Improving Large Language Models via Fine-grained Reinforcement Learning with Minimum Editing Constraint

Anonymous ACL submission

Abstract

Reinforcement learning (RL) has been widely used in training large language models (LLMs) for preventing unexpected outputs, *e.g.,* reducing harmfulness and errors. However, existing RL methods mainly adopt instance-level reward, which cannot provide fine-grained supervision for complex reasoning tasks. As a result, the RL training cannot be fully aware of the specific part or step that actually leads to the incorrectness in model response. To address it, we propose a new RL method named **RLMEC** that incorporates a generative model as the reward model, which is trained by the erroneous solution rewriting task under the minimum editing constraint, which can produce token-level supervision for RL training. Based on the generative reward model, we design the token-level RL objective for training and an imitation-based regularization for stabilizing RL process. And these two objectives focus on the revision of the key tokens for the erroneous solution, reducing the effect of other unimportant tokens. Experiment results on 8 tasks have demonstrated the effectiveness of our approach. Our code and data will be publicly released.

1 Introduction

Owing to unsupervised pre-training on large-scale text corpora, large language models (LLMs) have shown remarkable performance on various text generation tasks (Zhao et al., 2023a; Google, 2023), such as question answering, summarization and translation (OpenAI, 2023). To further improve the task solving capacity, researchers (Touvron et al., 2023; Bai et al., 2023) have proposed supervised fine-tuning (SFT) and reinforcement learning (RL) methods, which can better adapt LLMs to specific domains or downstream tasks after pre-training. Typically, SFT methods (Ouyang et al., 2022; Longpre et al., 2023) incorporate annotated input-output pairs (*e.g.,* question and solution, instruction and response) to train the LLM for learning the sequence-to-sequence pattern; RL methods (Schulman et al., 2017; Christiano et al., 2017) adopt a reward model to measure the quality of the generated outputs from the LLM, and then guide its training for maximizing and minimizing the expectation of generating high-quality and low-quality ones, respectively.

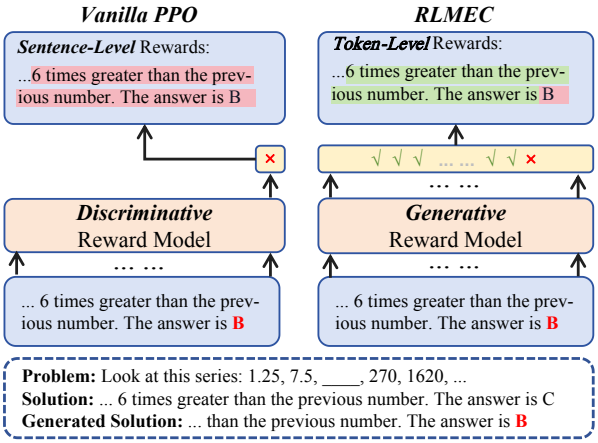


Figure 1: The comparison of our generative reward model and the traditional discriminative one in PPO. Red and green background colors denote negative and positive rewards, respectively.

As RL methods are capable of directly reducing the probability of LLMs for producing unexpected outputs, they have been widely used in optimizing LLMs towards better human alignment (*e.g.,* reducing harmfulness) and stronger ability (*e.g.,* reducing errors (Luo et al., 2023; Wang et al., 2023b)). Generally, RL methods first train a discrimination model for distinguishing desirable and undesirable outputs. Then, the model is used to produce the reward scores for the sampled outputs from the LLM, and the LLM would be trained by encouraging and punishing the generation of high-score and low-score ones accordingly.

Despite the success, as existing RL methods mostly utilize instance-level reward for each sampled output, it is often difficult to provide accurate fine-grained supervision on complex reasoning tasks (*e.g.,* mathematical reasoning). Concretely,

given a complex task, the sampled outputs from the LLM tend to be highly similar in surface expression, only with key differences in few specific words or steps (Yuan et al., 2023) that determine the correctness. We argue that instance-level RL approaches (Ouyang et al., 2022; Christiano et al., 2017; Zheng et al., 2023b) have two major limitations. First, as the unimportant parts would often occupy a large amount of supervision signals, instance-level rewards can not accurately emphasize the more important evidence related to correctness, leading to inefficient or redundant supervision. Second, the paired correct and incorrect outputs may share the overlapping content but receive opposite optimization goals, which may lead to the optimization conflict issue on such overlapped content, making it still infeasible to provide accurate fine-grained supervisions.

To address these issues, in this paper, we propose a novel method, *Reinforcement Learning with Minimum Editing Constraint* (**RLMEC**), to improve the training of LLMs by fine-grained supervision signals. Our approach is inspired by the homework correction process of professional teachers, in which she/he first identifies the incorrect parts and then provides necessary revisions or comments accordingly. Following such an idea, we train a generative reward model by an erroneous solution rewriting task under the constraint of minimum editing distance. The reward model plays a similar role to teachers by producing fine-grained supervision, *i.e.,* token-level quality assessment scores. Instead of using a new demonstration as positive, our reward model tries to correct the output with minimum edits. Specially, we utilize the specially trained reward model to produce the token probabilities for computing the token-level rewards, and optimize the LLM using the proximal policy optimization method (PPO) (Schulman et al., 2017). In this way, by contrasting the original and corrected outputs, the LLM would be instructed more informatively, thus becoming aware of the correct way to generate the response. Figure 1 illustrates the comparison of the Vanilla PPO and our proposed RLMEC approach.

The major novelty of this paper lies in the incorporation of a generative reward models with minimum editing constraint for RL training of LLMs. Table 1 presents the major differences between our method and previous work. To evaluate the effectiveness of our methods, we conduct the experiment on two types of complex reasoning tasks, *i.e.,* question answering (Aggarwal et al., 2021; Mihaylov et al., 2018a) and mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021c). In these evaluation tasks, our RLMEC mostly outperforms other competitive SFT and RL methods, based on 7B and 13B LLMs. Moreover, our analysis experiments also show that our method is able to stabilize the RL training process and reduce the erroneous steps in the sampled outputs of LLMs.

2 Related Work

Reinforcement Learning for LLMs. With the development of the LLMs, reinforcement learning (RL) (Christiano et al., 2017; Ziegler et al., 2019) is widely utilized to further improve the ability of LLMs. Proximal Policy Optimization (PPO) (Schulman et al., 2017) is the traditional algorithm to employ RL. To provide fine-grained supervision signals, previous work (Mnih et al., 2016; Zheng et al., 2023b) utilizes the critic model to calculate the reward of the current stage. Because of the instability of the training procedure of reinforcement learning, recent work (Rafailov et al., 2023; Liu et al., 2023a; Lu et al., 2022; Zhao et al., 2023b) has utilized supervised-finetuning (SFT) to simulate the RL procedure. These methods fuse the quality of the responses into the supervision signals. Moreover, existing work (Uesato et al., 2022; Luo et al., 2023; Wang et al., 2023b,a; Yang et al., 2023) has found that process-supervision signals can better guide the training process of LLMs. Besides, other methods (Swamy et al., 2024; Chen et al., 2024) improve the ability of LLMs during self-play procedure. In this work, we proposed a new RL framework with generative reward model to directly provide the fine-grained supervisions, which enable to focus on few key tokens.

LLMs for Reasoning. Previous work utilizes two types of methods (*i.e.,* prompting and training) to enhance the reasoning ability of LLMs. For the prompting methods, Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) guides LLMs to generate the intermediate reasoning steps before generating the final answer. Based on CoT, previous work decomposes the problem into several simple sub-problems (Dua et al., 2022), utilizes the external tools to help LLMs (Gao et al., 2022; Yao et al., 2022; Schick et al., 2023), designs the specific agents to perform reasoning (Yin

et al., 2023; Du et al., 2023), or post-process the generated response (Madaan et al., 2023; Wang et al., 2022). Besides, existing work also guides LLMs to perform reasoning in the specific structure, *e.g.,* tree (Yao et al., 2023; Ding et al., 2023) or graph (Besta et al., 2023). For the training methods, previous work (Lewkowycz et al., 2022) has leveraged domain-specific data to fine-tune the LLMs. Because of the limitation of the training data, the data generated by teacher model (*e.g.,* GPT-4) is utilized to augment the training data (Yue et al., 2023; Yu et al., 2023; Gou et al., 2023). In this work, we aim to train the LLMs via fine-grained RL to improve their reasoning ability.

3 Preliminary

In this work, we focus on improving the performance of LLMs on complex reasoning tasks with reinforcement learning (RL) algorithm. Typically, complex reasoning tasks require LLMs to perform step-by-step reasoning (*e.g.,* chain-of-thought (Wei et al., 2022; Kojima et al., 2022)) for each question, where LLMs progressively generate the solution for reaching the answer. In this process, LLMs are prone to make mistakes at the intermediate steps, which likely lead to totally wrong answer (Bang et al., 2023; Zhang et al., 2023). Our goal is to optimize a pre-trained LLM using RL algorithm, to reduce its errors and improve the task performance.

Formally, we are given a collection of question-solution pairs, denoted as $\mathcal{D} = \{\langle q_i, s_i \rangle\}_{i=1}^{n}$, where each question and solution are both composed by a sequence of tokens, denoted as $\{t_0, \cdots, t_m\}$. Then, we follow the proximal policy optimization (PPO) framework (Schulman et al., 2017) for RL, and make improvements about reward model and training loss. In PPO, the LLM to be optimized is the *policy model*, and its original parameters would be copied to compose the *reference model*. During training, the reference model outputs the sampled solutions for the given question, denoted as $\hat{s}$, and then the policy model would learn from the feedback from a *reward model*, which produces the reward $R_{\hat{s}}$ for the sampled output $\hat{s}$. Based on it, the parameters of the policy model will be optimized to maximize the reward expectation of all the sampled outputs, and the target function is:

$$\mathcal{J}(\theta) = \sum_{i=1}^{n} r(q_i, \hat{s}_i) \times R_{\hat{s}_i}, \quad r(q_i, \hat{s}_i) = \frac{P_\theta(\hat{s}_i|q_i)}{P_{\theta'}(\hat{s}_i|q_i)}, \quad (1)$$

where $r(q_i, \hat{s}_i)$ is the coefficient of importance sampling, θ and θ' are the parameters of policy model and reference model, respectively.

4 Approach

In this section, we present our proposed RLMEC, a new RL approach for improving LLMs on complex reasoning tasks. In RLMEC, we train a generative reward model to produce token-level reward scores for the sampled outputs from the policy model (*i.e.,* the LLM), then optimize the policy model via RL based on the fine-grained rewards. Figure 2 illustrates the overall framework of our RLMEC.

4.1 Generative Reward Model Training

To provide fine-grained supervision for RL, we train a generative model based on the sequence-to-sequence loss as the reward model. For a given task, the reward model aims to offer estimations for all the output tokens about their correctness. To achieve this, we design an *erroneous solution rewriting* task with the constraint of minimum editing distance to train the reward model, enabling it to focus on the key tokens that lead to the final wrong answer for punishing.

Erroneous Solution Rewriting. This task aims to correct the error tokens in the LLM generated solutions with minimum edits. Formally, given the question q, ground-truth solution s, and the generated solution $\hat{s}$, we rewrite $\hat{s}$ into a correct solution $\tilde{s}$. Specifically, we decompose it into two sub-tasks, *i.e.,* error locating and solution rewriting. For error locating, the model requires to locate the first erroneous reasoning step in $\hat{s}$, which would mislead the following steps into erroneous ones. Concretely, we split $\hat{s}$ into a sequence of reasoning steps according to the full stop or question mark: $\hat{s} = \{r_0, r_1, \ldots, r_n\}$. Then, the reward model needs to find the first undesired reasoning step r_t based on the given question and ground-truth solution:

$$RM(p_L, q, s, \hat{s}) \rightarrow r_t, \quad (2)$$

where p_L is the prompt to guide the model. Then, for solution rewriting, we leverage another prompt p_R to guide the reward model that rewrites the erroneous steps after r_t in $\hat{s}$ into the correct $\tilde{s}$:

$$RM(p_R, q, s, \hat{s}, r_t) \rightarrow \tilde{s}. \quad (3)$$

By training on the two tasks, the generative reward model would be able to rewrite erroneous solutions with the minimum editing constraint.

Distillation with Minimum Editing Constraint. To train the reward model for fulfilling the above

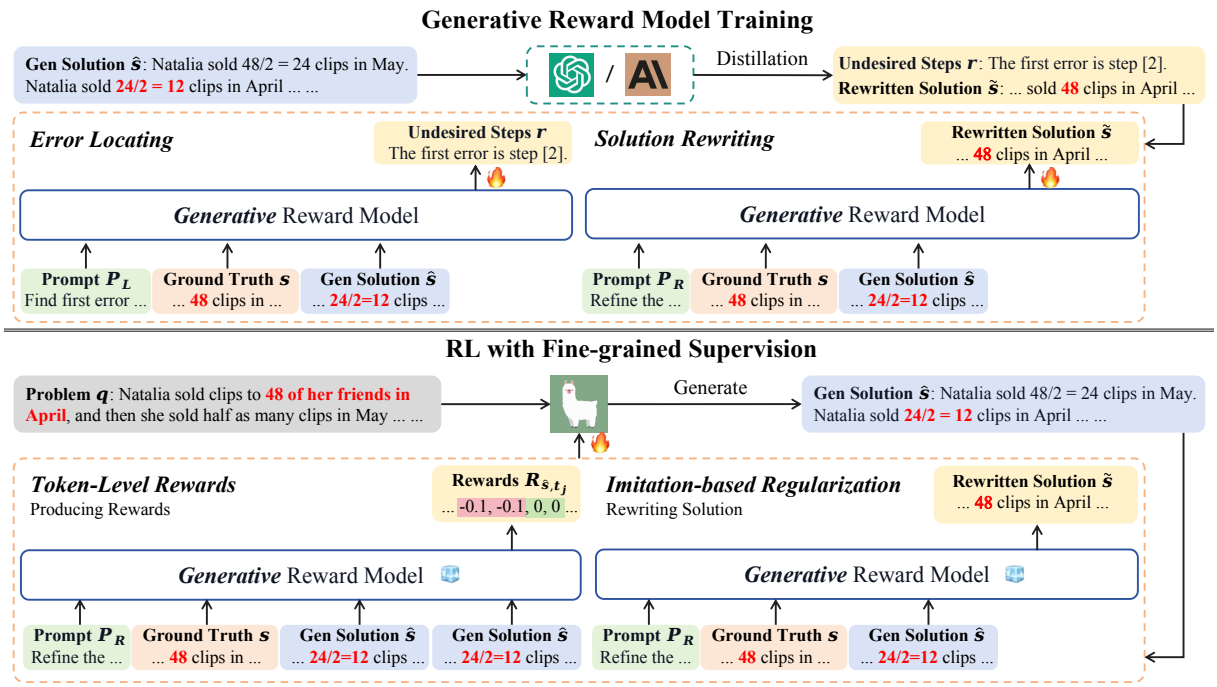


Figure 2: The overview of our RLMEC. Based on the sampled LLM solutions that contain errors, we train the generative reward model using the erroneous solution rewriting task and the distilled data with minimum editing constraint from the teacher model. Then, we perform RL training on the policy model (*i.e.,* our LLM) with fine-grained supervision using the token-level RL objective and the imitation-based regularization.

two subtasks, we collect the data from a powerful *teacher LLM* (*i.e.,* Claude 2 (Anthropic, 2023)) to distill the task knowledge for our reward model, while other models (*e.g.,* GPT-4) or human annotators can be also applied. Concretely, we first sample the generated solutions from our LLM, and select the wrong ones to compose the erroneous solution set $\{\hat{s}\}$. Then, we feed the given question q, ground-truth solution s, and the generated erroneous solution $\hat{s}$ into the teacher LLM, and add several annotated exemplars into the prompt, to guide the generation of the first error step r_t and the correct rewritten solution $\tilde{s}$. Here, in-context exemplars are human-crafted high-quality instances, and the ones for solution rewriting strictly satisfy the minimum editing constraint with only very few revised tokens. Therefore, we can obtain high-quality synthetic distilled data for the two subtasks. Finally, following Eq. (2) and Eq. (3), we prepare the inputs and outputs for the two subtasks, and merge them for training our reward model.

4.2 RL with Fine-grained Supervision

After training the generative reward model, we can leverage it to produce fine-grained supervision for the RL training of the policy model (*i.e.,* our LLM). We obtain the token-level rewards based on the generated probabilities from the reward model, and design the token-level RL objective with the imitation-based regularization for training our LLM.

Token-level Reward Generation. After distillation, the generative reward model can rewrite the original solution to provide the correct one. Owing to the minimum editing constraint, the error tokens would receive lower probabilities because they should be replaced by other tokens, and the correct tokens would obtain higher probabilities. Therefore, we can utilize the token probabilities from the reward model to assign the token-level rewards. This is quite distinct from the conventional reward model (Ouyang et al., 2022) which only produces instance-level reward scores. Concretely, given the prompt p_R, question q, ground-truth solution s, and the sampled solution $\hat{s}$ from our LLM, the token rewritten probabilities from the generative reward model are used as the reward scores for the tokens in $\hat{s}$. To better indicate the token quality, we normalize the reward scores by subtracting them from the median value of the probability (*i.e.,* 0.5) and then clip the extreme values as:

$$R_{\hat{s},t_j} = \text{CLIP}(P_{RM}(t_j|p_R, q, s, \hat{s}, t_{<j}) - 0.5, \alpha, \beta), \quad (4)$$

where $P_{RM}(t_j|p_R, q, s, \hat{s}, t_{<j})$ is the predicted probability of the correct token from the reward

model for the j-th token in $\hat{s}$, and α and β denote the minimum and maximum thresholds for the reward value. For implementation, we employ $\alpha = -0.1$ and $\beta = 0$ for the negative samples while adopt $\alpha = 0$ and $\beta = 0.5$ for the positive samples. In this way, for negative samples, the upper threshold $\beta = 0$ would lead to zero reward scores for all the non-error tokens, making the policy model only focus on punishing the error tokens. Otherwise, for positive samples, the lower threshold $\alpha = 0$ would assign zero reward score to error tokens, enabling the policy to focus on learning the correct tokens.

Token-level RL Objective. Given the token-level reward scores, we perform RL training on the policy model to correct its behaviors to avoid making errors. As mentioned in Section 3, we incorporate the PPO framework for RL, and revise its loss function to incorporate token-level reward scores. Concretely, we aim to maximize the expectation that generates the desired correct tokens in the solution. Thus, the gradients to optimize the policy model is given as:

$$\nabla \mathcal{J}_{RL}(\theta) = \sum_{i=1}^{n} \sum_{t_j \in \hat{s}_i} r(q_i, t_j) \times R_{\hat{s}_i, t_j} \times \nabla \log P_\theta(t_j | q_i, t_{<j}), \tag{5}$$

where θ is the parameters of the policy model, $P_\theta(t_j|q_i, t_{<j})$ is the predicted probability of the j-th token by the policy model, and $r(q_i, t_j)$ is the coefficient of the importance sampling in PPO as:

$$r(q_i, t_j) = \frac{P_\theta(t_j|q_i, t_{<j})}{P_{\theta'}(t_j|q_i, t_{<j})}. \tag{6}$$

Moreover, inspired by existing work (Schulman et al., 2017; Chen et al., 2019) that clips the gradients of RL, we design a simplified way that clips the coefficient of the gradient to reduce the variance of the reward and prevent the large difference between the policy and reference model:

$$\min\left(r(q_i, t_j) \times R_{\hat{s}_i, t_j}, \textsc{Clip}(r(q_i, t_j), 1-\varepsilon, 1+\varepsilon) \times R_{\hat{s}_i, j}\right), \tag{7}$$

where ε is a hyperparameter that controls the upper and lower bounds for positive and negative reward scores, respectively.

Imitation-based Regularization. As the RL training process is prone to be unstable, we further design a regularization loss based on imitation learning. The policy model is trained to imitate the generation of the rewritten solution $\tilde{s}_i$, only based on the question q_i. To compute the regularization term, we sample the generated wrong outputs $\hat{s}$ from the policy model, and utilize our generative reward model to rewrite it into a correct one $\tilde{s}$ for learning. As discussed before, the original solution $\hat{s}_i$ may contain only few error tokens that lead to the wrong solution. Therefore, we consider focusing on these error tokens in $\hat{s}$, and identify them for targeted learning. Specifically, we leverage the Levenshtein Distance algorithm (Levenshtein, 1965), an effective method to find the revised tokens in $\hat{s}$, and employ the token-level weights to emphasize them. The Levenshtein Distance algorithm utilizes dynamic programming (DP) to calculate the edit distance between $\hat{s}$ and $\tilde{s}$, and the replaced and added tokens are selected into the error token set $\mathcal{T}$. Then, the token-level weight is computed as:

Methods	NS	RL	TLS	RM
SFT (Ouyang et al., 2022)	✗	✗	✗	-
RFT (Yuan et al., 2023)	✗	✗	✗	DIS
CoH (Liu et al., 2023a)	✔	✗	✗	-
DPO (Rafailov et al., 2023)	✔	✗	✗	-
FIGA (Guo et al., 2023)	✔	✗	✔	DIS
PPO (Schulman et al., 2017)	✔	✔	✗	DIS
ToRA (Gou et al., 2023)	✔	✗	✗	-
Shep. (Wang et al., 2023b)	✔	✔	✗	DIS
WMath (Luo et al., 2023)	✔	✔	✗	DIS
RLMEC	✔	✔	✔	GEN

Table 1: The difference between RLMEC and previous related work. NS, RL, and TLS denote the usage of negative samples, reinforcement learning, and token-level supervision. RM denotes the type of the reward model. DIS and GEN denote the discriminative reward model and generative reward model, respectively.

$$w_j = \begin{cases} \gamma, & t_j \in \mathcal{T} \\ \phi \times \gamma, & t_j \notin \mathcal{T} \end{cases}, \tag{8}$$

where γ denotes the weight for emphasized tokens in $\mathcal{T}$, and ϕ is the penalty coefficient for unimportant tokens. By incorporating term-level weights, the gradients of the imitation regularization are:

$$\nabla \mathcal{L}_{IR}(\theta) = -\sum_{i=1}^{n} \sum_{t_j \in \tilde{s}_i} \nabla \log P_\theta(t_j|q_i, t_{<j}) \times w_j. \tag{9}$$

Finally, the policy model is optimized by both the RL objective and imitation-based regularization.

4.3 Summary and Discussion

Here, we present the summary of our approach and discuss its difference with existing methods.

Summary. We present the pseudo-code of RLMEC in Algorithm 1 to better demonstrate our

approach. The procedure of RLMEC can be divided into two parts, *i.e.,* generative reward model training and RL training with fine-grained supervision. For reward model training, we leverage a teacher model to synthesize the examples for the error locating and solution rewriting subtasks, to compose the dataset for distilling our generative reward model the capability of erroneous solution rewriting. Then, for RL training, we first generate the rewards for all the tokens in the sampled solutions from the LLM using Eq. 4, where we set suitable thresholds α and β to control our model to focus on key error tokens. Based on the token-level reward, we perform RL training using the PPO framework with the optimization function Eq. (5), and we design the reward clip strategy using Eq. (7) to stabilize the training process. Besides, we also add the imitation-based regularization using Eq. (9), to further help our LLM focus on learning key tokens.

Discussion. In Tabel 1, we present the difference between RLMEC and the existing work. Previous work mostly adopts the instance-level reward model, and only FIGA employs the token-level supervision but not using RL. Besides, there are several methods (*e.g.,* WizardMath, Math-Shepherd) that leverage step-level reward to perform RL. As a comparison, our proposed RLMEC enables token-level supervision in a RL framework, and thus can benefit from more fine-grained supervision and focus on punishing error tokens during training. A major novelty of our implementation is that we design the generative reward model trained by the erroneous solution rewriting task, to replace the conventional discriminative reward model, which can produce rewritten probability of each token that can be naturally used as token-level supervision. Besides, by comparing with supervised fine-tuning methods (*e.g.,* SFT and RFT), our approach can utilize the negative samples that will not be used by them, which extends the understanding of failed examples and fully utilizes the data.

5 Experiment

5.1 Experimental Settings

We simply introduce the experimental settings in this part. More details are shown in Appendix A.2.

Datasets. We employ mathematical tasks and question-answering tasks for evaluation. The specifics of each dataset are delineated in Table 2. Mathematical tasks include GSM8k (Cobbe et al., 2021), MATH (Hendrycks et al., 2021c), SVAMP (Patel et al., 2021) and the mathematical problems in MMLU (MM) (Hendrycks et al., 2021b,a). We adopt MathInstruct (Yue et al., 2023) as the training set and eliminate the code samples. Question-answering tasks contain ECQA (Aggarwal et al., 2021), QASC (Khot et al., 2020), OpenbookQA (Mihaylov et al., 2018b) and ARC-Easy (Clark et al., 2018). We merge the training set of ECQA and QASC, and adopt the mixture as the training set in the experiment.

Task	Train/Test	Dataset	Num. Data
Math	Train	MathInst	118088
	Test	GSM8k	1319
		MATH	5000
		SVAMP	1000
		MM	974
QA	Train	ECQA	7598
		QASC	8134
	Test	ECQA	2194
		QASC	926
		OBQA	500
		ARC	2376

Table 2: Statistics of the used datasets. MathInst and MM denote MathInstruct and the mathematical task in MMLU, respectively.

Baselines. For a more comprehensive assessment, we incorporate three categories of methods as baseline approaches. We conduct the SFT (Ouyang et al., 2022) and the Rejection sampling Fine-Tuning (RFT) (Liu et al., 2023b; Yuan et al., 2023) as the baseline methods of supervised fine-tuning. Besides, the representative methods of alignment without reinforcement learning, *e.g.,* DPO (Rafailov et al., 2023), CoH (Liu et al., 2023a), and FIGA (Guo et al., 2023) are conducted as the baseline. Moreover, We conduct the vanilla PPO (Schulman et al., 2017) and Actor-Critic version of PPO (PPO A2C) (Zheng et al., 2023b) as the baseline of RL methods. Additionally, we also report the performance of base LLMs, including LLaMA 2 (Touvron et al., 2023), Vicuna (Zheng et al., 2023a), and WizardLM (Xu et al., 2023).

5.2 Main Results

The evaluation results of RLMEC and the baseline methods are presented in Table 3.

First, RLMEC outperforms other baselines on the average accuracy of both scenarios. RLMEC demonstrates a strong capacity to further enhance

Methods	Question-Answering Tasks					Mathematical Tasks				
	ECQA	QASC	OBQA	ARC	Avg.	GSM8k	MATH	SVAMP	MM	Avg.
7B Parameters LLMs										
LLaMA 2	55.97	39.74	48.40	52.48	49.15	11.22	4.80	29.70	28.44	18.54
Vicuna	49.82	32.18	46.40	51.52	44.98	12.20	4.26	24.30	26.08	16.71
WizardLM	36.28	18.68	27.80	46.59	32.34	14.48	3.34	34.80	27.10	19.93
SFT LLM	71.88	55.40	52.00	56.27	58.89	51.02	10.48	47.80	38.50	36.95
+ SFT	70.65	55.94	51.60	56.99	58.80	50.34	11.04	47.20	38.40	36.75
+ RFT	72.24	58.64	55.20	57.15	60.81	49.66	10.80	48.30	39.01	36.94
+ RFT w/ GT	72.47	58.53	53.60	57.11	60.43	49.89	**11.26**	46.70	38.91	36.69
+ RFT w/ TD	73.11	58.21	54.20	57.53	60.76	**51.86**	11.04	49.40	38.19	37.62
+ RFT w/ RD	72.47	59.29	54.60	57.03	60.85	51.78	11.24	48.70	40.76	38.12
+ CoH	71.06	54.86	51.40	56.61	58.48	50.11	10.94	48.60	38.50	37.04
+ DPO	72.47	58.53	55.40	55.26	60.42	34.19	5.38	25.80	32.58	24.49
+ FIGA	69.83	52.48	51.00	46.21	54.88	-	-	-	-	-
+ Vanilla PPO	72.88	50.22	43.40	56.27	55.69	48.97	10.64	44.90	38.60	35.78
+ PPO A2C	70.83	55.08	52.40	56.02	58.58	50.94	9.38	46.60	38.50	36.36
+ RLMEC	**73.66**	**59.50**	**56.80**	**58.50**	**62.12**	51.18	11.16	**49.60**	**40.97**	**38.23**
13B Parameters LLMs										
LLaMA 2	61.53	45.46	57.90	64.31	57.30	21.23	6.58	34.40	34.39	24.15
Vicuna	50.14	39.96	48.40	53.70	48.05	24.10	4.74	33.80	29.98	23.16
WizardLM	52.60	40.93	52.30	58.96	51.20	31.01	3.18	52.00	21.36	26.89
SFT LLM	76.12	59.40	60.80	62.46	64.70	56.63	12.74	53.50	41.27	41.04
+ SFT	75.89	57.87	63.40	62.50	64.92	55.88	13.62	58.00	41.27	42.19
+ RFT	75.71	60.48	61.00	64.06	65.31	55.80	13.62	54.10	41.68	41.30
+ RFT w/ GT	76.66	60.37	63.40	63.17	65.90	57.32	13.74	56.70	43.94	42.93
+ RFT w/ TD	76.71	61.56	61.80	64.14	66.05	**58.15**	13.98	58.80	41.58	43.13
+ RFT w/ RD	76.62	62.20	63.20	63.17	66.30	57.39	**14.34**	56.20	42.81	42.96
+ CoH	76.62	60.37	59.80	63.93	65.18	57.31	13.10	54.00	42.30	41.68
+ DPO	78.26	61.45	62.20	63.80	66.43	44.20	4.38	39.70	32.14	30.11
+ FIGA	61.21	60.26	52.80	46.34	55.15	-	-	-	-	-
+ Vanilla PPO	76.34	57.99	61.80	62.29	64.61	53.45	11.76	55.10	43.12	40.86
+ RLMEC	**79.49**	**64.15**	**65.60**	**65.19**	**68.61**	**58.15**	14.00	**60.00**	**45.07**	**44.31**

Table 3: Experimental results on question answering tasks and mathematical tasks. Avg. is the average accuracy of all sub-tasks. GT, TD, and RD denote ground truth, the data generated by the teacher model, and the data generated by the generative reward model. The best are denoted in bold and the second-best are underlined.

the specific ability (*e.g.,* reasoning ability) of LLMs. With the limited training data, compared with the previous methods (*e.g.,* RFT, PPO), RLMEC leverages both positive and negative samples to provide fine-grained supervision signals, guiding LLMs to focus on the mistakes and correct them.

Second, RLMEC can prevent overfitting during domain adaption. Previous methods (*e.g.,* SFT) utilize the data from the training set or generated by LLMs to fine-tune the LLMs which might cause overfitting. We can observe that the performance decreases after SFT on the unseen tasks (*e.g.,* OBQA and SVAMP) of the 7B LLM. In contrast, the performance of LLMs on all of the unseen tasks is improved after RLMEC. The reason is that RLMEC makes LLMs focus on mistakes rather than correct components and utilize the clip mechanism to avoid overfitting.

Third, RLMEC can better leverage the generated response containing undesired components than other methods. Comparing the performance of RLMEC and DPO, we can observe that RLMEC enhance the reasoning ability of LLMs in both scenarios, but DPO only works on question-answer tasks. That is because RLMEC utilizes soft rewards to indicate positive or negative responses, while DPO collects the positive-negative response pairs to train LLMs which can be regarded as utilizing the hard labels to identify the quality of generated responses. Given the quality of generated responses is difficult to assess, it is hard to collect response pairs in the challenge tasks (*e.g.,* mathematical tasks). On mathematical tasks, the performance of DPO is even worse than the backbone LLM because of the low quality of the training data.

Finally, token-level supervision signals can further improve the performance of the policy model. The results of vanilla PPO, PPO A2C, and RLMEC present the importance of fine-grained supervision signals. Vanilla PPO utilizes instance-level signals

Methods			ECQA	ARC	GSM8k	MM
TLS	RL	IR	Acc.	Acc.	Acc.	Acc.
✔	✔	✔	79.49	65.19	58.15	45.07
✗	✔	✔	78.81	64.52	58.38	44.45
✗	✗	✔	77.85	64.18	58.56	43.84
✔	✔	✗	74.34	61.32	7.35	20.12

Table 4: The results of ablation study on 13B LLMs. TLS, RL, and IR denote token-level supervision, reinforcement learning, and imitation-based regularization.

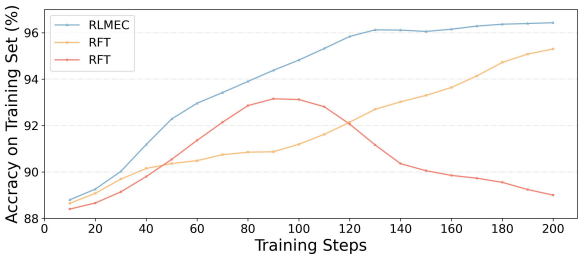


Figure 3: The performance of 7B LLMs on question-answering tasks during different training strategies. To better present the difference, we smooth out the lines.

to train the LLMs, which do not conform to reality because the generated response might contain both desired and undesired components. PPO A2C trains the critic model to provide fine-grained supervision signals which will increase the requirement of the computation resources. In RLMEC, the generative reward model is competent to implement the functionality of the reward model and the critic model in the PPO A2C at the same time.

5.3 Detailed Analysis

To further verify the effectiveness of RLMEC, we conduct the ablation study and analyze the model performance during the training process. Besides, we analyze the scaling of the generative reward model and present the case study of supervision signals and the model outputs in Appendix B and C.

Ablation Study. We evaluate the effectiveness of token-level supervision, reinforcement learning, and imitation-based regularization. Results are presented in Table 4. Given the results of the QA tasks (*i.e.,* ECQA and ARC), we can observe that removing any of the modules will hurt the performance of the LLMs. In the mathematical tasks, without token-level supervision and reinforcement learning, LLMs overfit the training set, which brings the improvement on the seen task (*i.e.,* GSM8k) and hurts the performance on the unseen task (*i.e.,* MM). The evaluation results demonstrate the ability of RLMEC to prevent overfitting and achieve the balance between seen tasks and unseen tasks. Besides, imitation-based regularization is also an important module in RLMEC. Without regularization, LLMs learn to generate correct responses only through token-level rewards. Because of the large search space, it is very difficult for LLMs to find the correct behavior in the challenge tasks. In the setting of removing imitation-based regularization, the decreasing performance on all of the tasks can verify our analysis.

Performance During Training Process. To comprehensively assess the performance of RLMEC, we conduct experiments on the accuracy of the training set during the training process. In Figure 3, we can observe that RLMEC can fit the training set more effectively and rapidly than other methods (*i.e.,* RFT and DPO). Around 120 training steps, the policy model almost fits the training set through RLMEC. That is because our methods focus on the mistakes in the generated response and guide LLMs to correct these errors, which is more efficient. In contrast, RFT optimizes the whole tokens in the correct solution which might include many unimportant tokens, and DPO is overemphasized about the negative samples. These futures will decrease the speed of optimization and hurt the performance.

6 Conclusion

In this paper, we proposed RLMEC, a new reinforcement learning framework with minimum editing constraint, to leverage fine-grained supervision signals to further improve the ability of LLMs. In our RLMEC, we first trained the generative reward model via the erroneous solution rewriting task under the minimum editing constraint, with the help of a teacher LLM. Then, we leveraged it to produce token-level rewards, and devised the token-level RL objective and an imitation-based regularization for training our LLM, which both focus on the revision of the key tokens leading to errors in the solution. Experimental results on mathematical tasks and question-answering tasks have demonstrated the effectiveness of RLMEC.

As future work, we will consider implementing our RL method on more advanced LLMs to further improve their performance on complex reasoning tasks. Besides, we will also evaluate the capacity of our approach on enhancing human alignment and reducing hallucination.

Limitations

In this section, we discuss the limitations of our work. First, in this work, we focus on the complex reasoning tasks and only conduct experiments on the QA tasks and mathematical tasks. However, RLMEC can also be employed in other scenarios, *e.g.,* human alignment and reducing hallucination, which has not been verified in this work. We leave it as the future work. Second, due to the limitation of computing resources, we only assess the performance of RLMEC on 7B and 13B LLMs, without the experiments on larger LLMs. Actually, by comparing the performance of baseline methods and RLMEC on 7B and 13B LLMs, we can observe the effectiveness of RLMEC. Third, our approach mainly focuses on enhancing LLMs on complex reasoning tasks, and does not consider the possible bias and ethic risks when using LLMs. It is also a promising direction that our RLMEC can be applied to, and we will investigate it in the future.

References

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. 2021. Explanations for commonsenseqa: New dataset and models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3050–3065.

Anthropic. 2023. Claude 2. *Anthropic Blog*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *CoRR*, abs/2309.16609.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *CoRR*, abs/2302.04023.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models. *CoRR*, abs/2308.09687.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 456–464.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *CoRR*, abs/2401.01335.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.

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. *CoRR*, 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. *CoRR*, abs/2110.14168.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of thoughts: Defying the law of penrose triangle for thought generation. *CoRR*, abs/2311.04254.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *CoRR*, abs/2305.14325.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 1251–1265.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. PAL: program-aided language models. *CoRR*, abs/2211.10435.

Google. 2023. Palm 2 technical report. *Google*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *CoRR*, abs/2309.17452.

Geyang Guo, Ranchi Zhao, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Beyond imitation: Leveraging fine-grained quality signals for alignment. *CoRR*, abs/2311.04072.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

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

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. QASC: A dataset for question answering via sentence composition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8082–8090.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.

V. I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics. Doklady*, 10:707–710.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *NeurIPS*.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. 2023a. Chain of hindsight aligns language models with feedback. *CoRR*, abs/2302.02676.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J. Liu, and Jialu Liu. 2023b. Statistical rejection sampling improves preference optimization. *CoRR*, abs/2309.06657.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pages 22631–22648.

Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. QUARK: controllable text generation with reinforced unlearning. In *NeurIPS*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *CoRR*, abs/2308.09583.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018a. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018b. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937.

OpenAI. 2023. Gpt-4 technical report. *OpenAI*.

Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2080–2094.

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. *CoRR*, abs/2305.18290.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal. 2024. A minimaximalist approach to reinforcement learning from human feedback. *CoRR*, abs/2401.04056.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275.

Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. 2023a. Making large language models better reasoners with alignment. *CoRR*, abs/2309.02144.

Peiyi Wang, Lei Li, Zhihong Shao, R.X. Xu, Damai Dai, Yifei Li, Deli Chen, Y.Wu, and Zhifang Sui. 2023b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR*, abs/2312.08935.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171.

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 *NeurIPS*.

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

Shentao Yang, Shujian Zhang, Congying Xia, Yihao Feng, Caiming Xiong, and Mingyuan Zhou. 2023. Preference-grounded token-level guidance for language model fine-tuning. *CoRR*, abs/2306.00398.

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. *CoRR*, abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *CoRR*, abs/2210.03629.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15135–15153.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *CoRR*, abs/2309.12284.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825.

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

Beichen Zhang, Kun Zhou, Xilin Wei, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023. Evaluating and improving tool-augmented computation-intensive math reasoning. *arXiv preprint arXiv:2306.02408*.

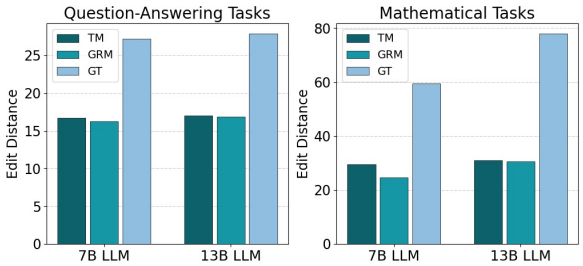
Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023a. A survey of large language models. *CoRR*, abs/2303.18223.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. 2023b. Slic-hf: Sequence likelihood calibration with human feedback. *CoRR*, abs/2305.10425.

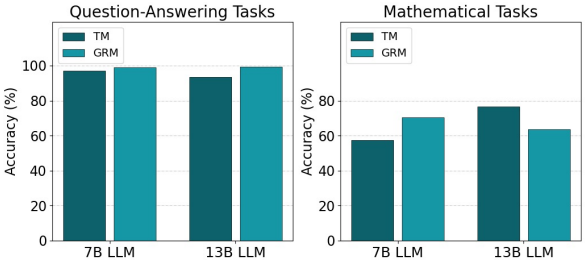
Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. 2023b. Secrets of RLHF in large language models part I: PPO. *CoRR*, abs/2307.04964.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593.



(a) Edit distance between refined responses and predictions.



(b) Accuracy of the refined response.

Figure 4: The comparison of the rewriting performance of teacher model and generative reward model. TM and GRM denote the response refined by the teacher model and the generative reward model, respectively. GT denotes the ground truth solution of the problems.

A Details for RLMEC

A.1 Prompts for Generative Reward Model Training

We present the template of the prompt for teacher model distillation, and generative reward model training and inference in Table 5 and Table 6, respectively. In practice, the information (*i.e.,* question q, Ground-Truth Solution s and Generated Erroneous Solution $\hat{s}$) should be filled into the corresponding curly brackets. For error locating task, to better guide teacher model and generative reward model to figure out the first undesired step, we utilize the index to format the ground-truth solution. The formatted solution is as follows,

[0] `The First Reasoning Step` r_0
[1] `The Second Reasoning Step` r_1
...
[n] `The Last Reasoning Step` r_n

For the generative reward model, the training instruction and inference prompt are similar. The target output of the training procedure (*i.e.,* the bold sentence in the table) will be removed during inference.

A.2 Implementation Details for Experiments

Datasets. We employ mathematical tasks and question-answering tasks for evaluation. Success-

Error Locating	Given the problem, correct solution and the prediction from language models. The method in prediction might be different with correct solution, but it is also correct. You need to identify which step of the prediction is the first wrong step, and write down the label of the first wrong step. Problem: {`Problem` q} Correct solution: {`Formatted Ground-Truth Solution` s} Prediction: {`Generated Erroneous Solution` $\hat{s}$} Which step of prediction is error? Only write down the label of the first wrong step. If the prediction is correct, you need to write down correct. You should not write down any other words.
Solution Rewriting	Given the problem and the correct solution, you need to correct the mistakes in prediction to get the correct answer. You should make minimal modifications. Problem: {`Problem` q} Correct solution: {`Generated Erroneous Solution` $\hat{s}$} Prediction: {`Generated Erroneous Solution` $\hat{s}$} Correct prediction:

Table 5: The prompt for the teacher model distillation.

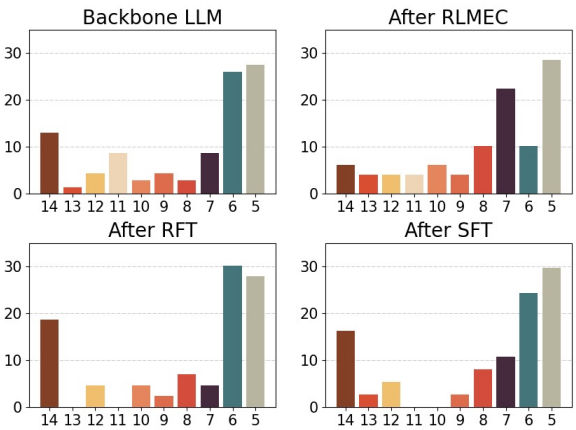


Figure 5: The position of the first error in the generated solution. The X-axis denotes how many reasoning steps between the first error and the final answer, and the Y-axis is the ratio of the corresponding problems in these problems.

fully solving these tasks necessitates LLMs to possess domain-specific knowledge and engage in systematic, step-by-step reasoning to reach the ultimate answer. The specifics of each dataset are delineated in Table 2.

• *Mathematical tasks* include GSM8k (Cobbe et al., 2021), MATH (Hendrycks et al., 2021c), SVAMP (Patel et al., 2021) and the mathematical problems in MMLU (MM) (Hendrycks et al., 2021b,a). We adopt MathInstruct (Yue et al., 2023) as the training set and eliminate the code samples. Given MathInstruct contains the training set of GSM8k and MATH, they are seen tasks for LLMs, while SVAMP and MM are unseen tasks.

• *Question-answering tasks* contain ECQA (Aggarwal et al., 2021), QASC (Khot et al., 2020), OpenbookQA (Mihaylov et al., 2018b) and ARC-Easy (Clark et al., 2018). We merge the training set of ECQA and QASC, and adopt the mixture as the training set in the experiment, Therefore, ECQA and QASC are seen tasks for LLMs, while OpenbookQA and ARC are unseen tasks for LLMs.

Baselines. For a more comprehensive assessment, we incorporate three categories of methods as baseline approaches. including supervised fine-tuning, alignment without reinforcement learning, and reinforcement learning.

• *Supervised Fine-tuning* trains LLMs to imitate the human desired behavior. We conduct the SFT (Ouyang et al., 2022) and the Rejection sampling Fine-Tuning (RFT) (Liu et al., 2023b; Yuan et al., 2023) as the baseline methods.

• *Alignment without Reinforcement Learning* is the method to align LLMs to human preference and prevent instability in reinforcement learning. Representative methods, *e.g.,* DPO (Rafailov et al., 2023), CoH (Liu et al., 2023a), and FIGA (Guo et al., 2023) are conducted as the baseline.

• *Reinforcement Learning* is the traditional method to guide LLMs to explore the world and learn from external feedback. PPO (Schulman et al., 2017) is the classical algorithm to employ reinforcement learning. We conduct the vanilla PPO (Schulman et al., 2017) and Actor-Critic ver-

Error Locating	Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: Given the problem, correct solution and the prediction from language models. The method in prediction might be different with correct solution, but it is also correct. You need to identify which step of the prediction is the first wrong step, and write down the label of the first wrong step. ### Input: Problem: {`Question` q} Correct solution: {`Formatted Ground-Truth Solution` s} Prediction: {`Generated Erroneous Solution` $\hat{s}$} ### Response: **The first error step is [{`First Undesired Reasoning Step` r_t}]**
Solution Rewriting	Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: Given the problem and the correct solution, you need to correct the mistakes in prediction to get the correct answer. You should make minimal modifications. ### Input: Problem: {`Question` q} Correct solution: {`Ground-Truth Solution` s} Prediction: {`Generated Erroneous Solution` $\hat{s}$} ### Response: Correct prediction:**{`refined solution` $\tilde{s}$}**

Table 6: The instruction for the generative reward model training. The bold sentence will be utilized to optimize the generative reward model in cross entropy loss. The prompt for inference is the same as the training instruction without the bold part.

GRM \ PM	7B PM		13B PM	
	QA	Math	QA	Math
7B GRM	62.12	38.23	66.40	43.74
13B GRM	61.32	37.46	68.61	44.31

Table 7: The comparison of the different scaling of the generative reward model. GRM and PM denote the generative reward model and policy model, respectively.

sion of PPO (PPO A2C) (Zheng et al., 2023b) in the experiment.

Moreover, we also report the performance of base LLMs, including LLaMA 2 (Touvron et al., 2023), Vicuna (Zheng et al., 2023a), and WizardLM (Xu et al., 2023).

Hyper-Parameters Setting. In the experiment, we adopt Claude 2 (Anthropic, 2023) as the teacher model. For backbone LLMs, we utilize the mixture dataset of ECQA and QASC to fine-tune LLaMA 2 (Touvron et al., 2023) to obtain the domain-adapted SFT backbone model in QA tasks, and adopt MAmmoTH (Yue et al., 2023) as the backbone model for mathematical tasks. The backbone LLMs of the policy model and the generative reward model are the same SFT LLMs. In the training procedure, we employ 5×10^{-6} as the learning rate for all tasks and train LLMs for 1 epoch. Besides, we set 128 and 768 as the batch size for QA tasks and mathematical tasks. For the value of ε, we leverage 0.3 and 0.4 for 7B model and 13B model, respectively. Because the LLMs have adapted to the corresponding domain after training, we adopt the 0-shot setting during evaluation.

B Performance Analysis of RLMEC

B.1 Analysis of Generative Reward Model.

The effectiveness of the generative reward model will influence the quality of the token-level rewards and the refined response. Thus, we present the comparison of the teacher model and the generative reward model on QA tasks in Figure 4. We can observe that both the teacher model and the generative reward model can significantly reduce the edit distance and even perform slightly better than the teacher model. That is because we utilize the two-stage prompting strategy to distillate knowledge from the teacher model and conduct the high-quality data to fine-tune the generative reward model. Through fine-tuning, it can adapt to the erroneous solution rewriting task well. Moreover, the teacher model and the generative reward model have shown similar performance on the accuracy of the refined responses, which verifies that the rewriting task can be easily learned by the LLMs

Algorithm 1: The RLMEC algorithm.

Input : Training set $\mathcal{D} = \{\langle q_i, s_i \rangle\}_{i=1}^{n}$, the teacher model (Claude 2), and the parameters of SFT model θ_{SFT}.
Output : A well trained policy model.

Initialize the parameters of the generative reward model: $\theta_{GRM} \leftarrow \theta_{SFT}$;
Initialize the parameters of the policy model: $\theta \leftarrow \theta_{SFT}$;

// Generative Reward Model Training
for *each instance* $\langle q_i, s_i \rangle$ *in* $\mathcal{D}$ **do**
  The policy model generates $\hat{s}_i$ based on q_i;
  if *the data is sampled* **then**
    The teacher model locals the first error at r_i using Eq. 2;
    The teacher model rewrites $\hat{s}_i$ to obtain $\tilde{s}_i$ using Eq. 3;

Use q_i, s_i, $\hat{s}_i$, r_i, and $\tilde{s}_i$ to construct $\mathcal{D}'$;
Leverage $\mathcal{D}'$ to supervised-finetune the generative reward model through Seq2Seq training paradigm;

// RL with Fine-grained Supervision
for *each instance* $\langle q_i, s_i, \hat{s}_i, r_i, \tilde{s}_i \rangle$ *in* $\mathcal{D}'$ **do**
  Generate the token-level rewards using Eq. 4;
  Compute the reward $\mathcal{J}_{RL}(\theta)$ using Eq. 5;
  Use Levenshtein Distance algorithm to compute the token-level weight;
  Compute the loss of imitation-based regularization $\mathcal{L}_{IR}(\theta)$;
  Update θ through $\mathcal{J}_{RL}(\theta)$ and $\mathcal{L}_{IR}(\theta)$;

with smaller parameters. Besides, given the performance of RFT w/ TD and RFT w/ RD in Table 3, we can observe that the higher accuracy of the refined responses will lead to higher performance in downstream tasks through simply supervised fine-tuning.

B.2 Scaling Analysis of Reward Model.

To explore the influence of the scale of the generative reward model, we conduct the experiment and present the results in Table 7. For both 7B and 13B LLMs, the rewriting model trained from the same backbone LLMs with the policy model performs better. The potential reason might be that the policy model and the rewriting model with the same backbone model will have a similar distribution. In this situation, the rewriting model can provide appropriate supervision signals and better guide the training process.

B.3 Position of the First Error

We conduct experiments about the position of the first error in the generated response after training. The results are shown in Figure 5, respectively. Rhe experiment on the position of the first error can verify the effectiveness of RLMEC. Compared with the backbone LLMs, the first error appears later after RLMEC. For example, after RLMEC, the number of problems where the first error occurs before the final answer 7 steps (*i.e.,* the third column on the right) has increased, while the number of problems where the first error occurs more than 7 steps has decreased. The reason is that LLMs focus on the mistakes and learn to correct the early errors during RLMEC. In the ideal situation, all of the mistakes will be corrected through further training. In contrast, after training through other methods, the position of the first error is irregular, which means that these methods do not consider the mistakes in the generated response and guide LLMs to learn to generate the correct solution without purposiveness.

C Case Study

C.1 Analysis of the Supervision Signals

We present the case study about the reward from different methods in Table 8. To better express the difference, we do not employ the clip mechanism in the case study. From the results, we can observe that the reasoning step of the generated solution is correct but the final answer is error. In PPO A2C, the reward will be calculated by the reward model and the critic model. The tokens generated earlier will receive a lower reward, which is contradictory to reality. That is because PPO A2C has assumed that the previous token will influence the last token. In this case, once the generated solution contains the wrong answer, the rewards of the previous tokens are likely lower than the last tokens. In contrast, we leverage the generative reward model to generate the reward in RLMEC. The reward of the current token is calculated based on the previous tokens. Therefore, the rewriting model in RLMEC can better indicate whether the token is correct and provide high-quality token-level supervision signals. Besides, for the outcome-supervised method (*i.e.,* Vanilla PPO), the reward of each token is equivalent and is based on whether the generated solution is correct. This method cannot describe the correctness of the tokens in the generated response.

C.2 Analysis of the Generated Responses

To further demonstrate the effectiveness of RLMEC, we present the case study about the performance of the LLMs trained by different methods in Table 9 and Table 10. In both cases, our proposed RLMEC can help the LLMs to focus on the previ-

Problem	Look at this series: 1.25, 7.5, ____, 270, 1620, ... What number should fill the blank? Answer Choices: (A) 50 (B) 65 (C) 45 (D) 55 (E) 40
Ground Truth Solution	Let's think about the multi-choice question. This is a simple multiplication series. Each number is 6 times greater than the previous number. The answer is C
Generated Solution	Let's think about the multi-choice question step by step. This is a simple multiplication series. Each number is 6 times greater than the previous number. The answer is B
Rewards from RLMEC	Let's think about the multi-choice question step by step . This is a simple multiplication series. Each number is 6 times greater than the previous number . The answer is B
Rewards from PPO A2C	Let's think about the multi-choice question step by step . This is a simple multiplication series . Each number is 6 times greater than the previous number . The answer is B
Rewards from Vanilla PPO	Let's think about the multi-choice question step by step . This is a simple multiplication series .Each number is 6 times greater than the previous number . The answer is B

Table 8: The comparison of the reward of the generated solution from different methods. We use different background colors to indicate the reward. The color changing from red to green denotes the reward changing from negative to positive .

ous errors and correct the errors in the next time generation. Concretely, in the question-answering tasks, the keywords of the problem are "even if they get it". After being trained through RLMEC, the LLMs can understand the meaning of the problem, figure the key point, and reach the correct answer. However, through other methods, the LLM is still unable to grasp the key works in the problem and generate the answer about the emotion of losing the job. Moreover, for mathematical problem, the LLM have made the mistake in calculating "12−15". The LLM trained by baseline methods still make similar mistakes. This case has shown that it is difficult for the previous methods to generate the supervised signals which can directly indicate the mistakes in the generated content and guide the LLMs to correct the errors. In contrast, RLMEC leverages the generative reward model to provide the token-level supervised signals and guide the LLMs to focus on the mistakes. Therefore, through RLMEC, the LLMs can correct the previous errors and obtain the correct answer.

Problem	Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: Applying for a job can make someone feel what sort of emotion, even if they get it? Options: working hard frustration rejection defeat stress ### Response: Let's think step by step.
Ground-Truth Solution	Stress is a feeling of tension about uncertainty. Applying for a job has an uncertain outcome. The answer is stress
Generated Solution Before Training	Applying for a job is followed by attending an interview. Attending an interview gives rejection or acceptance. Rejection is an emotion. The answer is rejection
Generated Solution After RLMEC	Applying for a job can make someone feel stress even if they get it. Stress is an emotion. The answer is stress
Generated Solution After RFT	Applying for a job is followed by attending an interview. Attending an interview gives rejection or acceptance. Rejection is an emotion. The answer is rejection
Generated Solution After DPO	Frustration is the feeling of being upset or annoyed as a result of being unable to change or achieve something. Applying for a job can make someone feel frustration even if they get it. The answer is frustration
Generated Solution After PPO A2C	Applying for a job is followed by attending an interview. Attending an interview gives an opportunity to sell one's skills and abilities to the employer. Employer decides the suitability of the candidate based on his skills and abilities. If the employer rejects the candidate, he feels frustrated. The answer is frustration

Table 9: The case study for question-answering tasks.

Problem	Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: A waiter had 12 customers. While 15 customers left he got 14 new customers. How many customers does he still have? ### Response: Let's think step by step.
Generated Solution Before Training	The waiter has 12-15=3 customers left. He got 14+3=17 customers. The answer is 17
Generated Solution After RLMEC	The waiter had 12 customers + 14 customers = 26 customers. But since 15 customers left, he now has 26 customers - 15 customers = 11 customers. The answer is 11
Generated Solution After RFT	The waiter has 12-15=3 customers left. He has 14+3=17 customers now. The answer is 17
Generated Solution After PPO A2C	The waiter has 12-15=3 customers left. He got 14+3=17 customers. The answer is 17

Table 10: The case study for mathematical tasks.