Self-Training Meets Consistency: Improving LLMs' Reasoning with Consistency-Driven Rationale Evaluation

Jaehyeok Lee[1], **Keisuke Sakaguchi**[2*], **JinYeong Bak**[1*]
[1]Sungkyunkwan University, Suwon, South Korea
[2]Tohoku University, Sendai, Japan
hjl8708@skku.edu, keisuke.sakaguchi@tohoku.ac.jp, jy.bak@skku.edu

Abstract

Self-training approach for large language models (LLMs) improves reasoning abilities by training the models on their self-generated rationales. Previous approaches have labeled rationales that produce correct answers for a given question as appropriate for training. However, a single measure risks misjudging rationale quality, leading the models to learn flawed reasoning patterns. To address this issue, we propose CREST (Consistency-driven Rationale Evaluation for Self-Training), a self-training framework that further evaluates each rationale through follow-up questions and leverages this evaluation to guide its training. Specifically, we introduce two methods: (1) filtering out rationales that frequently result in incorrect answers on follow-up questions and (2) preference learning based on mixed preferences from rationale evaluation results of both original and follow-up questions. Experiments on three question-answering datasets using open LLMs show that CREST not only improves the logical robustness and correctness of rationales but also improves reasoning abilities compared to previous self-training approaches.[1]

1 Introduction

Large language models (LLMs) can enhance multi-step reasoning abilities by generating intermediate reasoning steps (i.e., rationale) before arriving at an answer (Wei et al., 2022). Training LLMs on high-quality rationales has been shown to improve their reasoning capabilities (Chung et al., 2024; Liu et al., 2023; Shridhar et al., 2023). Therefore, collecting high-quality rationales is becoming increasingly important for training the reasoning abilities of LLMs. However, due to the high cost associated with collecting high-quality rationales, self-training approaches have emerged, focusing on training LLMs using self-generated rationales (Zelikman et al., 2022).

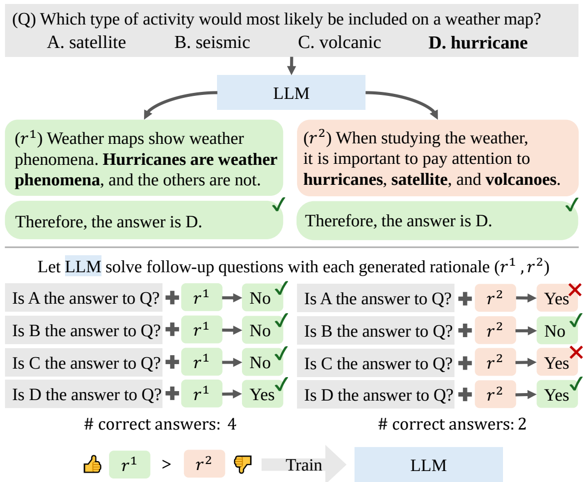

Figure 1: An example of rationale generation and evaluation in CREST: An LLM generates two rationales (r^1, r^2) and answer predictions to solve question Q. Even though r^2 lacks focus and clear support for the answer, previous approaches evaluate both r^1 and r^2 as equally right. Through a more fine-grained evaluation using follow-up questions, we can identify the better rationale, r^1, which leads to more consistent predictions across all questions.

In self-training approaches, accurately evaluating the quality of generated rationales is essential. Previous studies have evaluated rationale quality by examining whether the generated rationales lead to the correct answer to a given question (Zelikman et al., 2022; Hoffman et al., 2023; Feng et al., 2024; Hosseini et al., 2024; Singh et al., 2024). However, using the correctness of a single prediction is unstable, as LLMs can reach correct answers through inappropriate reasoning steps (Bao et al., 2024a). Figure 1 shows an example of two generated rationales, r^1 and r^2. Despite r^2 shows incomplete reasoning, previous approaches would consider both rationales equally appropriate since they both lead to the correct answer for Q. Training

[*]Corresponding authors
[1]Code: https://github.com/JaehyeokLee-119/CREST

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 10519–10539
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

models on such inappropriate rationales can cause them to learn flawed reasoning patterns.

To address this problem, we propose CREST (Consistency-driven Rationale Evaluation for Self-Training), a novel framework for LLM self-training. The core idea of CREST is to further evaluate rationales using follow-up questions that ask whether each answer option in the original question is correct or not. We first generate diverse rationales using temperature sampling and evaluate them with an LLM as shown in Figure 1. Subsequently, we train the LLM on these rationales, rewarding rationales that lead to more consistent predictions (i.e., r^1) and penalizing those that lead to less consistent predictions (i.e., r^2). To achieve this, we propose two methods: rationale filtering and preference learning. In rationale filtering, we remove rationales that lead to incorrect answers in more than a certain number of follow-up questions during the supervised fine-tuning process. In preference learning, we train the model on mixed preferences from results of both original and follow-up questions, to favor rationales that result in correct answers in a greater number of follow-up questions.

We conduct experiments on three natural language reasoning question-answering datasets, including ReClor (Yu et al., 2020), ARC (Clark et al., 2018), and CSQA (Talmor et al., 2019). We compare CREST to other self-training approaches using Llama 3 model (AI@Meta, 2024) and Gemma model (Team et al., 2024). Our findings show that CREST can train an LLM to generate more correct and robust rationales, improving its reasoning performance. Our contributions are as follows:

- We introduce consistency-driven rationale evaluation, which further evaluates generated rationales using follow-up questions that ask whether each answer option in the original question is correct or not.

- We propose CREST, which evaluates generated rationales via consistency-driven rationale evaluation and uses the evaluation results to train an LLM through two methods: rationale filtering and preference learning using mixed preferences derived from original and follow-up question evaluations.

- We conduct experiments and analyses with open LLMs such as Llama 3 model and Gemma model on three question-answering datasets. The results show that CREST generates more robust and correct rationales and improves reasoning ability compared to other self-training approaches.

2 Related Work

2.1 Self-Training Approaches

Chain-of-Thought (CoT) approach demonstrates that generating a step-by-step reasoning path before the final prediction enhances an LLM's reasoning abilities (Wei et al., 2022). Training LLMs on rationale data generated by humans (Chung et al., 2024) or advanced models like GPT-4 further enhances reasoning abilities (Liu et al., 2023). However, since high-quality rationale data is expensive to obtain, a number of approaches focus on training language models using self-generated rationales. STaR (Zelikman et al., 2022), an early type of self-training approach, trains the language model by selecting the correct rationales based on binary feedback regarding the correctness of the answers generated by these rationales. RFT (Yuan et al., 2023) enhances supervised data by generating and collecting diverse correct reasoning paths, focusing on mathematical reasoning. Other approaches, such as V-STaR, Iterative RPO, and Self-motivated Learning, also utilize incorrect rationales (Feng et al., 2024; Hosseini et al., 2024; Pang et al., 2024) and adopt preference learning techniques, such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Direct Preference Optimization (DPO) (Rafailov et al., 2023). Self-Explore (Hwang et al., 2024) provides fine-grained rewards by identifying incorrect steps within the rationales. Wei Jie et al. (2024) proposes a self-training framework that exposes a model to each question multiple times using temperature sampling, thereby assessing the model's confidence in the given question. CREST provides fine-grained rewards through evaluating a self-training rationale multiple times using follow-up questions augmented from the original dataset, emphasizing the rationale's ability to consistently lead to correct answers.

2.2 Reasoning with Consistency

Consistency is the ability to make consistent decisions in semantically equivalent contexts (Elazar et al., 2021). It is a desirable property of logically valid machine learning systems (Chen et al., 2024a) and an important characteristic for a model

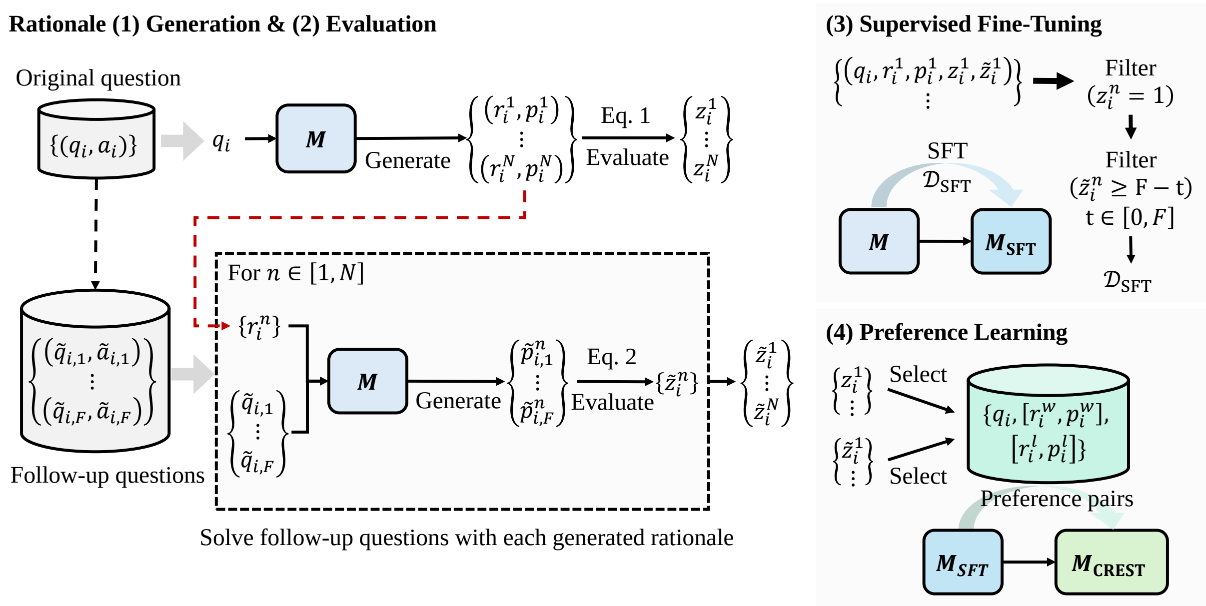

Figure 2: Overview of CREST. In Rationale Generation (1), given a question q_i and an answer a_i, an initial LLM $\boldsymbol{M}$ generates N rationales and answer predictions (r_i, p_i) to solve q_i, and then solves follow-up questions $\tilde{q}_{i,f}$ using each rationale r_i^n, resulting in $\tilde{p}_{i,f}^n$. Next, in Rationale Evaluation (2), we assign rewards z and $\tilde{z}$ to each rationale based on the correctness of the predictions as shown in Eq. 1 and Eq. 2. In Supervised Fine-Tuning (3), we train $\boldsymbol{M}$ on the rationales filtered by z and $\tilde{z}$ with a tolerance term t, resulting in $\boldsymbol{M}_{\textbf{SFT}}$. Finally, in Preference Learning (4), we build preference pairs based on z and $\tilde{z}$, and train $\boldsymbol{M}_{\textbf{SFT}}$ on them, resulting in $\boldsymbol{M}_{\textbf{CREST}}$.

to be considered trustworthy (Jang et al., 2022). As larger language models emerge that exceed human performance in many tasks, consistency is receiving increased attention due to its role in evaluating inference validity, even in models that outperform humans (Fluri et al., 2024). To evaluate a model's consistency, follow-up questions generated from existing questions are commonly used (Ribeiro et al., 2019; Elazar et al., 2021; Jang et al., 2022; Chen et al., 2024a; Zheng et al., 2024; Chen et al., 2024b). Several techniques have been developed to create these follow-up questions, including generating semantically identical texts by paraphrasing the original input texts (Elazar et al., 2021), crafting logically equivalent questions (Jang et al., 2022), and developing questions that investigate the implications of the model's answers (Ribeiro et al., 2019). Two main approaches have been proposed to enhance both the consistency and task performance of models: designing models specifically to reduce inconsistency (Kassner et al., 2021, 2023), and synthesizing consistent data to train models (Alberti et al., 2019; Asai and Hajishirzi, 2020; Elazar et al., 2021). CREST evaluates rationales that correspond to the reasoning process with augmented questions and trains an LLM to prefer those that consistently lead to correct answers.

3 Consistency-driven Rationale Evaluation for Self-Training

This section describes our approach, Consistency-driven Rationale Evaluation for Self-Training (CREST) which trains reasoning abilities through consistency-driven rationale evaluation with follow-up questions.

3.1 Notation

We have a pretrained large language model $\boldsymbol{M}$ and an original dataset of questions q with answers a, represented as $\mathcal{D} = \{(q_i, a_i)\}_{i=1}^{D}$. Each question has F answer choices. To solve q, $\boldsymbol{M}$ sequentially generates a rationale r, corresponding to intermediate reasoning steps, and an answer prediction p, where r leads to p.

3.2 CREST

The whole framework of CREST consists of four stages. Figure 2 outlines the overview of CREST.

- **Rationale Generation** We generate N diverse rationales r_i^n for each question q_i and the corresponding answer predictions p_i^n using $\boldsymbol{M}$, where $n \in [1, N]$.

- **Rationale Evaluation** We compare p_i^n with a_i to assign a reward z_i^n to r_i^n based on the correctness of the prediction. Subsequently, we generate multiple follow-up questions $\tilde{q}_{i,f}$ from q_i and further evaluate r_i^n using these follow-up questions. We assign an additional reward $\tilde{z}_i^n$ to r_i^n based on how many $\tilde{q}_{i,f}$ are answered correctly.

- **Supervised Fine-Tuning** We train $\boldsymbol{M}$ through supervised fine-tuning to create $\boldsymbol{M}_{\textbf{SFT}}$ using the generated rationales filtered based on the evaluation results.

- **Preference Learning** We train $\boldsymbol{M}_{\textbf{SFT}}$ using a preference learning algorithm according to the preferences indicated by the evaluation results, resulting in $\boldsymbol{M}_{\textbf{CREST}}$.

3.3 Rationale Generation

Initially, we generate diverse rationales and the corresponding answer predictions for a given original question q_i with $\boldsymbol{M}$. Specifically, $\boldsymbol{M}$ generates N rationales r_i^n as follows: $r_i^n \leftarrow \boldsymbol{M}(q_i)$, where r_i^n represents the nth rationale generated for the ith question. Subsequently, $\boldsymbol{M}$ derives answer predictions p_i^n for q_i from generated rationales r_i^n, as follows: $p_i^n \leftarrow \boldsymbol{M}(q_i, r_i^n)$.

3.4 Rationale Evaluation

We evaluate the rationale through a two-step process. Firstly, similar to previous studies (Zelikman et al., 2022; Yuan et al., 2023; Hosseini et al., 2024; Feng et al., 2024; Pang et al., 2024), we compare the ground truth answer a_i for q_i with the predicted answer p_i^n derived from r_i^n. Secondly, we further assess the rationales through F follow-up questions which are generated from the original question q_i.

In the first step, we assign a binary reward z_i^n of either 0 or 1 to each rationale based on whether p_i^n matches a_i as follows:

$$z_i^n = \mathbf{1}(p_i^n = a_i) \tag{1}$$

Assuming that rationales leading to the correct answer are of higher quality than those that do not, as suggested by Zelikman et al. (2022), this evaluation directly measures the quality of rationales.

In the second step, we evaluate the rationales using F follow-up questions $\{(\tilde{q}_{i,1}, \tilde{a}_{i,1}), ..., (\tilde{q}_{i,F}, \tilde{a}_{i,F})\}$ generated from q_i, where $\tilde{a}_{i,f}$ is the ground truth answer for the fth follow-up question corresponding to q_i. We then evaluate the rationales on all F follow-up questions: $\tilde{p}_{i,f}^n \leftarrow \boldsymbol{M}(\tilde{q}_{i,f}, r_i^n)$, where $\tilde{q}_{i,f}$ is f^{th} follow-up question for q_i.

We assign an additional reward $\tilde{z}$ to each rationale based on the number of correctly solved follow-up questions as follows:

$$\tilde{z}_i^n = \sum_{f=1}^{F} \mathbf{1}(\tilde{p}_{i,f}^n = \tilde{a}_{i,f}) \tag{2}$$

To generate follow-up questions that are closely related to the problem-solving process of each question in $\mathcal{D}$, we utilize the characteristics of multiple-choice questions: the solving process involves not only identifying the correct answer but also eliminating the incorrect options. We design each follow-up question to ask whether each of the answer options in the original question is correct or not. This type of follow-up question is used to evaluate the robustness of reasoning ability in multiple-choice question-answering datasets (Wang et al., 2024). Figure 1 shows an example of the follow-up questions and the evaluation.

3.5 Supervised Fine-Tuning

After evaluating the rationales, we use z and $\tilde{z}$ as filters to select the rationales for training $\boldsymbol{M}$ and produce $\boldsymbol{M}_{\textbf{SFT}}$ through supervised fine-tuning (SFT). Intuitively, the best rationales for q_i from the previous stage are those that lead to the correct answers to q_i and all F follow-up questions, indicated by $z_i^n = 1$ and $\tilde{z}_i^n = F$. However, simply removing rationales that lead to incorrect answers for any of the follow-up questions might drastically reduce the number of rationales available for training. Therefore, we also include some sub-optimal rationales with a tolerance term t that satisfies $t \in [0, F]$. Consequently, the dataset $\mathcal{D}_{\text{SFT}}$ used to train $\boldsymbol{M}$ in the SFT stage is represented as follows:

$$\begin{aligned}\mathcal{D}_{\text{SFT}} = &\{q_i, r_i^n, a_i | \\ &(n,i) \in \{(n,i) | z_i^n = 1, \tilde{z}_i^n \geq F - t\}\}\end{aligned} \tag{3}$$

The training objective for this stage aligns with that used during pretraining, specifically employing an auto-regressive language modeling objective or next-token prediction (Radford et al., 2018). We calculate the loss exclusively for the output section (i.e., r and a).

3.6 Preference Learning

We further train $\boldsymbol{M}_{\textbf{SFT}}$ by exploiting preferences between rationales to enhance its reasoning ability. To achieve this, we construct preference pairs and fine-tune $\boldsymbol{M}_{\textbf{SFT}}$ using offline preference learning methods, such as Direct Preference Optimization (DPO) (Rafailov et al., 2023).

3.6.1 Preference Pair Dataset Construction

We construct the preference pair dataset P_{total} for preference learning by first creating two sets of preference pairs P_z and $P_{\tilde{z}}$, which represent rationale preferences based on the rewards z and $\tilde{z}$, respectively. P_{total} is then formed by randomly sampling pairs from these two sets.

To construct P_z and $P_{\tilde{z}}$, we generate preference pairs in which rationales with higher rewards r^w are preferred over those with lower rewards r^l, based on z and $\tilde{z}$, respectively. Each preference pair consists of a question q, two generated rationales, and their corresponding predictions p^w and p^l: (q, r^w, p^w, r^l, p^l). Algorithm 1 outlines the detailed procedure for generating P_z and $P_{\tilde{z}}$.

Then, we construct P_{total} by sampling pairs from P_z and $P_{\tilde{z}}$ with a weighting factor λ, which controls the relative contribution of rationale preferences derived from z and $\tilde{z}$ during preference learning. The parameter λ satisfies $\lambda \in [0, 1]$, ensuring that the proportion of $P_{\tilde{z}}$ in P_{total} is λ. For instance, if a total of 10,000 pairs are used for preference learning and $\lambda = 0.4$, P_{total} would consist of 4,000 randomly selected pairs from $P_{\tilde{z}}$ and 6,000 randomly selected pairs from P_z. The total number of pairs used for preference learning is determined by the maximum number of training steps multiplied by the batch size.

3.6.2 Training

We train $\boldsymbol{M}_{\text{SFT}}$ on the preference pairs P_{total} using DPO, resulting in $\boldsymbol{M}_{\text{CREST}}$. Given the preference pairs P_{total}, the objective of this stage is to increase the log-likelihood of preferred outputs over dispreferred ones:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(r_i^w, p_i^w, r_i^l, p_i^l, q_i) \sim P_{total}} \left[\log \sigma \left(\hat{r}_\theta(q_i, r_i^w, p_i^w) - \hat{r}_\theta(q_i, r_i^l, p_i^l) \right) \right] \quad (4)$$

$$\hat{r}_\theta(q, r, p) = \beta \log \frac{\pi_\theta(r, p|q)}{\pi_{ref}(r, p|q)} \quad (5)$$

where $\pi_\theta(r, p|q)$ and $\pi_{ref}(r, p|q)$ represent the probability of outputs r and p given input q under the current policy parameterized by θ and a reference policy π_{ref}, respectively. Initially, both π_θ and π_{ref} are initialized as $\boldsymbol{M}_{\text{SFT}}$, and they are updated each epoch. π_{ref} is used to minimize distribution shift from the true reference distribution and is typically initialized through supervised fine-tuning on preferred outputs. β controls the deviation from the reference policy.

4 Experiments

This section describes the experiments and results of CREST compared to other self-training approaches. First, we introduce the three datasets used for model training and testing. Next, we present the experimental setup, including the base LLM, key hyperparameters, and performance metrics. We also introduce the baseline approaches used for comparison, and finally, we present the results of the experiments.

4.1 Experimental Settings

Datasets We evaluate CREST on three English natural language reasoning multiple-choice QA datasets: ReClor (Yu et al., 2020), ARC (Clark et al., 2018), CSQA (Talmor et al., 2019). ReClor comprises logical reasoning problems derived from American graduate school entrance exams and their preparatory materials. The ReClor test set is divided into an Easy set, which consists of biased data points, and a Hard set, which includes the remaining data points. ARC is sourced from grade-school science assessments for students of various grades. The questions are categorized into two sets: an Easy set and a Challenge set. In our experiments, we only test on the Challenge set, as in previous studies (Huang et al., 2023; Pang et al., 2024). CSQA consists of short questions that require common sense reasoning, built upon ConceptNet (Speer et al., 2017).

Models We conduct our experiments using the Llama 3 8B model[2] (AI@Meta, 2024) and the Gemma 7B model[3] (Team et al., 2024) from *HuggingFace* (Wolf et al., 2020), training them with Low-Rank Adaptation (LoRA) (Hu et al., 2022).

Implementation Details We generate rationales with temperature sampling with the following parameters: T=0.8, $TopP$=0.95, N=16, and max_new_tokens=512, then use greedy decoding for answer prediction. For supervised fine-tuning, we use epoch=6, batch size=32 and conduct learning rate search between $\{5e-6, 5e-3\}$. For preference learning, we use β=0.1, epoch=4, batch size=8, and search max number of steps among $\{3000, 5000\}$ and conduct learning rate search between $\{5e-7, 5e-5\}$ for all models. The input and output prompt templates for model evaluation

[2] https://huggingface.co/meta-llama/Meta-Llama-3-8B

[3] https://huggingface.co/google/gemma-7b

	Base model	Llama 3 8B			Gemma 7B		
Approach	Model	ReClor	ARC-C	CSQA	ReClor	ARC-C	CSQA
	Zero-shot	52.10	69.28	53.89	53.60	77.47	65.68
	Few-shot	55.30	77.21	70.76	58.70	83.11	75.02
Self-training	STaR	58.60	77.99	76.17	58.40	82.34	77.56
	RFT	64.40	80.72	78.54	66.90	83.36	80.02
	Self-motivated Learning	67.80	80.03	80.34	68.20	83.53	80.59
	M_{SFT}	66.10	81.40	79.36	67.90	84.22	80.51
	M_{CREST}	**69.50**	**81.91**	**81.41**	**70.00**	**84.47**	**80.67**
Direct fine-tuning	Fine-tune (Label)	77.40	80.80	80.18	81.90	85.58	84.44
	Fine-tune (Label)$_{\text{CREST}}$	**79.30**	**81.23**	**81.24**	**83.70**	**87.20**	**84.85**

Table 1: Accuracy of various models across three reasoning datasets with Llama 3 8B and Gemma 7B model. ARC-C denotes the challenge set in the ARC test set. CREST consistently improves accuracy across all three datasets.

are illustrated in Figures 7 and 10. For more details about the prompts used in this study, please refer to Appendix F.

4.2 Baselines

- **Fine-tune (Label)** involves directly fine-tuning the base model on ground truth labels using a negative log-likelihood loss term, without relying on any generated rationales.

- **STaR (Zelikman et al., 2022)** is an early approach for generating, filtering, and learning rationales using a generative language model. It generates a rationale for each question and trains the language model on rationales that lead to correct predictions. Additionally, STaR introduces a rationalization process that provides hints when the initial rationale fails to produce a correct prediction.

- **RFT (Yuan et al., 2023)** stands for Rejection Sampling Fine-Tuning. RFT generates diverse rationales with a non-zero temperature and selects rationales to train based on binary feedback on the correctness of the final prediction. Unlike STaR, RFT does not have a rationalization process. In our experiments, $\boldsymbol{M_{\text{SFT}}}$ with maximum tolerance corresponds to RFT.

- **Self-motivated Learning (Feng et al., 2024)** exploits the inherent preference between correct rationales and incorrect rationales. It first trains a base model on generated and filtered rationales through supervised fine-tuning. It trains a reward model that assigns higher rewards to correct rationales than to incorrect ones. This reward model is then used to improve the reasoning performance of a supervised fine-tuned model through reinforcement learning using Proximal Policy Optimization (PPO) (Schulman et al., 2017).

4.3 CREST

- $\boldsymbol{M_{\text{SFT}}}$ is supervised fine-tuned on filtered rationales from the base model. The performance difference between this model and RFT demonstrates the effect of the rationale filtering process.

- $\boldsymbol{M_{\text{CREST}}}$ **& Fine-tune (Label)$_{\text{CREST}}$** are models trained using preference learning in CREST, based on $\boldsymbol{M_{\text{SFT}}}$ and Fine-tune (Label), respectively. To evaluate the effectiveness of preference learning with P_{total}, we apply it to two models: $\boldsymbol{M_{\text{SFT}}}$, a model fine-tuned on filtered rationales, and Fine-tune (Label), a model fine-tuned directly on ground truth labels. For details on the prompt templates used to train Fine-tune (Label)$_{\text{CREST}}$, please refer to Appendix F.3. The resulting models, named $\boldsymbol{M_{\text{CREST}}}$ and Fine-tune (Label)$_{\text{CREST}}$, demonstrate how CREST enhances reasoning performance through preference learning.

4.4 Results

As shown in Table 1, $\boldsymbol{M_{\text{CREST}}}$ outperforms other self-training baselines across the three datasets.

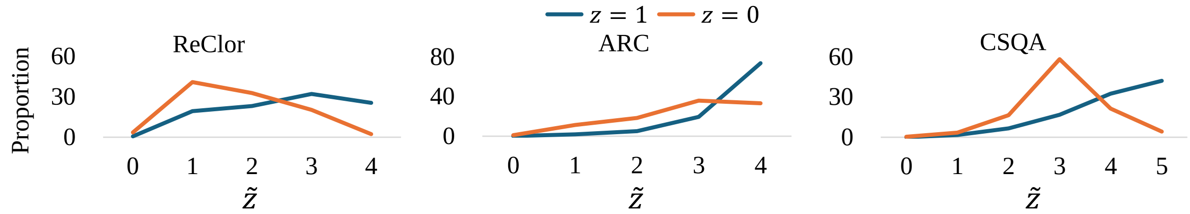

Figure 3: Distribution of rationale proportions based on $\tilde{z}$ for rationales with $z = 1$ and $z = 0$, respectively. For example, among the generated rationales with $z = 0$ for CSQA, approximately 60% have $\tilde{z} = 3$. Rationales with $z = 0$ are relatively concentrated at lower $\tilde{z}$ values compared to those with $z = 1$. This correlation between z and $\tilde{z}$ suggests that $\tilde{z}$ reflects the quality of the rationale.

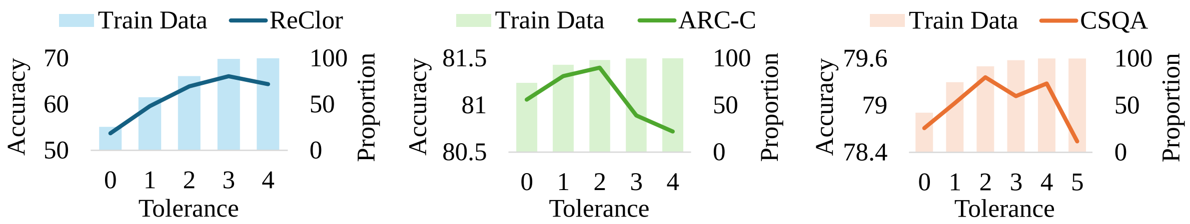

Figure 4: Proportion of rationale data used for training $\boldsymbol{M}_{\mathbf{SFT}}$ and task performance on three datasets, according to tolerance t. The results suggest that while moderate tolerance t improves performance, while overly high t values can degrade it, indicating the importance of excluding less robust rationales from training.

Both RFT and $\boldsymbol{M}_{\mathbf{SFT}}$ are models trained through supervised fine-tuning on the base model, with the key difference being whether rationale filtering based on $\tilde{z}$ was applied. The result that $\boldsymbol{M}_{\mathbf{SFT}}$ outperforms RFT across all three datasets demonstrates that rationale filtering based on $\tilde{z}$ consistently improves performance while reducing the amount of training data. Comparing $\boldsymbol{M}_{\mathbf{SFT}}$ with $\boldsymbol{M}_{\mathbf{CREST}}$, and Fine-tune (Label) with Fine-tune (Label)$_{\text{CREST}}$, we can see that preference learning with pairwise preference datasets constructed using follow-up questions consistently enhances performance across all three datasets.

5 Analysis

In this section, we explore the effectiveness of consistency-driven evaluation and the impacts of rationale filtering and preference learning in CREST on model performance, through analyses using the Llama 3 8B model as the base model. Our analysis includes examining the correlation between z and $\tilde{z}$ and conducting ablation studies on parameters such as t and λ to assess how the proposed methods in CREST contribute to performance improvement. To investigate the impact of preference learning with $P_{\tilde{z}}$, we create a model that trains $\boldsymbol{M}_{\mathbf{SFT}}$ using preference learning with only P_z, which we refer to as $\boldsymbol{M}_{\mathbf{SFT\ /w}\ \boldsymbol{P_z}}$, and compare it to $\boldsymbol{M}_{\mathbf{CREST}}$.

5.1 Incorrect Rationales on Follow-up Questions

To understand how evaluation through follow-up questions reflects the quality of rationales, we evaluate incorrect rationales ($z = 0$) generated from train datasets on the follow-up questions, as shown in Figure 3. The incorrect rationales are less robust on follow-up questions compared to correct rationales ($z = 1$), especially incorrect rationales have a significantly lower rate of getting all follow-up questions correct. This correlation between z and $\tilde{z}$ indicates that $\tilde{z}$ can reflect the quality of a rationale.

5.2 Effect of Tolerance t on Supervised Fine-Tuning

We investigate the impact of the tolerance value t during the supervised fine-tuning stage on task performance and the number of rationales used for training across the three datasets. Figure 4 shows the relationship between performance and the training data proportion based on the tolerance t. In the ARC-Challenge and CSQA datasets, performance improves as t increases, peaking at $t = 2$, and then tends to decrease as t continues to rise. This pattern shows that training on rationales that lead to incorrect predictions for most follow-up questions negatively affects task performance. At the maximum t value, accuracy is lower than at $t = 0$, where only 42% and 74% of the total gen-

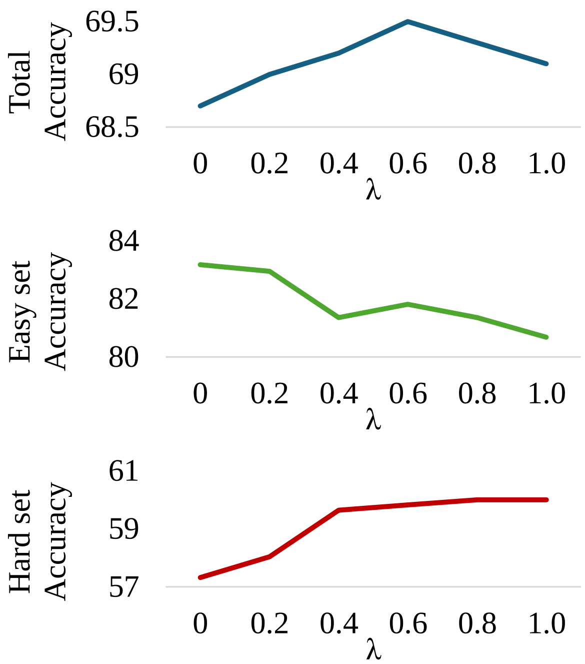

Figure 5: Task performance based on λ between P_z and $P_{\tilde{z}}$ in preference learning on ReClor. As λ increases, the model learns more from $P_{\tilde{z}}$ than from P_z, which leads to improved performance on the Hard set, while performance on the Easy set tends to decrease. Overall performance peaks at $\lambda = 0.6$, where the trade-off between the two datasets is balanced. These results suggest that preference learning on $P_{\tilde{z}}$ helps reduce the model's reliance on biases in the Easy set, enhancing the robustness of its reasoning ability.

erated rationales are used for training in CSQA and ARC, respectively. In ReClor, which requires more complex and broader logical reasoning, peak performance occurs at $t = 3$, differing from the other two datasets. However, including rationales with $\tilde{z} = 0$ in training leads to a decrease in performance. These results demonstrate that filtering out less robust rationales improves reasoning ability, even though it reduces the amount of training data.

5.3 Effect of λ on Preference Pair Dataset

To analyze how the two preference pair datasets, P_z and $P_{\tilde{z}}$, affect reasoning abilities through preference learning, we conduct experiments on ReClor using various λ values. As shown in Figure 5, we observe a trade-off where increasing λ improves performance on the Hard set but decreases performance on the Easy set. The overall performance peaks at $\lambda = 0.6$, where the trade-off is most balanced. Given that the ReClor Easy set consists of biased data points, preference learning on $P_{\tilde{z}}$ makes the model less dependent on these biases, thereby improving the robustness of its reasoning ability.

Model	Robustness	Correctness	Efficiency
RFT	2.66	3.17	2.88
M_{SFT}	2.92	3.28	3.29
$M_{\text{SFT /w } P_z}$	2.81	3.41	3.23
M_{CREST}	**2.95**	**3.51**	**3.33**

Table 2: Comparison of FLASK logical metrics for Llama 3 8B models trained using different methods on ReClor, evaluated with GPT-4o. The results show that CREST outperforms the baselines in all three metrics, especially in terms of rationale robustness.

5.4 Evaluating Quality of Rationales

To qualitatively evaluate how the CREST impacts the model's rationale generation, we randomly sample 100 questions from the ReClor validation set and evaluate the rationales from each model with GPT-4o. Following the methodology of Hwang et al. (2024), we employ FLASK (Ye et al., 2024), a fine-grained evaluation protocol for model-based evaluation, which exhibits a high correlation with human-based evaluation. Specifically, we focus on the 'logical thinking' category in FLASK, which encompasses three aspects: logical correctness, logical robustness, and logical efficiency. Logical correctness evaluates the model's ability to produce logically correct final answers. Logical robustness evaluates the generalizability of the step-by-step reasoning process without contradictions. Logical efficiency examines whether the reasoning process is concise and free of unnecessary steps. For the exact prompt templates used in the FLASK evaluation, please refer to Figures 11 and 12.

As shown in Table 2, CREST enhances rationale generation across all three aspects. Especially, rationale filtering in supervised fine-tuning improves the logical robustness and efficiency of the rationales. While preference learning on P_z makes $\boldsymbol{M_{\text{SFT}}}$ generate more logically correct rationales, it decreases the robustness of the rationales. However, preference learning on P_{total} yields higher performance across all three metrics compared to using only P_z. These evaluation results show that $\boldsymbol{M_{\text{CREST}}}$ generates more logically robust and correct rationales than the baselines.

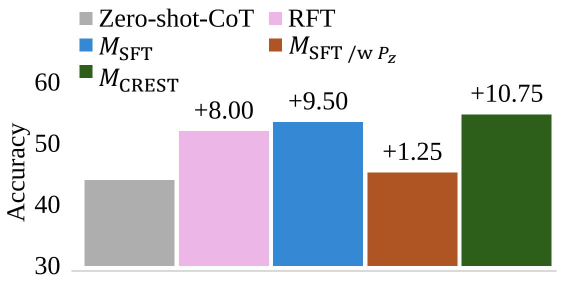

Figure 6: Comparison of follow-up questions accuracy across different training methods. The numbers above each bar indicate the absolute accuracy improvement over Zero-shot-CoT. The performance gain shows that CREST trains the LLM to generate rationales that are more robust at follow-up questions.

5.5 Evaluating CREST Models on Follow-up Questions

We evaluate the rationales generated by each trained model for the original questions in the ReClor validation set using follow-up questions, which is shown in Figure 6. As in the Rationale Generation and Evaluation stage, we input the generated rationales and follow-up questions into the base model (Llama 3 8B), then measure accuracy over all follow-up questions. To assess how different training methods affect the rationale generation, we employ Zero-shot-CoT (Kojima et al., 2022) as a baseline model. The improvement between RFT and $M_{\mathbf{SFT}}$ shows the effect of rationale filtering in generating rationales that are more robust to follow-up questions. As shown in Figure 6, CREST trains the LLM to generate rationales that are more robust to follow-up questions.

6 Conclusion

In this paper, we propose CREST, a novel self-training framework that evaluates generated rationales in a fine-grained manner by letting the LLM solve follow-up questions derived from the original question. We propose two methods for utilizing the evaluation results in training: filtering out less consistent rationales for supervised fine-tuning and employing preference learning to favor more consistent rationales over less consistent ones. Experimental results on three question-answering datasets show that CREST enables an LLM to generate more correct and robust rationales and achieves better performance compared to previous approaches.

7 Limitations

The main idea of our proposed framework CREST is to evaluate rationales with multiple follow-up questions, which is conceptually task-agnostic. In this paper, we assume a multiple-choice question-answering task as the primary setting. However, there are other types of tasks that differ significantly in structure and may require adaptations of our framework to maintain its effectiveness. For future work, we plan to extend the CREST beyond multiple-choice question-answering, applying it to scenarios such as math questions (Cobbe et al., 2021) or open-ended questions (Ling et al., 2023) where choices are not provided.

We treat all follow-up questions equally and focus solely on the number of follow-up questions answered correctly to calculate the additional reward $\tilde{z}$. However, since each follow-up question asks whether a given option is correct, the interpretation of follow-up questions for correct and incorrect answers can differ. For instance, consider two rationales that receive the same reward, $\tilde{z} = 2$, for a question with the correct answer being A. The first rationale accurately answers the follow-up questions about the correct option (A) and an incorrect option (B), while the second rationale accurately answers the follow-up questions about two incorrect options (B and C). Although both rationales receive the same reward, their interpretations differ: the first rationale provides information about the correct answer, whereas the second does not. This difference in interpretation may affect rationale evaluation and training. Kawabata and Sugawara (2023) show the differences in LLMs' ability to handle each option, revealing that LLMs struggle with questions related to incorrect answers, whereas questions related to correct answers are easier for them. Future research could exploit this difference to further extend CREST.

Additionally, while our study primarily focuses on self-training of language models, the methods we propose for evaluating rationales and leveraging these evaluations during training can be applied to broader scenarios such as distilling reasoning abilities from larger teacher models to smaller student models (Liu et al., 2023; Shridhar et al., 2023; Hsieh et al., 2023).

8 Acknowledgements

We would like to thank the anonymous reviewers for their helpful questions and comments. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00509258, Global AI Frontier Lab, No. RS-2024-00398115, Research on the reliability and coherence of outcomes produced by Generative AI and No. RS-2019-II190421, AI Graduate School Support Program(Sungkyunkwan University)) and abductive inference framework using omni-data for understanding complex causal relations & ICT Creative Consilience program (2022-0-00680).

References

AI@Meta. 2024. Llama 3 model card.

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics.

Guangsheng Bao, Hongbo Zhang, Linyi Yang, Cunxiang Wang, and Yue Zhang. 2024a. Llms with chain-of-thought are non-causal reasoners. *Preprint*, arXiv:2402.16048.

Qiming Bao, Alex Yuxuan Peng, Zhenyun Deng, Wanjun Zhong, Gael Gendron, Timothy Pistotti, Neset Tan, Nathan Young, Yang Chen, Yonghua Zhu, Paul Denny, Michael Witbrock, and Jiamou Liu. 2024b. Abstract meaning representation-based logic-driven data augmentation for logical reasoning. *Preprint*, arXiv:2305.12599.

Angelica Chen, Jason Phang, Alicia Parrish, Vishakh Padmakumar, Chen Zhao, Samuel R. Bowman, and Kyunghyun Cho. 2024a. Two failures of self-consistency in the multi-step reasoning of LLMs. *Transactions on Machine Learning Research*.

Yanda Chen, Chandan Singh, Xiaodong Liu, Simiao Zuo, Bin Yu, He He, and Jianfeng Gao. 2024b. Towards consistent natural-language explanations via explanation-consistency finetuning. *Preprint*, arXiv:2401.13986.

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. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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

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. *Preprint*, arXiv:2110.14168.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Yunlong Feng, Yang Xu, Libo Qin, Yasheng Wang, and Wanxiang Che. 2024. Improving language model reasoning with self-motivated learning. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8840–8852, Torino, Italia. ELRA and ICCL.

Lukas Fluri, Daniel Paleka, and Florian Tramèr. 2024. Evaluating superhuman models with consistency checks. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 194–232.

Matthew Douglas Hoffman, Du Phan, david dohan, Sholto Douglas, Tuan Anh Le, Aaron T Parisi, Pavel Sountsov, Charles Sutton, Sharad Vikram, and Rif A. Saurous. 2023. Training chain-of-thought via latent-variable inference. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STaR: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*.

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*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. 2024. Self-explore to avoid the pit: Improving the reasoning capabilities of language models with fine-grained rewards. *CoRR*, abs/2404.10346.

Myeongjun Jang, Deuk Sin Kwon, and Thomas Lukasiewicz. 2022. BECEL: Benchmark for consistency evaluation of language models. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3680–3696, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*.

Nora Kassner, Oyvind Tafjord, Ashish Sabharwal, Kyle Richardson, Hinrich Schuetze, and Peter Clark. 2023. Language models with rationality. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14190–14201, Singapore. Association for Computational Linguistics.

Nora Kassner, Oyvind Tafjord, Hinrich Schütze, and Peter Clark. 2021. BeliefBank: Adding memory to a pre-trained language model for a systematic notion of belief. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8849–8861, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Akira Kawabata and Saku Sugawara. 2023. Evaluating the rationale understanding of critical reasoning in logical reading comprehension. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

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*, volume 35, pages 22199–22213.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Chen Ling, Xuchao Zhang, Xujiang Zhao, Yanchi Liu, Wei Cheng, Mika Oishi, Takao Osaki, Katsushi Matsuda, Haifeng Chen, and Liang Zhao. 2023. Open-ended commonsense reasoning with unrestricted answer candidates. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. 2023. Logicot: Logical chain-of-thought instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2908–2921.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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 *Thirty-seventh Conference on Neural Information Processing Systems*.

Marco Tulio Ribeiro, Carlos Guestrin, and Sameer Singh. 2019. Are red roses red? evaluating consistency of question-answering models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6174–6184, Florence, Italy. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073, Toronto, Canada. Association for Computational Linguistics.

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 A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, 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. *Transactions on Machine Learning Research*. Expert Certification.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

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.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Haochun Wang, Sendong Zhao, Zewen Qiang, Nuwa Xi, Bing Qin, and Ting Liu. 2024. Beyond the answers: Reviewing the rationality of multiple choice question answering for the evaluation of large language models. *Preprint*, arXiv:2402.01349.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Yeo Wei Jie, Teddy Ferdinan, Przemyslaw Kazienko, Ranjan Satapathy, and Erik Cambria. 2024. Self-training large language models through knowledge detection. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15033–15045, Miami, Florida, USA. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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, Online. Association for Computational Linguistics.

Zihang Xu, Ziqing Yang, Yiming Cui, and Shijin Wang. 2023. IDOL: Indicator-oriented logic pre-training for logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8099–8111, Toronto, Canada. Association for Computational Linguistics.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2024. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *The Twelfth International Conference on Learning Representations*.

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

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. *Preprint*, arXiv:2308.01825.

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

Danna Zheng, Danyang Liu, Mirella Lapata, and Jeff Z. Pan. 2024. Trustscore: Reference-free evaluation of llm response trustworthiness. *Preprint*, arXiv:2402.12545.

A Phi-2 Experiment Results

To demonstrate the robustness of CREST, we also test CREST with Phi-2 model[4] (Javaheripi et al., 2023). Phi-2 has 2.7B parameters, which is much smaller compared to Llama 3 8B and Gemma 7B which have 8.0B and 8.5B parameters, respectively. As shown in Table 3, CREST outperforms other self-training baselines across the three datasets, and preference learning to Fine-tune (Label) model consistently improves performance. This result shows that CREST can function effectively with this relatively small model.

B Evaluating CoT Performance in Zero- and Few-Shot Settings

To measure the accuracy of $\boldsymbol{M}$ itself using Chain-of-Thoughts (CoT) without fine-tuning, we conduct experiments with $\boldsymbol{M}$. Specifically, we examine the performance of M instructed to generate the rationale and prediction, represented as $(r, p) = \boldsymbol{M}(q)$. We refer to these approaches as Zero-shot-CoT (instructed to generate a rationale and prediction without examples) and Few-shot-CoT (given few-shot examples and then instructed to generate a rationale and prediction). The input-output format used for those CoT models is the same as the input-output format of $\boldsymbol{M}, \boldsymbol{M}_{\mathbf{SFT}}, \boldsymbol{M}_{\mathbf{CREST}}$.

As shown in Table 4, these CoT approaches underperformed compared to their non-CoT counterparts in many cases. Some previous studies support this performance degradation. Wei et al. (2022) show that models with size not big enough would not benefit from chain-of-thought reasoning. Some studies (Bao et al., 2024b; Xu et al., 2023) have reported common performance degradation with CoT approaches in complex reasoning tasks.

C Data and Rationale Statistics

Table 5 describes the number of examples in train, validation, and test splits for the data we use. Additionally, Table 6 shows the number of rationales generated in the rationale generation stage in our experiments according to the z and $\tilde{z}$ values. Since the official test set of CSQA is evaluated every two weeks, we use the official Dev set as the test set in our experiment and extract a new validation set with the same number of samples from the train set.

[4] https://huggingface.co/microsoft/phi-2

Algorithm 1 Formation of Preference Pairs

$P_z \leftarrow []$ {initialize z-based preference pairs}
$P_{\tilde{z}} \leftarrow []$ {initialize $\tilde{z}$-based preference pairs}
for all question $q_i \in \mathcal{D}$ **do**
  for all $(w, l) \in \{(w, l) | 1 \leq w \leq N, 1 \leq l \leq N\}$ **do**
    if $z_i^w = 1$ **and** $z_i^l = 0$ **then**
      P_z+=$\{q_i, [r_i^w, p_i^w], [r_i^l, p_i^l]\}$
    end if
    if $z_i^w = z_i^l = 1$ **and** $\tilde{z}_i^w > \tilde{z}_i^l$ **then**
      $P_{\tilde{z}}$+=$\{q_i, [r_i^w, p_i^w], [r_i^l, p_i^l]\}$
    end if
  end for
end for

D Rationale Generation and Evaluation Case Study

Table 7 shows an example of generated rationales from a CSQA question and their evaluation. We can see the rationale which leads to an incorrect answer to the question ($z = 0$) represents incorrect reasoning steps and conclusion. The rationale with $\tilde{z} = 2$ leads to the correct answer D but does not show a convincing reasoning process, causing readers to be confused between C and D. In contrast, the rationales with higher rewards of $\tilde{z} = 4$ and $\tilde{z} = 5$ provide more convincing reasoning processes. They offer a comprehensive explanation for arriving at the correct answer D and include judgments about why other choices are incorrect, respectively.

E Preference Pair Datasets Construction Algorithm

This section presents a more detailed algorithm for constructing the preference pair dataset used in preference learning. As shown in Algorithm 1, we construct two preference pair sets, P_z and $P_{\tilde{z}}$, based on z and $\tilde{z}$, respectively.

F Prompts

In this section, we introduce the prompt templates used for rationale generation, inference, and evaluation with FLASK. We construct input text for the language model based on these templates. All the

Approach	Model	ReClor	ARC-C	CSQA
	Zero-shot	42.00	68.51	57.74
	Few-shot	43.10	71.84	62.41
Self-training	STaR	56.00	79.52	72.89
	RFT	55.90	78.67	74.37
	Self-motivated Learning	55.50	79.01	75.43
	M_{SFT}	57.50	79.61	75.02
	M_{CREST}	**59.20**	**79.86**	**75.51**
Direct fine-tuning	Fine-tune (Label)	66.20	78.33	76.25
	Fine-tune (Label)$_{\text{CREST}}$	**67.20**	**79.78**	**76.90**

Table 3: Accuracy of various models across three reasoning datasets with phi-2 model. Test-E and Test-H denote Easy and Hard sets in ReClor Test dataset, respectively.

Base model	Model	ReClor	ARC-C	CSQA
Llama 3 8B	Zero-shot	52.10	69.28	53.89
	Zero-shot-CoT	45.70	63.74	48.57
	Few-shot	55.30	77.21	70.76
	Few-shot-CoT	40.60	74.23	71.25
Gemma 7B	Zero-shot	53.60	77.47	65.68
	Zero-shot-CoT	43.50	65.96	48.24
	Few-shot	58.70	83.11	75.02
	Few-shot-CoT	51.90	80.29	72.89
Phi-2	Zero-shot	42.00	68.51	57.74
	Zero-shot-CoT	38.40	66.01	52.17
	Few-shot	43.10	71.84	62.41
	Few-shot-CoT	43.00	75.68	70.02

Table 4: Accuracy of base models in zero-shot and few-shot settings, with and without CoT prompting, on the three reasoning datasets. In many cases, CoT prompting results in performance degradation.

Dataset	Train	Valid	Test
ReClor	4,638	500	1,000
ARC	3,370	869	1,172
CSQA	8,520	1,221	1,221

Table 5: Data statistics of the datasets we use in this paper. Train, Valid, and Test mean the number of samples in each split.

prompt templates we present are designed for the ReClor dataset (Yu et al., 2020). Unlike ReClor, the ARC (Clark et al., 2018) and CSQA (Talmor et al., 2019) datasets do not include a passage, so we use different prompt templates for them. As a result, the [Question] part in the prompt templates for ARC and CSQA consists only of the question and the answer choices.

F.1 Rationale Generation and Evaluation

We use the prompt template shown in Figure 7 as input to the language model to generate rationales. For generating answer predictions from a given rationale, we use the prompt template in Figure 8.

F.2 Follow-up Questions

Figure 9 shows the prompt template for follow-up questions. The language model is instructed to judge whether the given '*(target option)*' is correct or not with the given generated rationale.

F.3 Training Fine-tune (Label)$_{\text{CREST}}$

Fine-tune (Label)$_{\text{CREST}}$ is obtained by training Fine-tune (Label) on rationale preferences. Since Fine-tune (Label) is trained through supervised fine-tuning to directly predict answers, Fine-tune (Label)$_{\text{CREST}}$ undergoes training with two different prompt templates. In the supervised fine-tuning stage, Fine-tune (Label)$_{\text{CREST}}$ is trained using the prompt template in Figure 10, while in the preference learning stage, it is trained using the prompt template in Figure 7.

F.4 Evaluating Models

Figure 7 shows the prompt template used for evaluating models in self-training approaches (Table 1) as well as Zero-shot-CoT and Few-shot-CoT models (Table 4). Figure 10 shows the prompt template for direct answering, where models are provided with a question and tasked with predicting the answer directly, without generating rationales. This template is used to evaluate Zero-shot, Few-shot, and direct fine-tuning methods, as detailed in Table 1.

Base model	Dataset	z=0	z=1					
			$\tilde{z}$=0	$\tilde{z}$=1	$\tilde{z}$=2	$\tilde{z}$=3	$\tilde{z}$=4	$\tilde{z}$=5
Llama 3 8B	ReClor	39,679	224	6,621	7,917	11,021	8,737	-
	ARC	10,884	63	770	2,170	8,309	31,541	56
	CSQA	51,868	38	1,426	5,571	14,192	27,484	35,530
Gemma 7B	ReClor	38,635	169	16,611	7,384	6,625	4,623	-
	ARC	8,931	121	5,265	7,251	12,262	19,785	29
	CSQA	47,336	71	7,136	12,825	18,193	23,389	27,301
Phi-2	ReClor	42,496	221	21,936	6,079	2,682	794	-
	ARC	7,398	36	1,822	4,371	11,325	28,916	52
	CSQA	52,790	100	4,934	10,919	19,297	26,135	22,145

Table 6: The number of rationales generated from the train sets of each dataset during the rationale generation and evaluation stages in the experiments of this paper, presented according to the z and $\tilde{z}$ values. In the case of the ARC dataset, most of the questions in the train split have 4 answer choices, resulting in a very low number of rationales for $\tilde{z}$=5.

F.5 Prompt and Example of Qualitative Analysis with FLASK

We use the prompt template shown in Figure 11 and Figure 12 for the qualitative analysis with GPT-4o, as suggested by Ye et al. (2024). Figure 13 shows an example of a response from GPT-4o. To measure the scores, we automatically extract the Python dictionary portion from the output.

G Implementation Details

We use lora rank=16, alpha=16 and target modules = {gate_proj, down_proj, up_proj, q_proj, k_proj, v_proj, o_proj}. We use cosine scheduler and adamW optimizer (Loshchilov and Hutter, 2017). For saving memory, we use half-precision (fp16) in training $M_{\mathbf{SFT}}$. During inference, if the model fails to fully generate the answer label within the designated generation length, we clarify the prediction by appending '[Answer] Therefore, the answer is' to the end of the initial output and conduct an additional query. We select models that show the highest performance on the validation set without early-stopping. For Llama 3 8B experiments on ReClor, the best-found hyperparameter values for the supervised fine-tuning stage were: learning rate=8e-4, batch size=8, tolerance=3. For the preference learning stage, the best-found hyperparameter values were λ=0.6, learning rate=6e-6, and max number of steps=5000. Our hardware setting is Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz (CPU), and NVIDIA RTX A6000 (GPU). We use *vllm* (Kwon et al., 2023) library for efficient rationale generation and evaluation. We use *trl* (von Werra et al., 2020) library for supervised fine-tuning and preference learning stages. For the datasets we use in this paper, CSQA is under the MIT license, and ARC is under the CC BY-SA 4.0 license. You can see terms for use of ReClor in here. We use these datasets and the models solely for research purposes.

H Computational Costs

In this section, we present the overall computational costs of our experiments, measured in GPU hours. Using the Llama 3 8B model and the ReClor dataset, the computational costs are as follows:

- Rationale Generation: 12 GPU hours.
- Rationale Evaluation: 3.2 GPU hours.
- Supervised Fine-Tuning: 7.4 GPU hours.
- Preference Learning: 19.2 GPU hours.

In the rationale evaluation stage, inference for the original questions (q) took approximately 1 hour, while inference for follow-up questions ($\tilde{q}$) required about 2.2 hours.

I Adjustments in Implementation of Baseline Models

Some of the baseline approaches target domains and environments that differ from our setting; therefore we adjust them to fit our task setup while preserving their core ideas. First, although STaR (Zelikman et al., 2022) is an iterative process, we

Question	You can share files with someone if you have a connection to a what? A. freeway B. radio C. wires **D. computer network** E. electrical circuit
$\tilde{z} = 5$ $z = 1$	A freeway is a long stretch of road. Radio is for broadcasting. Wires are for electric power transmission, distribution and telecommunications. Computer network is a telecommunications network which allows computers to exchange data. So, the answer is computer network.
$\tilde{z} = 4$ $z = 1$	We know that a computer network is a group of computers that are connected with each other. You can share files with someone if you have a connection to a computer network. This means that a computer network is a group of computers that are connected with each other.
$\tilde{z} = 3$ $z = 1$	To share files, you need to connect to a computer network. A computer network is a network of computers that allows the users to share resources such as files, printers, and applications.
$\tilde{z} = 2$ $z = 1$	To share files with someone, you need to have a connection to a computer network. You can share files by using a computer. You can connect to a computer network with wires. With wires, you can connect to a computer network and share files.
$z = 0$	To share files with someone, we need to have a computer network. We can share files by using a computer network and transferring the files from one computer to another. So, we need to have a computer network to share files. A computer network is made up of computers connected by wires. So, we need wires to share files. So, our answer is wires.

Table 7: Examples of generated rationales and corresponding rewards z and $\tilde{z}$ from consistency-driven rationale evaluation for a CSQA question. The colored texts represent the incorrect parts of the rationales.

do not perform iterations in order to ensure a fair comparison with other models. RFT (Yuan et al., 2023) is an approach that generates diverse reasoning paths, and only the reasoning paths that produce correct answers are selected to train language models. RFT requires an initial generator to generate reasoning paths. Since it was designed for GSM8K (Cobbe et al., 2021), a mathematical reasoning dataset that includes reasoning paths in its training set, the generator in the original RFT is obtained by training a base model on these reasoning paths. However, since our dataset does not include reasoning paths, we generate rationales using few-shot prompting with the base model instead. They also verify the selected reasoning paths by executing the equations included in them using a Python interpreter, a step that is not feasible in our experiments.

Input

[Instruction]
(instruction here)

[Question]
<Passage> *(passage here)*
<Question> *(question here)*
Answer Choices:
A. *(option A here)*
B. *(option B here)*
C. *(option C here)*
D. *(option D here)*

[Rationale]
Let's think step by step.

Output

(generated rationale here)

[Answer]
Therefore, the answer is *(answer label here)*.

Figure 7: Prompt template for rationale generation and inference. This template is used for generating rationales and evaluating models in self-training approaches (Table 1), as well as Zero-shot-CoT and Few-shot-CoT models (Table 4).

Input

[Instruction]
(instruction here)

[Question]
<Passage> *(passage here)*
<Question> *(question here)*
Answer Choices:
A. *(option A here)*
B. *(option B here)*
C. *(option C here)*
D. *(option D here)*

[Rationale]
Let's think step by step.
(*generated rationale here*)

[Answer]
Therefore, the answer is

Output

(answer label here).

Figure 8: Prompt template for deriving an answer prediction from a given rationale. The answer prediction is compared to the ground truth to evaluate each generated rationale and calculate the reward z for it.

Input

[Instruction]
(instruction here)

[Question]
<Passage> *(passage here)*
<Question> *(question here)*
Answer Choices:
A. *(option A here)*
B. *(option B here)*
C. *(option C here)*
D. *(option D here)*

Is a given choice *(target option)* the correct answer?

[Rationale]
Let's think step by step.
(*generated rationale here*)

[Answer]
Therefore, *(target option)* is

Output

(the/not the) correct answer.

Figure 9: Prompt template for evaluation using follow-up questions. This template evaluates a given rationale by prompting models to solve a follow-up question based on the rationale. As shown in the input part, the follow-up question asks whether the target option is the correct answer to the original question. Results for all target answer choices are aggregated to validate the given rationale and compute the reward $\tilde{z}$.

Input

[Instruction]
(instruction here)

[Question]
<Passage> *(passage here)*
<Question> *(question here)*
Answer Choices:
A. *(option A here)*
B. *(option B here)*
C. *(option C here)*
D. *(option D here)*

[Answer]
The correct answer is

Output

(answer label here).

Figure 10: Prompt template for direct answer prediction. This template is used to evaluate Zero-shot, Few-shot, and Direct fine-tuning approaches (Table 1). Unlike other templates, it does not require models to generate or utilize rationales.

We would like to request your feedback on the performance of the response of the assistant to the user instruction displayed below. In the feedback, I want you to rate the quality of the response in these 3 categories according to each score rubric:

[Skill 1. Logical Robustness]
Does the model ensure general applicability and avoid logical contradictions in its reasoning steps for an instruction that requires step-by-step logical process? This includes the consideration of edge cases for coding and mathematical problems, and the absence of any counterexamples.
Score 1: The logic of the model's response is completely incoherent.
Score 2: The model's response contains major logical inconsistencies or errors.
Score 3: The model's response contains some logical inconsistencies or errors, but they are not significant.
Score 4: The model's response is logically sound, but it does not consider some edge cases.
Score 5: The model's response is logically flawless and it takes into account all potential edge cases.

[Skill 2. Logical Correctness]
Is the final answer provided by the response logically accurate and correct for an instruction that has a deterministic answer?
Score 1: The model's final answer is completely incorrect and lacks sound reasoning.
Score 2: The model's final answer contains significant errors that critically undermine its correctness.
Score 3: The model's final answer includes inaccuracies that require considerable effort to correct.
Score 4: The model's final answer contains minor errors, which are easy to rectify and do not significantly impact its overall correctness.
Score 5: The model's final answer is completely accurate and sound.

Figure 11: Prompt template for the FLASK evaluation. (1/2)

[Skill 3. Logical Efficiency]
Is the response logically efficient? The logic behind the response should have no redundant step, remaining simple and efficient. For tasks involving coding, the proposed solution should also consider time complexity.
Score 1: The logic behind the response is significantly inefficient and redundant, necessitating a complete reorganization of logic for clarity and efficiency.
Score 2: The logic of the response lacks efficiency and conciseness, requiring a substantial reorganization for better optimization.
Score 3: The logic of the response is not efficient enough, necessitating major edits for improved optimization.
Score 4: The logic of the response is largely efficient, but it still has some redundant steps. It could be handled from minor edits for better optimization.
Score 5: The logic of the response is optimally efficient, requiring no further optimization.

[Instruction]
{question}

[Ground truth Answer]
{ground truth answer}

[Assistant's Response]
{rationale and prediction}

[The End of Assistant's Response]
Please give feedback on the assistant's responses. Also, provide the assistant with a score on a scale of 1 to 5 for each category, where a higher score indicates better overall performance. Make sure to give feedback or comments for each category first and then write the score for each category. Only write the feedback corresponding to the score rubric for each category. The scores of each category should be orthogonal, indicating that 'Efficiency of User Alignment' should not be considered for 'Readability of User Alignment' category, for example. Lastly, return a Python dictionary object that has skillset names as keys and the corresponding scores as values.

Figure 12: Prompt template for the FLASK evaluation. (2/2)

Feedback:

Skill 1. Logical Robustness:
The assistant's response does capture the fundamental logical connection between the apparent discrepancy and the selected answer choice by identifying the potential cause for the paradox. However, it does not explicitly address alternative possibilities or examine each of the provided choices. Therefore, the response does not explore all potential edge cases or fully determine why B is the most fitting choice among the others.

Score: 4

Skill 2. Logical Correctness:
The assistant's final answer is logically correct. It accurately concludes that government success in removing counterfeit bills from circulation has made merchants and bank tellers lax in checking for counterfeit bills—this fits well with the provided ground truth answer and the context of the question.

Score: 5

Skill 3. Logical Efficiency:
The response is rather succinct, but it lacks depth in contemplating why alternative choices are not the best fit or how the logic follows without redundancy. However, the response does directly lead to the right conclusion without unnecessary steps.

Score: 4

Scores:
```python
{
    “Logical Robustness”: 4,
    “Logical Correctness”: 5,
    “Logical Efficiency”: 4
}
```

Figure 13: A result of GPT-4o FLASK evaluation for a generated rationale. The input prompt is shown in Figure 11 and Figure 12.