.21	29.88	32.16	**21.95**
SIPF-Iter3	37.14	31.10	**33.70**	20.73

Table 2: Comparison of different methods on the code generation tasks MBPP (in-domain) and HumanEval (out-of-domain).

is used as the evaluation metric. By default, all results are generated using greedy decoding by the model. For more details on the evaluation, please refer to Appendix A.1.

4 Results and Analysis

4.1 Comparison of In-Domain Performance

Tables 1 and 2 indicate that SIPF demonstrates superior in-domain performance on multi-step reasoning tasks: mathematics (GSM8K) and code generation (MBPP). Specifically, SIPF shows significant improvements over SFT and CoT. This improvement is observed across different models.

With just one iteration, SIPF's in-domain performance surpasses self-taught and self-refine methods and their process feedback-based versions (pRFT and pSRF). Additionally, with the help of self-iteration, SIPF achieves continuous improvements in most cases, as shown in Tables 1 and 2.

4.2 Comparison of Out-of-Domain Performance

As mentioned in Section 1, most methods based on external signals are constrained by limited resources and fail to exhibit good out-of-domain performance. As indicated in Table 2, SFT shows degraded performance in the code generation task HumanEval (out-of-domain) compared to CoT.

In contrast, in most cases, SIPF demonstrates strong out-of-domain performance, surpassing existing methods like STaR, RFT, and SRF. SIPF also further improve its performance on MMLU_Math through self-iteration, as shown in Table 1. However, for the code generation task HumanEval, the improvements from self-iteration are limited, as shown in Table 2.

4.3 Analysis of Performance Degradation in DPO-based Self-Refine Methods

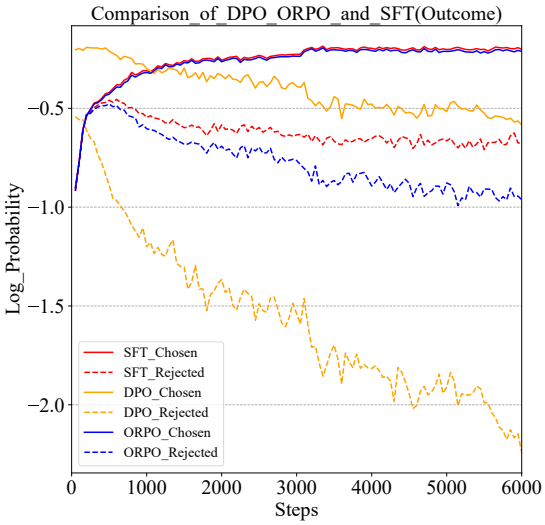
Tables 1 and 2 indicate that SRF shows degraded performance compared to SFT in all cases, regardless of whether it is based on outcome or process feedback. As shown in Figure 3, we analyze the chosen and rejected probability across different models and feedback types to further explore the underlying reasons.

Figure 3 shows that DPO reduces the probability of generating τ^w during training, which directly causes performance degradation in SFT models after aligning feedback signals. These results are consistent with findings from previous works (Hong et al., 2024; Pang et al., 2024). In fact, the loss form of DPO causes the model to focus excessively on what is a poor response (as shown in Figure 3) while overlooking what makes a good response. It has also been confirmed in recent work (Feng et al., 2024). In contrast, ORPO-based SIPF improves the probability of generating τ^w due to the supervised fine-tuning on τ^w.

4.4 Analysis of Aligning Positive and Negative Examples

Self-Taught methods focus solely on iterative supervised fine-tuning on self-generated positive examples, while SIPF further incorporates aligning on positive and negative examples via ORPO.

Figure 3 shows that SFT increases the generation probability of τ^w while it does not effectively



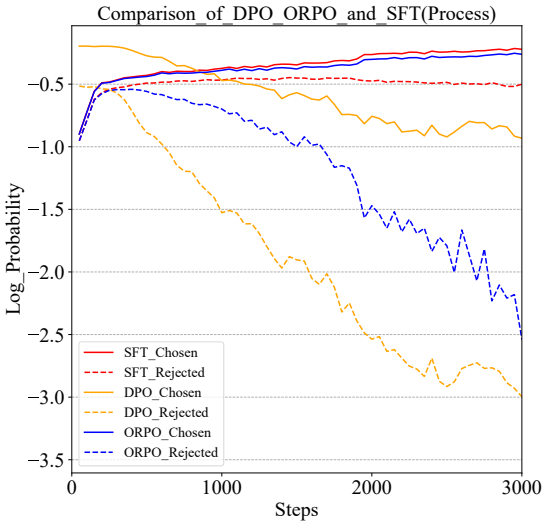


Figure 3: Comparison of the training processes of SFT-based self-taught, DPO-based SRF, and ORPO-based SIPF on GSM8K using Gemma-2B. The DPO-based SRF consistently reduced the probability of generating τ^w. SIPF is more effective than self-taught methods at reducing the probability of generating τ^l. Additionally, different types of feedback (outcome or process) have different effects on training.

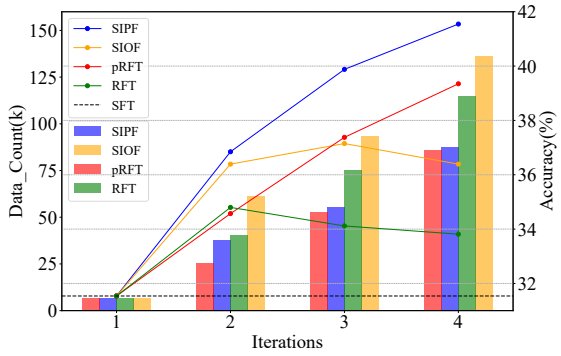


Figure 4: Comparison of accuracy and data count for various self-iteration methods across different iterations on GSM8K.

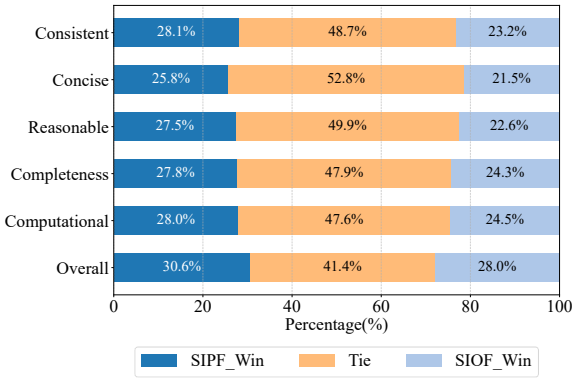


Figure 5: Automatic evaluation of the reliability of rationales generated by different methods (process-based or outcome-based) using GPT-4.

reduce the probability of τ_l, which is consistent with the observations of previous works. In comparison, SIPF further lowers the likelihood of generating rejected examples, indicating that it more effectively reduces erroneous reasoning in SLMs and improves reasoning performance. Additionally, different types of feedback affect SFT and ORPO differently. The rejected probability curves of SFT and ORPO differ more significantly from process feedback to outcome feedback. Under outcome feedback, both SFT and OPRO reduce the probability of generating rejected samples, leading to a similar performance in handling rejected samples.

4.5 Comparison of Different Self-Iterative Learning Methods

We compare the accuracy and data count changes during continuous iteration for four self-iterative methods based on process or outcome feedback. Specifically, RFT can also be implemented as a self-iterative optimization method, similar to (Singh et al., 2024). SIOF represents iterative learning using only outcome feedback, with other settings consistent with SIPF.

Figure 4 shows that all methods achieve higher accuracy than the SFT baseline through self-iteration learning. SIPF with process feedback signals consistently outperforms other methods in

Method\Type	ORM	PRM
TinyLlama-v1.1	59.64	**67.00**
Llama-2-7B	70.09	**78.38**
deepseek-math-instruct-7B	75.60	**85.24**
deepseek-math-rl-7B	77.38	**87.58**

Table 3: Accuracy analysis of different reward models trained with process and result feedback on manually annotated GSM8K evaluation dataset.

every iteration.

It is worth noting that the differing impacts of process feedback and outcome feedback on reasoning performance become evident through multiple iterations. RFT and SIOF, based on outcome feedback, show degraded performance after reaching their capacity limits. In contrast, process feedback methods (SIPF and RFT) continue to improve, likely due to higher-quality examples. This is also reflected in the observed data count changes across iterations, where more training examples do not always lead to better reasoning performance.

Additionally, aligning positive and negative samples benefits reasoning performance in iterative learning. Figure 5 shows that SIPF and pRFT outperform SIOF and RFT, respectively.

4.6 Effect of Process Feedback on Reasoning

To explore whether process feedback can effectively reduce reasoning errors and improve the reasoning process, we follow Zheng et al. (2023) and use GPT-4 to automatically evaluate the rationales generated under different methods.

The GPT-4 is required to analyze the reliability of the reasoning process from six different perspectives: *Computational*, *Completeness*, *Reasonable*, *Concise*, *Consistent*, and *Overall Performance*.

Specifically, we prompt GPT-4 to compare and evaluate the rationales generated by SIPF and SIOF (as described in Section 4.5) to determine which method produces more reliable paths. To eliminate the influence of the final answer, we removed examples where the result is incorrect under both methods. The results in Figure 5 show that process-based SIPF outperforms outcome-based SIOF methods across multiple dimensions of rationales evaluation. For more details on the evaluation, please refer to Appendix B.

Method\Model	Gemma-2B	TinyLlama-v1.1
SIPF	38.15	16.22
w ITR	**43.97**	**18.57**
w/o PF	37.83	15.24
w/o OR	34.64	14.63
w/o PFOR	34.57	14.40

Table 4: Ablation study on GSM8K (Accuarcy). ITR denotes self-iterative learning (3 rounds), *PF* represents process feedback, *OR* refers to the relative ratio loss in the ORPO objective, and *PFOR* represents both process feedback and relative ratio loss.

4.7 Analysis of Performance across Different Reward Models

As described in Section 2.3, the fine-tuned reward models can be used to assign reward scores to each step in the rationales, thus constructing a reference dataset. Therefore, the accuracy of the reward model is crucial for enhancing SIPF's performance. However, recent works have shown a lack of analysis concerning the reward models.

To evaluate the performance of the reward model, we manually labeled a dataset specifically for assessing step correctness. Following Lightman et al. (2024), we train PRMs with step-annotated rationales and ORMs with result-annotated rationales. The fine-tuned reward models then assign correctness scores to each step in the evaluation dataset to assess accuracy. Please refer to Appendix C for more details on manual annotations.

Table 3 shows that, compared to ORM, PRM can more accurately evaluate the correctness of reasoning steps, consistent with previous work (Lightman et al., 2024). Additionally, the performance of the reward model is closely related to its language modeling capabilities (Cobbe et al., 2021). The stronger the model's reasoning ability, the better it performs in evaluations. Specifically, models like deepseek-math-rl-7B and deepseek-math-instruct-7B, which are aligned using large-scale, high-quality mathematical datasets, significantly outperform TinyLlama-v1.1 and Llama-2-7B (Touvron et al., 2023), especially in the case of PRM.

5 Ablation Analysis

We conduct an ablation analysis on different components of the proposed method to validate their effectiveness. Table 4 presents the ablation experiment results of Gemma-2B and TinyLlama-v1.1 on the GSM8K task, with SIPF using only one itera-

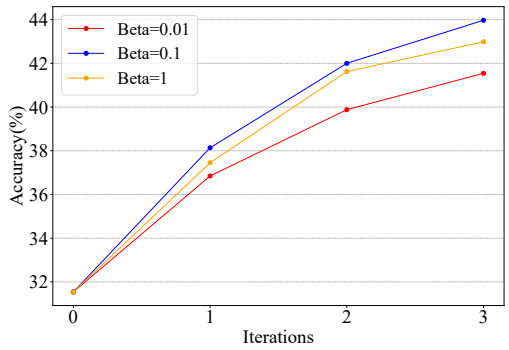


Figure 6: Comparison of performance with different weights of relative ratio loss across various iterations on GSM8K using Gemma-2B.

tion as the reference. It shows that performance degrades across different models when various components are removed, highlighting the necessity of each part. We also explore the performance differences with varying weight of relative ratio loss, as shown in Figure 6. The results indicate that an OR weight of around 0.1 is suitable for SIPF.

6 Conclusion

This work introduces a self-iterative process feedback method aimed at improving the reasoning abilities of SLMs, where process feedback is generated through reasoning simulation and verification, without relying on manual annotation. Extensive experimental results across multiple multi-step reasoning tasks demonstrate the effectiveness and generalizability of the proposed method.

Limitations

The proposed method has only been experimented on pre-trained models with sizes $\leq$ 2B, and its feasibility has not been validated on larger open-source models ($\geq$ 7B). Current experimental results only demonstrate the method's effectiveness on common multi-step reasoning tasks (e.g., mathematics and code) and do not confirm its applicability to a broader range of reasoning tasks. Additionally, the resource overhead of the self-iterative approach itself is significant and cannot be overlooked.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61966038 and 62266051, and the Postgraduate Practice and Innovation Foundation of Yunnan University under Grant No. ZC-242410094. We would like to thank the anonymous reviewers for their constructive comments.

References

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*.

Oliver Bentham, Nathan Stringham, and Ana Marasović. 2024. Chain-of-thought unfaithfulness as disguised accuracy. *arXiv preprint arXiv:2402.14897*.

Kaiyuan Chen, Jin Wang, and Xuejie Zhang. 2024a. Mathematical reasoning via multi-step self questioning and answering for small language models. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 81–93. Springer.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Duanyu Feng, Bowen Qin, Chen Huang, Zheng Zhang, and Wenqiang Lei. 2024. Towards analyzing and understanding the limitations of dpo: A theoretical perspective. *arXiv preprint arXiv:2404.04626*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. Deepseek-coder:

When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14852–14882. Association for Computational Linguistics.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: monolithic preference optimization without reference model. *CoRR*, abs/2403.07691.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron C. Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *CoRR*, abs/2402.06457.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8003–8017. Association for Computational Linguistics.

Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy F. Chen, and Shafiq Joty. 2024. Learning planning-based reasoning by trajectories collection and process reward synthesizing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 334–350, Miami, Florida, USA. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Kyungjae Lee, Dasol Hwang, Sunghyun Park, Youngsoo Jang, and Moontae Lee. 2024. Reinforcement learning from reflective feedback (rlrf): Aligning and improving llms via fine-grained self-reflection. *arXiv preprint arXiv:2403.14238*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adámek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1773–1781. Association for Computational Linguistics.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Tiesunlong Shen, Jin Wang, and Xuejie Zhang. 2025. Knowledge distillation via adaptive meta-learning for graph neural network. *Information Sciences*, 689:121505.

Tiesunlong Shen, You Zhang, Jin Wang, and Xuejie Zhang. 2023. Graphs get personal: learning representation with contextual pretraining for collaborative filtering. *Applied Intelligence*, 53(24):30416–30430.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T. Parisi, Abhishek Kumar, Alexander A. Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin

Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Trans. Mach. Learn. Res.*, 2024.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2717–2739. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

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

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

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

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2024a. How language model hallucinations can snowball. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024b. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

A Experimental Details

A.1 Dataset

GSM8K. GSM8K (Cobbe et al., 2021) is a challenging multi-step math reasoning task, where each sample consists of grade school math word problems annotated with step-by-step solutions. It contains 7,473 samples as the training set, while the test set consists of 1,319 samples.

MMLU_Math. MMLU (Hendrycks et al., 2021a) is a benchmark task designed to evaluate a model's ability to solve problems by applying world knowledge across various domains, including mathematics, law, and computer science. The MMLU_Math used in our experiments is a subset of MMLU. Specifically, it includes college, high school, and elementary school math problems from MMLU. Since all questions in the original task are multiple-choice, we only retain those where the correct answer is a standard number. After filtering, only 354 samples are used as the test set.

MBPP. MBPP (Austin et al., 2021) consists of 1,000 entry-level Python programming problems. Each problem includes a task prompt, a code solution, and 3 test cases. We use a training set containing 374 samples and a test set with 500 samples in

Hyperparameters	Setting
learning_rate	5e-5, 8e-5, 1e-4
optimizer	AdamW
warmup_ratio	0.1
max_grad_norm	0.3
OR_weight	0.01, 0.1, 1
bf16	True
lora_alpha	128
lora_dropout	0.05
lora_r	256

Table 5: Unified parameter settings for self-iteration alignment across different tasks.

the experiments. Notably, because different combinations of test cases can be included in the prompts, the actual number of training samples used for fine-tuning is 2,244. For testing, to prevent label leakage, we selected 2 out of 3 test cases as prompts for each problem, resulting in 6 different input formats per question. Under these conditions, we evaluated the performance on MBPP using Pass@k metrics.

HumanEval. HumanEval (Chen et al., 2021) contains 164 entry-level programming problems to assess understanding, arithmetic, and reasoning. We use this dataset to evaluate the model's out-of-domain generalization capabilities on code generation. Additionally, to maintain consistency with the format of MBPP, we transform the input format of Humaneval problems. Specifically, the input for a problem is changed from *Function_Name+Prompt+Test_Case* to *Prompt+Test_Case+Function_Name*.

A.2 Experiment Setup

Training. For efficiency considerations, we use Unsloth in our experiments and apply QLoRA for parameter-efficient fine-tuning. All methods are based on algorithms and models implemented in the transformer (Wolf et al., 2020) and trl[2] libraries. Table 5 shows the hyperparameter settings used for different tasks and models. When training Gemma-2B, we set the batch size to 20, while the batch size is 48 for TinyLlama-v1.1 and Phi-1.5. Due to the complexity of the self-iteration learning, we select the best performance under a predefined set of learning rates and relative ratio loss weights from Table 5 and leave the search for optimal parameters for future work. We set the batch size to 40 for training the reward model and used a low learning rate 5e-5.

[2] https://github.com/huggingface/trl

Iteration\Model	Gemma-2B	Phi-1.5
Iter-0	2.24	2.24
Iter-1	13.12	18.35
Iter-2	20.06	30.43
Iter-3	31.27	39.87

Table 6: Training data size (k) obtained through sampling and filtering by reward values at different iteration rounds on MBPP.

Iteration\Model	Gemma-2B	TinyLlama-v1.1
Iter-0	7.47	7.47
Iter-1	37.78	36.96
Iter-2	55.24	50.49
Iter-3	87.41	68.94

Table 7: Training data size (k) obtained through sampling and filtering by reward values at different iteration rounds on GSM8K.

Sampling. In each iteration, temperature sampling ($T = 1$) wil be used to collect enough positive and negative samples (reasoning paths), labeling them based on the correctness of the final answer for math problems or test passes for program problems. Specifically, we specify the required number of positive and negative samples per question. Based on subsequent rounds, the number will be set to half or a quarter of the number of positive and negative samples from previous rounds. Since not all math problems yield the required number of samples through sampling, we set a time threshold for sampling to improve efficiency. For code generation problems, samples are labeled post-sampling, so no time threshold is set. Tables 6 and 7 show the count of sampled datasets obtained after filtering at different iteration rounds.

A.3 Baseline

CoT. On GSM8K and MMLU_Math, we used the same few-shot prompt as in the previous work (Wei et al., 2022), while the zero-shot prompt is used for MBPP and HumanEval.

SFT. The models are fine-tuned on the existing training sets of GSM8K and MBPP.

STaR. STaR (Zelikman et al., 2022) is a Self-Taught method that enhances reasoning by iteratively learning from self-generated rationales, with each rationale generated through greedy decoding. Due to the inclusion of low-quality samples, we did not adopt the rationalization from prior work (Zelikman et al., 2022). With a low learning rate

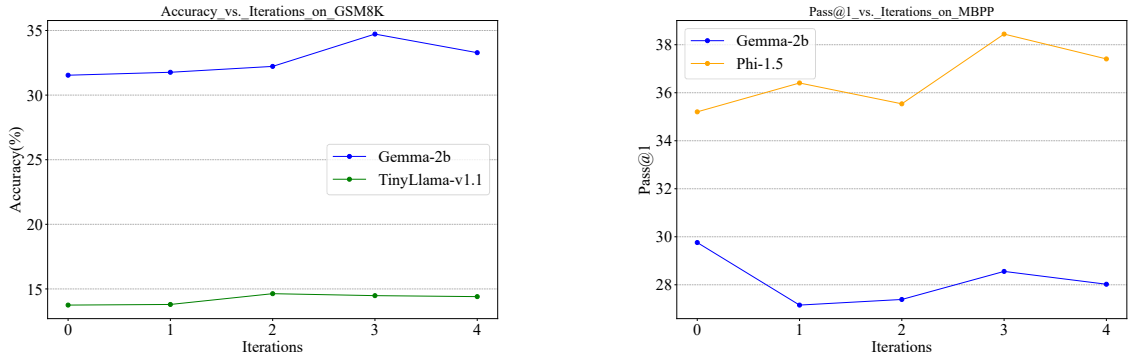


Figure 7: The performance of STaR at different iteration rounds on GSM8K (Acc) and MBPP (Pass@1). Iteration-0 represents supervised fine-tuning on the original dataset.

of 5e-5, the performance of models converges after 4 iterations as shown in Figure 7, and we selected the best performance from different iterations.

RFT. Rejection Sampling Fine-Tuning (RFT) (Yuan et al., 2023) enhances the training dataset by self-sampling from a supervised model, with each sample filtered for uniqueness in its reasoning path. Unlike STaR, RFT does not use self-iterative optimization. Our experiments set the sampling temperature to 1 and increased reasoning diversity through path deduplication for all tasks.

SRF. Compared to Self-Taught, Self-Refine (Yuan et al., 2024; Lee et al., 2024) focuses on correcting errors through negative feedback. We apply DPO to align the supervised model using the first round of sampling dataset from SIPF. Due to resource constraints, we only classify samples based on whether the final answer is correct (for math) or if it passes test cases (for code).

RPO. Iterative Reasoning Optimization (RPO) (Pang et al., 2024) improves reasoning performance through iteration by combining negative log-likelihood loss on positive examples with a DPO learning objective on both positive and negative examples. However, this method is still based on outcome feedback. In the experiments, we applied RPO to the first round of sampling datasets from SIPF.

pRFT and pSRF. Original RFT and STR optimize solely based on outcome feedback. Therefore, we use a process reward model to further filter the feedback results, meaning only samples with high reward values are used for model optimization.

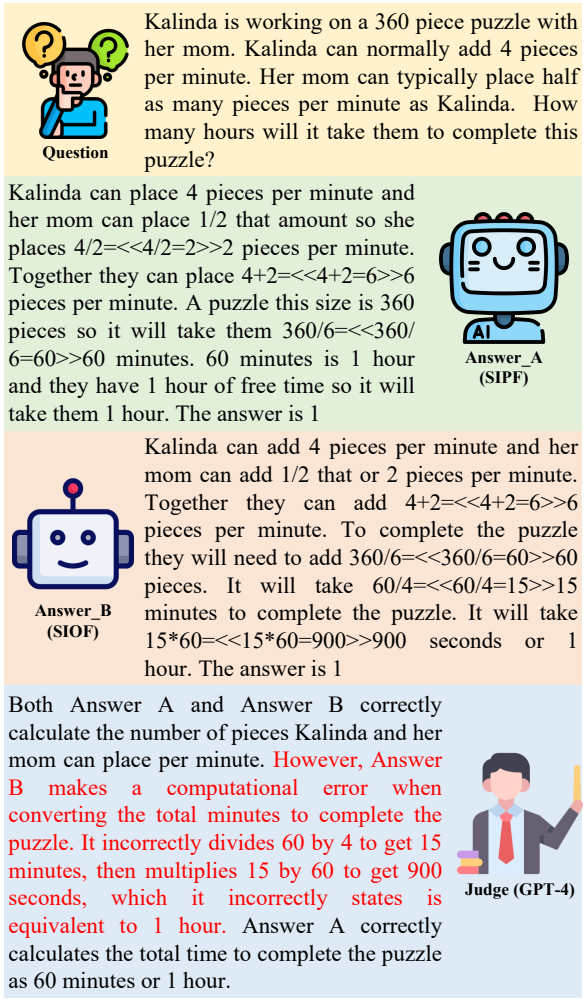


Figure 8: The case study of automatic evaluation of rationales. With prompting, GPT-4 identifies the reasoning error in Answer_B, even though both Answer_A and Answer_B have the same correct results.

B Automatic Evaluation of Rationales

Due to the advanced capabilities, powerful LLMs like GPT-4 can align closely with human preferences, allowing them to serve as judges in various evaluation tasks (Zheng et al., 2023). To evaluate the rationales generated by different methods, we

Please act as an impartial judge and evaluate the quality of the answer to the question displayed below. You will be given a question, a standard answer, answer A, and answer B. Your job is to evaluate which answer is better according to standard answer. Identify and correct any mistakes. Avoid any position biases and ensure that the order in which the answers were presented does not influence your decision. Do not favor certain names of the answers. Be as objective as possible. Your evaluation should consider following standards:

Standards:
Computational Correctness: The computations involved in the answer are correct.
Completeness: The answer is complete without missing any necessary steps.
Reasonable: The answer is reasonable, which means each conclusion should be inferred by collecting evidence, instead of making up unknown facts.
Concise: The answer should not tell something irrelevant to the question.
Consistent: There must not be a contradiction in the answer itself.
Overall Performance: the final performance, considering all the above aspects

After providing your explanation, for each aspect of the above standards, select one winner (A or B), or judge it as a tie. Note that each aspect should be evaluated independently, following the format:

Explanation: [Your Explanation]
Computational Correctness: A/B/Tie
Completeness: A/B/Tie
Reasonable: A/B/Tie
Concise: A/B/Tie
Consistent: A/B/Tie
Overall Performance: A/B/Tie

Question
{{question}}

Standard Answer:
{{ref_answer}}

Answer A:
{{answer_a}}

Answer B:
{{answer_b}}

Figure 9: In the GPT-4-based automatic evaluation of rationales, the prompt templates provide definitions for six different evaluation dimensions and ask GPT-4 to determine which answer, A or B, is better or equivalent, after providing the relevant explanations.

used GPT-4 as the judge. Specifically, GPT-4 was given a question from GSM8K along with two different answers, and with appropriate prompting, it determined which response was better. Building on the definitions of reasoning errors and validity from previous work (Jiao et al., 2024; Wang et al., 2023a), we compared the rationales across six different dimensions: Computational, Completeness, Reasonable, Concise, Consistent, and Overall Performance. The definitions of each dimension and the prompt templates used are shown in Figure 9. Additionally, we provide a case study of judgments from GPT-4 in Figure 8.

C Manual Annotation of Rationales

To explore the alignment between the reward model and human standards, we collected 1,000 rationales from GSM8K (all generated by model sampling) and distributed them to different human evaluators, asking them to annotate the correctness of each step in the rationales. Specifically, we provided four criteria for judging the correctness of reasoning steps, as shown in the annotation guidelines in Figure 10. After removing some invalid labeled samples, we retained 791 samples for evaluating the reward model.

D Related Work

Chain-of-thought Reasoning. Previous work (Wei et al., 2022) has shown that by inserting a few Chain-of-Thought examples into the input context or by simply adding *Let's think step by step* (Kojima et al., 2022) after the question, pre-trained LLMs can be guided to perform multi-step reasoning, thereby enhancing reasoning performance. Recently, guiding models to perform multi-step reasoning has also been deliberately introduced into open-source language models through supervised fine-tuning (Cobbe et al., 2021; Chung et al., 2024) or knowledge distillation (Magister et al., 2023; Ho et al., 2023; Hsieh et al., 2023; Shen et al., 2023; Chen et al., 2024a; Shen et al., 2025).

Please annotate the correctness of each step for sampled rationales in math problems. Descriptions of some fields in the samples are as follows:

{
"idx": Problem ID
"question": Problem statement
"gold answer": Standard rationale
"sample_answer": Sampled rationale
"gold_ans": Standard final answer
"sample_ans": Sampled final answer
"sample_answer_step_correctness": Correctness of each step in the sampled rationale
[
"step_idx": Step number in the sampled rationale
"human_ans": Correctness of the step as judged by human evaluators (0/1)
]
}

The criteria for judging the correctness of reasoning steps are as follows:
1. Operational Correctness: The current step should involve correct basic calculations.
2. Factual Accuracy: The current step should not contradict the problem description. Any fabrication, distortion, or deviation from the facts is incorrect.
3. Logical Consistency: The current step should maintain logical coherence with previous reasoning steps or the problem itself. Any contradictions in the reasoning are incorrect.
4. Reasoning Completeness: If the current step is missing any key steps necessary for complete reasoning, the reasoning is considered incomplete.

If a reasoning step meets the above criteria, it is considered correct; otherwise, it is incorrect. Please mark the correctness of the current step with 0 (incorrect) or 1 (correct) in the **human_ans** field.

Figure 10: The guideline for annotating reasoning steps in sampled rationales from GSM8K. The guidelines provide four criteria for evaluating the correctness of indirect steps, including: *Operational Correctness*, *Factual Accuracy*, *Logical Consistency*, and *Reasoning Completeness*.

However, such methods often rely on expensive labeled data from humans or LLMs, which limits the scalability of the approach to some extent. The limited data availability constrains the further enhancement of the model's capabilities (Chen et al., 2024b), reducing its generalization (Yuan et al., 2023) and potentially leading to hallucination issues (Rawte et al., 2023).

Self-Training Reasoning. To overcome the limitations of manual annotation, STaR (Zelikman et al., 2022) was the first to propose leveraging the inherent language modeling capabilities of pre-trained language models to improve reasoning. Specifically, STaR fine-tunes the model on self-generated rationales that yield correct results, and this process is iteratively repeated several times. Unlike STaR, RFT (Yuan et al., 2023) believes that high-temperature sampling from pre-trained models can significantly enhance the diversity of generated rationales, and diverse reasoning approaches are crucial for improving the model's reasoning performance. However, RFT does not emphasize improving reasoning through iterative training. Building on the previous works, ReST^{EM} (Singh et al., 2024) further describes self-training as an expectation-maximization reinforcement learning (RL) process. These Self-Taught methods only learn from correct solutions and do not consider the impact of incorrect solutions.

Benefiting from preference alignment techniques (e.g., DPO (Rafailov et al., 2023)), recent works (Yuan et al., 2024; Chen et al., 2024b) have attempted to use self-generated positive and negative samples to improve the instruction-following capabilities of LLMs, where LLMs themselves typically assign each sample a label. For reasoning tasks, V-STaR (Hosseini et al., 2024) proposes using DPO to train a outcome-supervised verifier (or ORMs) on self-sampled positive and negative examples and combining this verifier to enhance the reasoning performance of self-iterative models. Additionally, RPO (Pang et al., 2024) enhances reasoning per-

formance through iterative learning by combining negative log-likelihood loss on positive examples with a DPO learning objective that applies to both positive and negative examples. However, these works still rely on outcome feedback.

Learn Reasoning from Process Feedback. Due to the inherent complexity of reasoning and the potential lack of fidelity in language models, the correctness of the result often does not align with the reasoning process (Zhang et al., 2024a; Bentham et al., 2024). This means that a reasoning process that yields a correct answer may contain erroneous steps, while a reasoning process that leads to an incorrect answer can also include correct steps. This uncertainty limits further improvements in the reasoning performance of language models.

Based on these considerations, Lightman et al. (2024) introduced the Process Supervision Reward Model (PRM), which is trained on reasoning paths with manually annotated step correctness and used for evaluating sampled paths. To avoid the tedious and costly manual annotation process, recent works (Jiao et al., 2024; Wang et al., 2024) have used path simulation in Monte Carlo Tree Search to assign labels for the correctness of each step. Alternatively, fine-grained feedback for the reasoning process is introduced through the self-reflective capabilities of advanced LLMs (Lee et al., 2024), such as GPT-4. Our work is based on assigning correctness to each step through inference simulations and improving reasoning performance through a self-iterative optimization process.