Mitigating Feedback Inconsistency in Large Language Model Self-Tuning

Anonymous ACL submission

Abstract

Large language models have demonstrated various abilities, i.e. Chain-of-Thought reasoning for Math Reasoning datasets. Can models learn to self-improve these skills? First, we statistically analyzed the potential of the self-evaluation ability of language models. Then, we present a novel self-tuning framework, `STC`, that leverages reinforcement learning to enhance reasoning capabilities in large language models. `STC` encourages the generation of logical explanations by evaluating the greedy decoded responses against the diverse sampled responses. Results highlight the effectiveness of our framework across various model sizes (1B-20B). We observe improvements in the accuracy of up to 5% on four different math reasoning datasets, simultaneously improving commonsense ability and retaining language understanding ability. Additionally, human and machine evaluation confirms the quality of the generated responses became more detailed and logical after training.

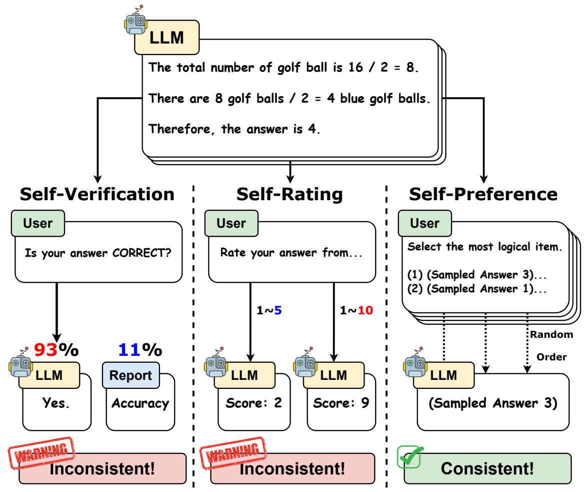

Figure 1: Comparison between self-verification (Madaan et al., 2023), self-rating (Pang et al., 2023), and self-preference (proposed method) as self-evaluation framework of large language models.

1 Introduction

Developing reasoning systems has long been a fundamental goal in the field of Artificial Intelligence (McCarthy, 1959). Reasoning systems can determine the solutions to complex problems through logical justification. With the advent of large language models and their success in generating human-like text (Brown et al., 2020), researchers started to exploit language models' reasoning ability to solve logical problems. This includes but is not limited to Chain-of-thought (CoT) (Wei et al., 2023; Kojima et al., 2023) or Tree-of-Thought (Yao et al., 2023), which have shown that step-wise thinking allows more accurate responses in reasoning tasks.

Meanwhile, researchers have actively applied reinforcement learning (RL) on language models to align their behavior with human preferences (Ouyang et al., 2022; Ziegler et al., 2020) or AI preferences (Bai et al., 2022; Lee et al., 2023). This approach has led to the development of models which can follow complex instructions such as ChatGPT[1], Claude[2], and Stable Vicuna [3]. In contrast to auto-regressive token-wise objectives, RL allows the model to be fine-tuned according to a scalar reward that can be configured according to a desired direction, which does not require the existence of the gold answer.

Arithmetic reasoning task datasets typically provide the question and the target answer but often lack explicit reasoning chains. This absence of reasoning paths is due to the numerous possible ways to arrive at the same final answer, rendering the question open-ended and allowing for multiple valid solutions. This characteristic has been highlighted in previous studies focusing on prompt en-

[1] `https://openai.com/blog/chatgpt`

[2] `https://www.anthropic.com/index/introducing-claude`

[3] `https://stability.ai/blog/stablevicuna-open-source-rlhf-chatbot`

gineering for chain-of-thought reasoning (Fu et al., 2023c; Wang et al., 2023a). Taking advantage of the unsupervised setting in RL, we propose a novel self-guided approach that does not require any external tools or human annotations, Self-Tuning with Choice (STC). Our working assumption is that intensifying the pre-trained language model's inherent reasoning ability through a self-guided manner yields more logical and accurate reasoning.

Our paper addresses two fundamental research objectives: 1) making chain-of-thought reasoning in instruction-tuned language models informative and explanatory; 2) enabling more accurate reasoning in arithmetic reasoning tasks with unlabeled data. And our proposed method, Self-Tuning with Choice (STC), fine-tunes the instruction-tuned language model as an active policy with RL using a dual reward function consisting of a self-logicality reward and a QA similarity reward.

We conducted experiments using STC framework on arithmetic reasoning task where the goal is to predict the final answer by generating a rationale given question. We use four different math datasets with varying difficulty levels and eight models ranging from 1B to 20B. As far as we know, it is the first trial of applying a self-reinforcement strategy for arithmetic reasoning tasks with unlabeled data. Our experimental results demonstrated that this framework improves the accuracy up to around 5% in both multi-choice (AQUA) and descriptive questions (SVAMP). Furthermore, STC leads up to a 6% accuracy improvement in non-arithmetic tasks, like CommonsenseQA. Moreover, both machines and humans consistently favoured the response generated from the model trained with STC, particularly regarding logical coherence. The summarized contributions of our work are presented as follows:

1. We study the use of language models as a reward function in self-evaluation frameworks finding limitations with consistency.

2. We introduce a novel reinforcement learning approach that enables self-tuning with consistent feedback (STC).

3. We quantitatively and qualitatively validate that the models fine-tuned with STC make more explanatory and accurate responses.

2 Related Works

Large Language Model Self-Supervision The self-supervision capability of large language models has been studied recently on behalf of previously introduced works (Huang et al., 2022; Pang et al., 2023; Madaan et al., 2023; Ye et al., 2023).

As large language models can be used to elicit zero-shot chain-of-thought reasoning, Huang et al. (2022) sampled diverse responses from the model and selected the most probable answer by hard-voting to supervise itself. Madaan et al. (2023) improved the accuracy on non-arithmetic reasoning tasks by querying the models to refine the initial responses of itself. Ye et al. (2023) expanded this approach by conducting self-revision in a single inference stage to improve the response quality.

Pang et al. (2023) applied reinforcement learning on self-supervision by using a large language model as a reward model to score the response. However, it is essential to note that while (Pang et al., 2023) and (Madaan et al., 2023; Ye et al., 2023) applied self-improvement methods on large language models, they reported limitations specifically in math reasoning datasets.

Chain-of-Thought Reasoning Chain-of-thought (CoT) reasoning represents a sequence of sentences with step-wise explanations which contribute to reaching the final answer (Wei et al., 2023; Kojima et al., 2023). Wei et al. (2023) made large language models generate step-by-step explanations in a few-shot manner and significantly improved reasoning tasks. Kojima et al. (2023) showed that a simple prompt *"Let's think step-by-step."* can enable CoT reasoning in a zero-shot setting. After the emergence of CoT reasoning, there have been many attempts to increase the accuracy of large language models by utilizing emergent CoT reasoning abilities of large language models (Zhang et al., 2022; Wang et al., 2023a; Zhou et al., 2023; Wang et al., 2023b; Du et al., 2023). Zhang et al. (2022), Wang et al. (2023a) and Zhou et al. (2023) showed the importance of prompting by demonstrating that design of prompts in zero-shot and few-shot settings further improves CoT reasoning abilities of large language models. Meanwhile, Du et al. (2023) and Wang et al. (2023b) used the CoT reasoning responses as part of the bigger pipeline for achieving more accurate and precise reasoning. Unlike these previous methods, we aim to propose a new self-fine-tuning method rather than relying on prompt engineering.

Reinforcement Learning for Language Models RL has recently shown significant success in field of NLP (Wu et al., 2016; Wu and Hu, 2018; Jang et al., 2022; Ouyang et al., 2022; Rafailov et al., 2023). Prior to the recent success, there were approaches to improve the generation quality in downstream tasks such as machine translation (Wu et al., 2016) and summarization (Wu and Hu, 2018). While these works used pre-defined reward functions, (e.g. BLEU (Papineni et al., 2002), ROUGE (Lin, 2004)), OpenAI trained a new reward model with human preference data to leverage the human feedback as the reward signal and fine-tuned large language models with RL (Ziegler et al., 2020; Stiennon et al., 2022; Ouyang et al., 2022). Furthermore, Rafailov et al. (2023) merged the reward model training and the language model fine-tuning into a single stage by converting it into a classification task on human preference data. While RL allowed notable improvements in the general abilities of language models, fine-grained human-annotated data are still required for either training the reward model or directly fine-tuning the policy.

3 Can LLMs Evaluate Themselves?

In this section, we empirically show the capabilities of language models as a consistent logicality checker. Under the criteria of consistency and preciseness, we test three different self-evaluating prompts shown in Figure 1. The first two methods are *self-verification* and *self-rating*, which were proposed by Madaan et al. (2023) and Pang et al. (2023). Identifying limitations in LLM consistency in these previous works motivates our proposed method, which incorporates self-checking with multi-choice preference.

Self-Verification We queried models to evaluate whether their response is correct with the template shown in Appendix B.1. The models were asked yes/no questions for the given question and answer pair. As shown in Table 1, every model tells *yes* in the majority of cases, which contradicts the baseline accuracy shown in Table 1. Miscalculation problems of language models (Yuan et al., 2023; Imani et al., 2023) also support the inadequacy of self-correction as a logicality-checking mechanism.

Self-Rating We assess the consistency of language models as scorer, which was proposed by Pang et al. (2023). We queried models with the template shown in Appendix B.2 to rate their responses in two different scales: 1) from 1 to 5 and 2) from 1 to 10. We apply the Mann-Whitney U Test (Mann and Whitney, 1947) where the null hypothesis is: *The population distribution of the scores from a scale of 1 to 5 and 1 to 10 are the same.*

	Test Accuracy (%)	Self-Accuracy (%)
Alpaca (7B)	4.17	91.96
Vicuna (13B)	11.07	92.65
FLAN-T5-XXL (11B)	17.36	98.18
FLAN-UL2 (20B)	24.71	99.32

Table 1: Comparison between true accuracy on GSM8K and the self-verification result. 'Self-Accuracy' denotes the proportion of 'yes' in self-verification results.

As the P-value in Table 2 shows, we can conclude that none of the tested models consistently evaluate with interval scale scoring.

	Statistics	**P-Value** ($\alpha = 0.05$)
Alpaca (7B)	382615.5	2.67e-160
Vicuna (13B)	2058.0	6.0394e-19
FLAN-T5-XXL (11B)	1211065.5	1.90e-105
FLAN-UL2 (20B)	999689.0	5.08e-35

Table 2: Mann-Whitney U-Test results for self-rating.

Self-Preference We measure the consistency of preference in multi-choice question setting through both specific examples and general metrics. For specific examples, we sample diverse responses on a single question and query the model to select the most logical response with the example shown in Appendix B.3. Four sampled CoT responses were given as choices. To prevent the order bias in options, we also queried the model with reversed orders. Figure 2 shows that the model has a consistent preference for a single, more logical option regardless of the order of options.

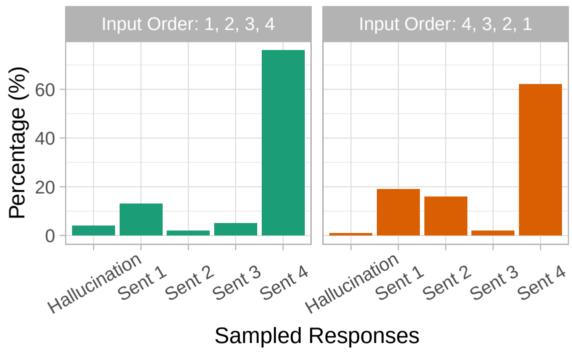

Figure 2: Example from a single instance showing that the models are not sensitive to the ordering of options in multi-choice preference evaluation.

Also, we measured Cohen's κ (Cohen, 1960) of FLAN-T5-XXL and FLAN-UL2 on GSM8K by randomizing the order of items in a multi-choice question setting. With the sampled responses from the GSM8K test set, FLAN-T5-XXL and FLAN-UL2 got 0.443 and 0.498, respectively, indicating reasonable consistency. Meanwhile, this also highlights the limitation in evaluation with LLMs without self consistency evaluation.

Comparison Self-checking mechanisms from previous works showed either inconsistent or consistent but unreliable results. Even though Madaan et al. (2023) suggested a consistent and reliable self-checking mechanism, it required a large size as they have shown the limitation of their method with Vicuna 13B (Chiang et al., 2023). Also, Rafailov et al. (2023) fine-tuned the policy to align its preference to human preference with RL on text generation tasks, but their method still required fine-grained human-annotated data. On the other hand, asking the preference through multi-choice questions allowed both consistent and reliable self-checking with small instruction-tuned models. In this vein, we propose a novel framework STC for reasoning tasks relying on the preference of the instruction-tuned models in terms of logicality.

4 Methodology

STC employs Reinforcement Learning (RL) to enhance the reasoning abilities of the instruction-tuned language models by searching for more logical responses than greedy decoding through sampling. This is further explained in Section 4.2.

4.1 Preliminary

The main objective of RL is to train the policy to maximize the expected return of the rewards, defined in Equation 1:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s,a,r}\left[\sum_{t=0}^{\tau} \gamma^t R(s_t, a_t)\right] \quad (1)$$

where π represents the policy initialized from the parameters of the pre-trained language model, s_t denotes the space of input token sequences, a_t represents the token generated by the language model, r_t is the reward received by the language model at time t, and γ is the discount factor, and τ represents the generated sequence length.

We define the active model π^{Act} as the model that will be trained, and the reference model π^{Ref} as the original model that would be compared with π^{Act} during training. π^{Act} and π^{Ref} will get an input $x_i = [Q_i; p]$ where Q_i denotes each question and p denotes the fixed zero-shot chain-of-thought prompts (e.g. *Let's think step-by-step.*) (Kojima et al., 2023). Each model will return the response $A_{\pi^{Act}}$ and $A_{\pi^{Ref}}$ respectively.

4.2 Framework

STC operates by first generating zero-shot CoT responses by appending the prompt p, *Answer the question by reasoning step-by-step.* π^{Ref} and π^{Act} generates $A_{\pi^{Ref}}$ and $A_{\pi^{Act}}$ respectively, using greedy decoding and top-p sampling (Holtzman et al., 2020). First, we query π^{Act} to select the more logical answer and give the reward of 1 if $A_{\pi^{Act}}$ is selected and 0 otherwise. In addition, we give an explanatory reasoning reward by comparing the bi-gram overlap between Q and $A_{\pi^{Act}}$ using ROUGE (Lin, 2004). Finally, this dual reward function, comprising the *Self-Logicality Reward* and *QA Similarity Reward*, determines the final reward $R(Q, A_{\pi^{Act}}, A_{\pi^{Ref}})$ for $A_{\pi^{Act}}$. This reward is then utilized to optimize the policy through the use of PPO.

4.3 Reward Design

We propose a dual reward function which serves two different purposes: logicality and informativeness of the model's responses. The full reward function $R(Q, A_{\pi^{Ref}}, A_{\pi^{Act}})$ is as follows:

$$\mathbb{I}_L \times \mathrm{Sim}(Q, A_{\pi^{Act}}) \quad (2)$$

where $\mathbb{I}_L$ and $\mathrm{Sim}(Q, A_{\pi^{Act}})$ refer to self-logicality and QA similarity reward respectively.

Self-Logicality Reward We rely on the internal reasoning capabilities of large language models for rewarding. Self-logicality reward is an indicator function that can be written as follows:

$$\mathbb{I}_L = \begin{cases} 1 & \text{if } \mathit{Logi}(A_{\pi^{Act}}, A_{\pi^{Ref}}) = A_{\pi^{Act}} \\ 0 & \text{if } \mathit{Logi}(A_{\pi^{Act}}, A_{\pi^{Ref}}) = A_{\pi^{Ref}} \end{cases} \quad (3)$$

where $\mathit{Logi}(A_{\pi^{Act}}, A_{\pi^{Ref}})$ refers to the choice of π^{Act} in evaluating the logicality between $A_{\pi^{Act}}$ and $A_{\pi^{Ref}}$. Self-logicality reward tells which reasoning paths should be further explored or pruned. The detailed dichotomous question prompt template and its example are in Appendix C.

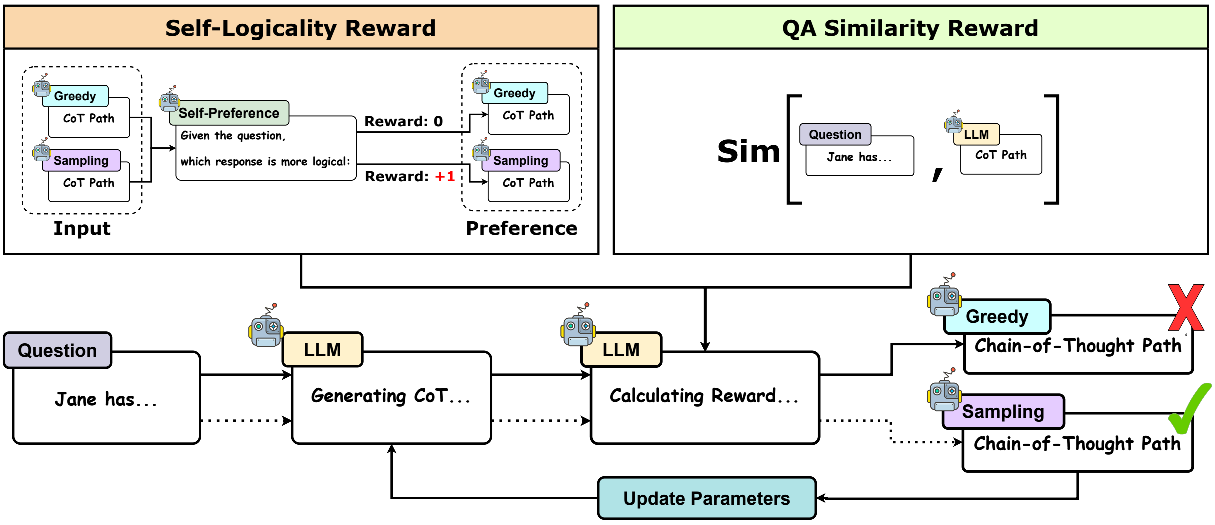

Figure 3: The pipeline of self-tuning with joint reward function by comparing sampled and greedy decoded chain-of-thought reasoning path. We sample the reasoning paths to find the paths which can be potentially more logical than the greedy decoded path. The dashed line and dotted line refer to greedy decoding and sampling respectively.

QA Similarity Reward QA Similarity Reward is driven by the nature of arithmetic tasks, where the necessary information for problem-solving is present within the question itself. Therefore, QA Similarity Reward measures how well the answer incorporates those clues.

$$\text{Sim}(Q, A_{\pi^{Act}}) \tag{4}$$

We measure the text similarity between the question and the response from the active model, which indicates how well π^{Act} is using the key information given in the question. We empirically justified this in Appendix A, which led us to choose ROUGE-2 as a QA similarity metric.

5 Experimental Design

In this section, we provide a detailed explanation of the datasets (5.1) and language models (5.2) that were utilized for both training and testing purposes. We also provide a concise overview of the implementation details in Appendix (D) employed during training and testing our method.

5.1 Datasets

We use GSM8K (Cobbe et al., 2021), AQUA (Ling et al., 2017) , MultiArith (Roy and Roth, 2015), and SVAMP (Patel et al., 2021) which consist of school-level arithmetic problems for our experiments. We train our model on the GSM8K train dataset and evaluate the trained model on the test dataset of GSM8K, AQUA, MultiArith, and SVAMP. As shown in Table 3, GSM8K is the hardest dataset among them, so we expected the model to generalize arithmetic reasoning ability through GSM8K training.

Moreover, we use a multitask language understanding dataset MMLU (Hendrycks et al., 2021), which consists of a wide range of problems (e.g. history, medicine, etc.), to measure the general ability of the trained model. Following Chia et al. (2023), we evaluate the model with 5-shot prompting on 57 different tasks.

5.2 Models

We use four encoder-decoder models and four decoder-only models, which are instruction-tuned. For encoder-decoder models, we use FLAN models, including FLAN-T5-Large, XL, XXL, and FLAN-UL2 (Tay et al., 2023). For decoder-only models, we use Vicuna (Chiang et al., 2023), Alpaca (Taori et al., 2023), and Falcon[4] (Almazrouei et al., 2023). Along with model architecture, we note that encoder-decoder models were specifically pre-trained on arithmetic reasoning datasets, and decoder-only models were not.

6 Results

We evaluated our method both quantitatively and qualitatively. First, we quantified the zero-shot and few-shot reasoning capabilities of the trained model on all four datasets in Section 6.1. Then, in

[4] https://huggingface.co/tiiuae/falcon-40b, the related publication not provided.

			Specialized				Generalized	
			GSM8K	AQUA	MultiArith	SVAMP	MMLU	Common
Random Choice			0%	20%	0%	0%	25%	20%
Encoder-Decoder								
FLAN-T5-Large	1B	Baseline	5.91%	22.83%	13.33%	7.00%	**41.94%**	82.55%
		STC	**6.90%**	**23.23%**	**15.00%**	7.00%	41.85%	**82.80%**
FLAN-T5-XL	3B	Baseline	**11.75%**	24.02%	23.33%	16.00%	49.27%	86.07&
		STC	10.54%	**27.17%**	**25.56%**	**20.67%**	**49.34%**	**86.65%**
FLAN-T5-XXL	11B	Baseline	**17.36%**	**28.74%**	51.11%	31%	**54.54%**	84.02 %
		STC	16.53%	25.59%	**53.33%**	**32.33%**	54.52%	**84.11%**
FLAN-UL2	20B	Baseline	24.71%	21.26%	64.44%	32.33%	55.13%	89.10%
		STC	**26.31%**	**23.62%**	**67.22%**	**33.67%**	**55.23%**	**89.10%**
Decoder-Only								
Alpaca	7B	Baseline	4.17%	20.47%	**8.89%**	21.67%	40.23%	27.93%
		STC	4.17%	**25.98%**	8.33%	**25.67%**	**40.24%**	**33.08%**
Falcon	7B	Baseline	5.31%	20.47%	**26.11%**	16.67%	25.37%	20.88%
		STC	**6.98%**	**22.44%**	22.22%	**17.67%**	**25.41%**	**21.04%**
Vicuna	7B	Baseline	7.88%	23.62%	**26.66%**	33.33%	44.73%	35.87%
		STC	**9.10%**	**24.41%**	24.44%	**35.00%**	**44.89%**	**37.01%**
Vicuna	13B	Baseline	11.07%	26.38%	43.33%	31.33%	51.26%	43.16%
		STC	**11.37%**	**28.35%**	**44.44%**	**32.67%**	**51.33%**	**43.98%**

Table 3: Accuracy table on four math reasoning datasets, general language understanding (MMLU), and common sense reasoning (CommonSenseQA). Across eight models, results show consistent improvement across all datasets, with some anomalies in the MultiArith dataset explained in Section 8.

Section 6.2, we assessed if the general language understanding abilities were retained after self-tuning. Furthermore, we qualitatively validate if our method strengthened the logicality and explanatory reasoning of instruction-tuned models through both human and machine evaluation in Section 6.3.

6.1 Specialized: Arithmetic Reasoning

For the arithmetic reasoning task, we compare the accuracy of the baseline and self-tuned version of eight instruction-tuned models shown in Table 3. In the case of the FLAN series, which are the encoder-decoder models, GSM8K and AQUA were partially used for instruction-tuning (Chung et al., 2022). Every dataset was unseen for the rest of the cases.

Zero-Shot CoT The encoder-decoder models showed improvements in every case on unseen datasets, which are MultiArith and SVAMP. The smallest model, FLAN-T5-Large showed the least improvement, and the models with the most improvement varied. Specifically, FLAN-UL2 increased by 3.22% on MultiArith, and FLAN-T5-XL increased by 4.67% on SVAMP. Meanwhile, for the seen datasets, FLAN-T5-Large and FLAN-UL2 improved in GSM8K, and all the models except for FLAN-T5-XXL improved in AQUA.

On the other hand, the decoder-only models improved in every case for GSM8K, AQUA, and SVAMP. For example, Falcon increased the most on GSM8K by 1.67%, and Alpaca increased the most on AQUA and SVAMP by 5.51% and 4%, respectively. While Vicuna 13B was the only model that improved in MultiArith.

Overall, for FLAN-UL2 and Vicuna 13B, which are the largest models in the encoder-decoder models and decoder-only models, the accuracy increased in all four datasets.

Few-Shot CoT To further evaluate the effectiveness of our method, we assess the few-shot chain of thought (CoT) reasoning abilities of the trained models. For this evaluation, we selected four models, Alpaca, Vicuna-13b, FLAN-T5-XXL, and FLAN-UL2, and tested on the GSM8K dataset using few-shot examples from (Fu et al., 2023a). The results are shown in Table 4.

		Few-Shot CoT	
Models		Baseline	STC
Alpaca	7B	5.61%	5.61%
Vicuna	13B	21.99%	**25.17%**
Flan-T5-XXL	11B	**16.76%**	16.24%
FLAN-UL2	20B	26.23%	**26.61%**

Table 4: Accuracy table for Few-Shot Chain-of-Thought reasoning on GSM8K dataset.

We observe that both trained models demonstrated the same pattern in few-shot CoT reasoning as zero-shot CoT. STC-trained Alpaca performed the baseline, and few-shot FLAN-T5-XXL decreased performance just like Zero-Shot. However, it is noteworthy that the accuracy was improved for all of the models in comparison to zero-shot, which is expected in a few shot settings.

6.2 Generalized: Language Understanding

MMLU We also report the accuracy shift of the models on MMLU in Table 3. As Fu et al. (2023b) analyzed the trade-off in specializing the language models on certain tasks and generalized ability, we analyzed the shift in general ability after training with `STC`. As shown in Table 3, all the models were improved in generalization ability in the decoder-only models, while FLAN-T5-Large and XL showed a fraction of a percent decrease in the encoder-decoder models. Overall, the trained models retained their general ability.

We attribute this to using PEFT, which updates the partial parameters during training. Additionally, the adaptive KL regularization of rewards and the effect of the clip, which are components of PPO, contribute to preventing significant deviations of the active model from the reference models.

Commonsense Reasoning We extended our method evaluation to include a Commonsense reasoning dataset to assess its impact on non-numerical datasets. For this evaluation, we tested all the trained models on CommonSenseQA (Talmor et al., 2019) dataset. As shown in Table 3, the results demonstrate a substantial improvement on all of the models. This further substantiates our claim that STC maintains generalisation and enhances the overall capabilities of the model.

6.3 Generation Quality

We selected two of the most prominent models, namely FLAN-UL2 and Vicuna 13B, from the Encoder-Decoder and Decoder-Only architectures for qualitative evaluation. Through qualitative analysis, we provide insights into the logicality and reasoning quality of the generated responses. The model selection was based on their overall performance in both the baseline and self-tuned versions. We evaluated the generated responses of these models using Human and machine evaluation, explained further below.

Human Evaluation We hired two annotators to assess the quality of the responses. The responses were presented to them in a randomized order, and they were instructed to select the response that appeared to be more logical. Additional details can be found in Appendix E. The results of the human evaluation are presented in Figure 4, indicating that the `STC` got a higher preference from both of the annotators for being a more logical response.

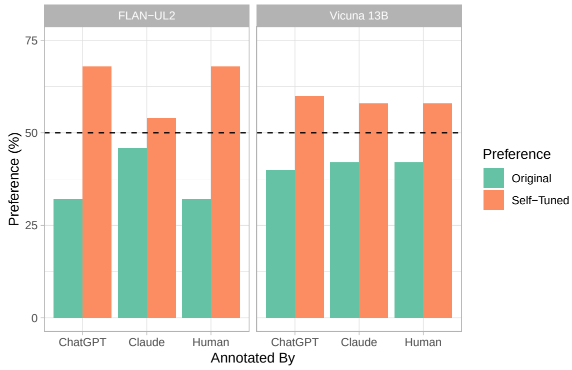

Figure 4: Human and machine evaluation results for FLAN-UL2 and Vicuna 13B. The dotted line is the random choice percentage (50%).

Machine Evaluation In addition to human evaluation, we employed Large Language Models (LLMs), specifically ChatGPT and Claude, to evaluate the quality and logicality of the generated responses. The models were prompted to select the more logical response, as further explained in Appendix F. These results complement the human evaluation, and as shown in Figure 4, indicate that LLMs also favor the responses generated by the Self-Tuned model over the baseline.

7 Ablation

In this section, we empirically validate the necessity of joint reward function through an ablation study. We used Vicuna-7B to train with either QA similarity only or self-logicality checking only.

QA Similarity Only By using QA similarity only as a reward function, the response length decreased as the reward mean increased, as shown in Figure 5. This is an expected flow as ROUGE-2 is a recall-based similarity metric that gives a higher score as two texts contain more same words.

Figure 5: Plot of response length and reward mean with using ROUGE-2 only.

Self-Logicality Only We compare the objective KL divergence of using self-logicality reward only and joint reward to assess the training stability in Figure 6. Ideally, the stable PPO training should result in the objective KL converging to the pre-defined target KL divergence with stable updates, which is 6 in our case. As shown in Figure 6, STC reached the target value with stable updates, while the self-logicality-only case failed to reach.

Figure 6: Plot of objective KL divergence from self-logicality reward only and STC.

As language models depend heavily on the input and the prompt, they still have inconsistencies even with the self-preference method. Although our method is more consistent than the previous works, as shown in Section 3, the language models are still a sub-optimal oracle. In that sense, we used the QA similarity, which is a rule-based static metric, along with self-logicality checking for stability.

8 Discussion

STC We demonstrate that our method enhances the reasoning abilities of fine-tuned Language models with both quantitative and qualitative analysis. We show that STC improved overall accuracy across various datasets while preserving the models' general abilities. Furthermore, human and machine evaluation confirms that our method enhances the quality and logicality of the generated responses. Additionally, we provide several examples in Appendix G for a comparative analysis of logicality between the trained and baseline models.

However, it is worth noting that specific models underperformed on particular datasets. Specifically, we observed decreased performance in the FLAN-T5-XL and XXL on the GSM8K and the MultiArith for decoder base models. The reasons for the decline in MultiArith for decoder-only models are explained in the following paragraph.

MultiArith Despite the fact that the decoder-only models were not directly trained for the arithmetic reasoning task, they showed notable improvements in GSM8K, AQUA, and SVAMP. However, they showed lower accuracy in MultiArith while all the encoder-decoder models increased. We believe there are mainly two reasons for this.

First, MultiArith is not an optimal dataset for CoT reasoning (Cobbe et al., 2021; Fu et al., 2023c). This is shown in both annotations of MultiArith (Roy and Roth, 2015; Cobbe et al., 2021) and the previous work, which studied the effects of complex prompting for CoT reasoning (Fu et al., 2023c). As Cobbe et al. (2021) stated early math reasoning datasets, including MultiArith, were not made for testing the capabilities of large language models; it does not have human annotation on step-wise reasoning. Also, Fu et al. (2023c) reported that accuracy has increased for large language models (>175B) by intensifying the number of steps in exemplars for few-shot reasoning in GSM8K and MathQA, while it decreased in MultiArith. This aligns with our results in Table 3, which shows that strengthening the explanatory reasoning of the language model can cause degradation in easy tasks.

Moreover, FLAN series models were specifically trained for multi-step math reasoning tasks, including GSM8K. The significant contribution of multi-task instruction-tuning in Chung et al. (2022) was that the fine-tuned models could perform well on unseen tasks in a zero-shot setting. Since GSM8K and MultiArith are for the same task with the only difference that comes from the difficulty (Fu et al., 2023c), FLAN series models have performed well for both before and after applying our method despite the problem mentioned in the first point.

9 Conclusion

We propose a novel self-tuning framework STC, where the model is trained based on its own evaluation. The novelty of this paper lies in creating a reasoning model that learns through self-evaluating without relying on external knowledge. By facilitating the model's self-checking ability for on its own chain-of-thought responses, we encourage the model to discover more elaborate rationales through its own efforts. Our results show that the self-guided method can strengthen the reasoning ability in both the quantity and quality aspects. Also, in comparison with Fu et al. (2023b), our method preserves the general knowledge about the world in original pre-trained models.

Limitations

Our framework shows the potentiality of self-guided learning in arithmetic reasoning tasks. While we currently utilize the self-logicality check reward to evaluate the overall response of the active model, recent papers have introduced more fine-grained evaluation metrics (Golovneva et al., 2022) and problem decomposition methods for COT (Zhou et al., 2023). These approaches analyze the logicality and consider fluency, informativeness, and other aspects of the explanation. By incorporating these fine-grained methods, we can potentially obtain rewards that better reflect the quality of the rationales, which can enable the model to find more optimal reasoning paths.

Furthermore, in this study, we focus on training and testing our model using the arithmetic reasoning dataset. However, reasoning ability is not limited to arithmetic but includes logical reasoning (Liu et al., 2020; Saparov and He, 2023), commonsense reasoning (Huang et al., 2019), etc. The advantage of the self-guided strategy is that it does not require labels or human annotations, and language models can be trained to obtain desired abilities without weakening the general knowledge existing in pre-trained model. Hence, we encourage applying self-guided learning methods to other reasoning tasks or more broad fields in NLP.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional ai: Harmlessness from ai feedback.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. Instructeval: Towards holistic evaluation of instruction-tuned large language models.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023a. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023b. Specializing smaller language models towards multi-step reasoning.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023c. Complexity-based prompting for multi-step reasoning.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2022. Roscoe: A suite of metrics for scoring step-by-step reasoning.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models.

Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. 2022. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation : Learning to solve and explain algebraic word problems.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

H. B. Mann and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60.

John McCarthy. 1959. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London. Her Majesty's Stationary Office.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. 2023. Language model self-improvement by reinforcement learning contemplation.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems?

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. Zero-offload: Democratizing billion-scale model training.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought.

Lysandre Debut Younes Belkada Sayak Paul Sourab Mangrulkar, Sylvain Gugger. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2022. Learning to summarize from human feedback.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. Ul2: Unifying language learning paradigms.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, and Nathan Lambert. 2020. Trl: Transformer reinforcement learning. `https://github.com/lvwerra/trl`.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

Seonghyeon Ye, Yongrae Jo, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, and Minjoon Seo. 2023. Selfee: Iterative self-revising llm empowered by self-feedback generation. Blog post.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks?

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences.

Appendix

This appendix contains the following contents: (1) The ablation study for text similarity reward in Section 4.3 (2) The query templates and toy examples for three different self-checking mechanisms introduced in Section 3 (B); (3) The templates for self-logicality checking in Section 4.3 (C) (4) Computational resources and hyperparameter settings in Appendix (D) (5) The question templates and instructions used for human evaluation in Section 6.3 (E); (6) The prompt template used for machine evaluation with ChatGPT and Claude in Section 6.3 (F); (7) Actual examples generated from trained FLAN-UL2 and Vicuna 13B (G)

A QA Similarity Ablation Study

In this section, we report the results of the ablation study on selecting the QA similarity metric. We demonstrate this by first comparing the sensitivity and adequacy of each metric on our task, and measuring the alignment of each metric against the actual thoroughness of the generated responses. For Table 5 and Table 7, we used "*Janet's ducks lay 16 eggs per day and, Janet eats 3 eggs for breakfast every morning, so she has 16 - 3 = 13 eggs available for sale each day.*" as a toy example.

Alignment Against Thoroughness of Reasoning Paths We conducted an experiment calculating the similarity between question and answer across Flan-T5 models. We can see the sentence similarity scores exhibited a consistent rise, except for BLEURT in Table 5.

Model	ROUGE-2	ROUGE-L	BERTScore	SimCSE	BLEURT
FLAN-T5-Base (223M)	0.1595	0.2808	0.8568	0.7023	0.5586
FLAN-T5-Large (1B)	0.1977	0.3286	0.8718	0.7363	0.5560
FLAN-T5-XL (3B)	0.1972	0.33	0.8726	0.7207	0.5492
FLAN-T5-XXL (11B)	0.2177	0.3544	0.8783	0.7525	0.5544

Table 5: Text similarity between question and answer among Flan-T5 models.

Sensitivity Comparison We compare the sensitivity of rule-based metric and embedding vector-based metric. As shown in Table 5, BERTScore changes for 0.002 when we change number '*16*' to other close numbers. However, ROUGE-2 changes for 0.05 when we change '*16*' to other close numbers. Since the numbers are also considered a token in embedding vector-based similarity metrics like BERTScore, those scores depend on the margin of error while calculating the similarity score. This can be a helpful aspect in some tasks, but we use the QA similarity metric as a signal of how precisely the models utilize the information and clues given in the question. Therefore, embedding vector-based similarity scores (e.g. BERTScore, BLEURT, SimCSE) would not be appropriate for our task.

Answer	BERTScore	ROUGE-2
Janet's ducks lay 16 eggs per day. Janet eats 3 eggs for breakfast every morning, ...	0.907	0.253
Janet's ducks lay 17 eggs per day. Janet eats 3 eggs for breakfast every morning, ...	0.905	0.202
Janet's ducks lay 15 eggs per day. Janet eats 3 eggs for breakfast every morning, ...	0.905	0.202
Janet's ducks lay 11 eggs per day. Janet eats 3 eggs for breakfast every morning, ...	0.905	0.202
Janet's ducks lay 19 eggs per day. Janet eats 3 eggs for breakfast every morning, ...	0.905	0.202

Table 6: The sensitivity comparison between BERTScore and ROUGE-2 by altering the numbers only in the statement. As close numbers are also close in the embedding space, BERTScore gives a similar score to errors like miscalculation. However, ROUGE-2 penalizes any numerical errors as it measures the exact match only.

Furthermore, we compare two rule-based similarity metrics, BLEU and ROUGE. While BLEU and ROUGE calculate the exact match of words between two given sentences, ROUGE is a recall-based method, and BLEU is a precision-based method. This is also well shown in the results in Table 7. Regarding our training objective, the model will use the words that do not appear in the question while generating the chain-of-thought responses. In this case, precision-based metrics like BLEU will give stronger penalties, and recall-based metrics like ROUGE will give relatively minor penalties to the new tokens. Therefore, ROUGE-2, which is a recall-based similarity metric, would be more appropriate for our objective.

Answer	ROUGE-2	BLEU-2
Janet's ducks lay 16 eggs per day and, ... for sale each day. [Gen1].	0.247	0.277
Janet's ducks lay 16 eggs per day and, ... for sale each day. [Gen1] [Gen2].	0.244	0.256
Janet's ducks lay 16 eggs per day and, ... for sale each day. [Gen1] [Gen2] [Gen3].	0.241	0.238
Janet's ducks lay 16 eggs per day and, ... for sale each day. [Gen1] [Gen2] [Gen3] [Gen4].	0.238	0.222
Janet's ducks lay 16 eggs per day and, ... for sale each day. [Gen1] [Gen2] [Gen3] [Gen4] [Gen5].	0.235	0.208

Table 7: Sensitivity comparison between ROUGE-2 and BLEU-2. BLEU-2 is relatively more sensitive to the new tokens as a precision-based metric. On the other hand, ROUGE-2 is relatively robust to the new tokens as a recall-based metric.

B Can LLMs Evaluate Themselves?

This section reports the detailed templates used for each logicality checking mechanism in Section 3. Every query template shown in the following paragraphs is used after generating a zero-shot chain-of-thought reasoning response from the model.

B.1 Self-Correction

Using the query template shown in Pang et al. (2023), we queried the model with yes/no questions according to the following format:

Is the answer to the question correct? The question is: `{question}`. *The answer is:* `{response}`

where `{question}` and `{response}` refer to the given question and generated response from the model, respectively. We greedy decoded responses from FLAN-T5-Large, FLAN-T5-XL, and FLAN-T5-XXL. All the responses from three models were either *yes* or *no* as shown in Table **??**.

B.2 Self-Rating

Using the query template shown in Pang et al. (2023), we queried the model with open question according to the following format:

Please evaluate the answer to the question and give me an evaluation score from 1 to `{max_score}`. *The question is:* `{question}`. *The answer is:* `{response}`

where `{max_score}`, `{question}` and `{response}` refer to the maximum scale of score, given question and generated response from the model, respectively. We greedy decoded responses from FLAN-T5-Large, FLAN-T5-XL, and FLAN-T5-XXL. All the responses from three models were floats (e.g. 1.0, 8.0) as shown in Figure **??**.

B.3 Self-Preference

We queried multi-choice question with four choices as the following example:

Select the item which derived the answers with the most logical ways.
(1) There are 16 balls / 2 = 8 golf balls. There are 8 golf balls * 2 / 2 = 4 blue golf balls. Therefore, the answer is 4.
(2) If half of the balls are golf balls, the juggler can juggle 16. 1 / 2 * 16 = 8 golf balls. Since each juggler has 8 golf balls, then the 8 golf balls are 8 / 2 = 4 balls. The 8 golf balls are all blue golf balls. Therefore, the answer is 4.
(3) 16 / 2 = 8 golf balls. 8 / 2 = 4 balls are blue. Therefore, the final answer is 4. Therefore, the final answer is 4.
(4) Let x be the number of blue balls. Half of the balls are golf balls. If 1 / 2 of the balls are golf balls, this will be x * 2 = 4. This means that there are 4 blue balls. Therefore the final answer is 4.

We sampled the responses from this example for hundred times to measure if the model shows a consistent preference for a certain chain-of-thought reasoning style. If the model's response is not one of 1 to 4, we consider it as a hallucination.

Figure 2 shows that the model prefers the fourth option, *Let x be the number of blue balls. Half of the balls are golf balls. If 1 / 2 of the balls are golf balls, this will be x * 2 = 4. This means that there are 4 blue balls. Therefore the final answer is 4.* in both original and reversed order.

C Logicality Checking

We query the model in the format of a dichotomous question in Table 8 to select the more logical answer with two options, the greedy decoded response $A_{\pi^{Ref}}$ from the original model and the top-p sampled response $A_{\pi^{Act}}$ from the active model. If the model selects $A_{\pi^{Act}}$, we give a reward of 1 and 0 otherwise. This leads the model to search the new reasoning paths which are likely to be more logical than $A_{\pi^{Ref}}$.

Instruction	Given the question "`{question}`", which of the following responses is more logical:
Options	(1) `{Greedy Decoded Response from the Reference Model}` (2) `{Sampled Response from the Active Model}`

Table 8: The dichotomous question template for self-logicality checking.

D Implementation Details

All models were trained on either A6000 or A100 NVIDIA GPUs with Parameter Efficient Fine-Tuning and model parallelism for three epochs using AdamW (Loshchilov and Hutter, 2019) optimiser. Each model was trained with its respective original floating point precision. For example, Flan-UL2 was trained using bf16, while Flan-T5-XXL utilized full precision fp32. Hyperparameters for PPO and LoRA are shown in Table 9. Each model took around one day per epoch for training.

`init_kl_coef` was 0.1 for the encoder-decoder models and 0.05 for the decoder-only models. We distinguished the KL coefficient as the encoder-decoder models have seen the math reasoning tasks at the pre-training stage, while the decoder-only models did not. Also, the mini-batch size was 4 on the models with more than 10B parameters and 8 for the rest. We use TRL (von Werra et al., 2020), the huggingface implementation of Proximal Policy Optimization (PPO) to optimise the model using the dual reward function explained in Section 4.3.

Parameter Efficient Fine-Tuning Training large language models demands significant computational resources, making it impractical for many use cases. To address this challenge and enable training models of sizes ranging from 1 billion to 20 billion parameters within computationally constrained environments, we employed a Parameter Efficient Fine-Tuning method called LoRA (Hu et al., 2021; Sourab Mangrulkar, 2022). In our approach, we adopted the rank 64 for LoRA for all models, ensuring a balance between having sufficient trainable parameters and avoiding excessive memory consumption.

Model Parallelism While PEFT (Appendix D) effectively addresses the challenge of training LLMs up to a specific limit, the substantial number of parameters of huge Language models make them unable to fit within the memory capacity of a single GPU. Consequently, we utilize model parallelism techniques to overcome the memory limitations inherent in training Large Language Models (LLMs). Specifically, we employ the Hugging Face implementation of DeepSpeed Stage 2 (Rajbhandari et al., 2020; Ren et al., 2021; Rajbhandari et al., 2021) to distribute the training process across multiple GPUs.

Hyperparameter	Setting
ppo_epoch	4
init_kl_coef	0.1 (Encoder-Decoder), 0.05 (Decoder-Only)
horizon	1,000
batch_size	128
mini_batch_size	4, 8
gradient_accumulation_steps	1
output_min_length	200
output_max_length	400
optimizer	AdamW
learning_rate	5e-05
gamma	0.99
rank	64

Table 9: Hyperparameter settings for PPO and LoRA.

E Human Evaluation

We hired two non-expertise annotators for human evaluation. Each annotator was tested on FLAN-UL2 and Vicuna 13B. Each question asked the annotators to select the more logical answer between the response from the original language model and the self-tuned language model. The responses from both models were greedy decoded responses.

The annotators were first informed with the given instruction in Table 10. Then they annotated for fifty questions with the template shown in Table 11. The results for human preference can be found in Figure 4.

Instruction	This would be the instruction and following questions for 50 questions: Read the given question and the following responses to answer the questions.
Criteria	Consider (1) Preciseness (2) The validity of reasoning steps (3) The degree of hallucination (4) Elaborativeness to answer the questions.

Table 10: The first instruction page given to annotators. the instruction provides general criteria for evaluating the logicality of responses.

Instruction	Read the given question and the following responses to answer the questions. Consider preciseness, the validity of reasoning steps, and elaborativeness to answer the questions.

Options	Question: Robin's hair was 16 inches long...
	
	(1)
	`Response 1`
	(2)
	`Response 2`
	
	Which of the responses is more logical? Enter 1 or 2.

Table 11: The actual human evaluation template shown for the annotators. The responses were saved automatically into a JSON file.

F Machine Evaluation

Table 12 show the prompt that was used to evaluate the quality of generated responses using ChatGPT and Claude. While making the template as similar as possible to human evaluation, we removed the notation "Question:" in Table 11 to clarify the prompt.

Instruction	Which of the given responses provides more logical reasoning to answer the question? Select either (1), (2), or both.

Options	(1)
	`Response 1`
	(2)
	`Response 2`

Table 12: The actual machine evaluation template prompted to ChatGPT and Claude. The responses were saved automatically into JSON file.

G Generation Examples

We demonstrate and compare the example responses between the original instruction-tuned models and the trained models. The responses were greedily decoded for both models. We categorize the examples into three different types: 1) Examples with same responses (G.1); 2) Examples with minor differences (G.2); 3) Examples with significant differences (G.3). The examples may contain the responses with miscalculation or hallucinations for both baseline and trained models. We report the responses from FLAN-UL2 and Vicuna 13B each for three categories.

G.1 Examples with Exact Same Responses

Query	Question: While on vacation, Gwen took 41 pictures at the zoo and 29 at the museum. If she later deleted 15 of the pictures, how many pictures from her vacation did she still have? Answer the question by reasoning step-by-step.
STEPER	Gwen took 41 + 29 = 70 pictures in total. After deleting 15 pictures, she had 70 - 15 = 55 pictures left. Therefore, the answer is 55.
Baseline	Gwen took 41 + 29 = 70 pictures in total. After deleting 15 pictures, she had 70 - 15 = 55 pictures left. Therefore, the answer is 55.

Table 13: Examples with exact same responses generated from FLAN-UL2

Query	Question: We ordered 9 pizzas. Each pizza has 10 slices. If there are 2 of us How many slices of pizza does each of us get if distributed equally? Answer the question by reasoning step-by-step. Answer:
STEPER	9 pizzas 10 slices/pizza = «9*10=90»90 slices 2 people 90 slices = «2*90=180»180 slices So, each person gets 180 slices/2 people = «180/2=90»90 slices. Therefore, each person gets 90 slices of pizza if the pizzas are distributed equally.
Baseline	9 pizzas 10 slices/pizza = «9*10=90»90 slices 2 people 90 slices = «2*90=180»180 slices So, each person gets 180 slices/2 people = «180/2=90»90 slices. Therefore, each person gets 90 slices of pizza if the pizzas are distributed equally.

Table 14: Examples with exact same responses generated from Vicuna 13B

G.2 Examples with Minor Differences

Query	Question: Robin's hair was 16 inches long. He cut off 11 inches. Then it again grew by 12 inches. How long is his hair now? Answer the question by reasoning step-by-step.
STEPER	Robin's hair was 16 inches long and he cut off 11 inches so 16 - 11 = 5 inches long. His hair was 5 inches long and grew 12 inches so **it is now** 5 + 12 = 17 inches long. Therefore, the answer is 17.
Baseline	Robin's hair was 16 inches long and he cut off 11 inches so 16 - 11 = 5 inches long. His hair was 5 inches long and grew 12 inches so 5 + 12 = 17 inches long. Therefore, the answer is 17.

Table 15: Examples with minor differences generated from FLAN-UL2. The different phrase is **underlined**.

Query	Question: Carol and her mom were picking carrots from their garden. Carol picked 29 and her mother picked 16. If only 38 of the carrots were good, how many bad carrots did they have? Answer the question by reasoning step-by-step. Answer:
STEPER	First, we need to find out how many carrots they picked altogether. Carol picked 29 and her mother picked 16, so they picked a total of 29 + 16 = 45 carrots. Next, we need to find out how many good carrots they picked. Since they only picked 38 good carrots **and they picked a total of 45 carrots,** **it means that they left 45 - 38 = 7 bad carrots in the garden.** Therefore, Carol and her mother picked a total of 45 carrots, and 7 of them were bad.
Baseline	First, we need to find out how many carrots they picked altogether. Carol picked 29 and her mother picked 16, so they picked a total of 29 + 16 = 45 carrots. Next, we need to find out how many good carrots they picked. Since they only picked 38 good carrots, this means that they left 45 - 38 = 7 bad carrots in the garden. Therefore, Carol and her mother picked a total of 45 carrots, and 7 of them were bad.

Table 16: Examples with minor differences generated from Vicuna 13B. The different phrase is **underlined**.

G.3 Examples with Major Differences

Query	Question: In a video game, each enemy defeated gives you 9 points. If a level has 11 enemies total and you destroy all but 3 of them, how many points would you earn? Answer the question by reasoning step-by-step.
STEPER	If there are 11 enemies and you destroy all but 3 of them, then you would have destroyed 11 - 3 = 8 enemies. If each enemy is worth 9 points and you destroy 8 enemies, then you would earn 8 * 9 = 72 points. Therefore, the answer is 72.
Baseline	You would earn 9 * 11 - 3 = 93 points. Therefore, the answer is 93.

Table 17: Examples with major differences generated from FLAN-UL2.

Query	Question: Brenda's mother made 14 cookies for 2 guests. If each of them had the same number of cookies How many did each of them have? Answer the question by reasoning step-by-step. Answer:
STEPER	Answer: 7 Explanation: Let's assume that each guest gets x cookies. Since each guest gets the same number of cookies, we can write: x = x Since there are 14 cookies in total, we can write: 2x = 14 Solving this equation, we find that x = 7. Therefore, each guest gets 7 cookies.
Baseline	Answer: 7 Explanation: Let's assume that each guest gets x cookies. Since there are 2 guests and Brenda's mother made 14 cookies in total, each guest will get 14/2 = 7 cookies. Therefore, each guest gets 7 cookies.

Table 18: Examples with major differences generated from Vicuna 13B.