Improving Language Model Reasoning with Self-motivated Learning

Yunlong Feng[†], **Yang Xu**[†], **Libo Qin**[†], **Yasheng Wang**[‡], **Wanxiang Che**[†*]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{ylfeng, yxu, lbqin, car}@ir.hit.edu.cn
[‡]Huawei Noah's Ark Lab
yashengwang@huawei.com

Abstract

Large-scale high-quality training data is important for improving the performance of models. After trained with data that has rationales (reasoning steps), models gain reasoning capability. However, the dataset with high-quality rationales is relatively scarce due to the high annotation cost. To address this issue, we propose *Self-motivated Learning* framework. The framework motivates the model itself to automatically generate rationales on existing datasets. Based on the inherent rank from correctness across multiple rationales, the model learns to generate better rationales, leading to higher reasoning capability. Specifically, we train a reward model with the rank to evaluate the quality of rationales, and improve the performance of reasoning through reinforcement learning. Experiment results of Llama2 7B on multiple reasoning datasets show that our method significantly improves the reasoning ability of models, even outperforming text-davinci-002 in some datasets.

Keywords: Reasoning, Chain of Thought, Reinforcement Learning

1. Introduction

Large Language Models (LLMs) that are pretrained on extensive text corpora have exhibited profound capability across a diverse array of downstream tasks. Particularly, their adaptability in both few-shot and zero-shot learning contexts, achieved by assimilating task-specific instructions and demonstrations, has garnered significant attention (Raffel et al., 2020; Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2022; Lampinen et al., 2022; Gu et al., 2022; Ye et al., 2023). This approach emphasizes generating a series of intermediate reasoning steps, which can be achieved through CoT demonstrations in prompts (Wei et al., 2022) or by guiding models with instructions in zero-shot scenarios (Kojima et al., 2022).

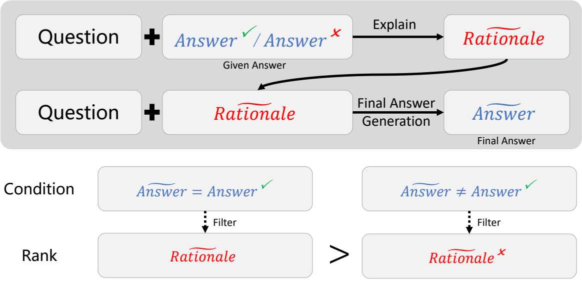


Figure 1: The motivation of our method. We note that the $\widetilde{Rationale}$ and $\widetilde{Answer}$ means they are generated by the language model. The main idea is that: (1) The correct given answer more likely leads to correct rationale. (2) The rationale that leads to the correct answer is better than the rationale that leads to the wrong answer.

Some studies have demonstrated that large-scale, high-quality data is crucial for enhancing the reasoning abilities of models (Kim et al., 2023; Ho et al., 2023; Geva et al., 2021a; Cobbe et al., 2021). But there is a scarcity of datasets with reasoning steps due to the high annotation cost. On one hand, series of works resort to manual annotations for datasets (Lu et al., 2022; Xie et al., 2020; Mihaylov et al., 2018; Khot et al., 2020). Training models with data obtained in this manner can significantly enhance their performance, albeit at a substantial cost. On the other hand, some studies generate data using large-scale models (Ho et al., 2023; Kim et al., 2023; Luo et al., 2023; Liu et al., 2023; Wang et al., 2023; Li et al., 2023a), utilizing such data to train models to improve their performance. Both manual annotation and generating data using large models incur considerable costs. Beyond these methods, some strategies involve models generating rationales and filtered by answers, subsequently using this data for finetuning (Zelikman et al., 2022). In contrast, we propose a method that motivates the model itself to generate rationales of varying quality with existing datasets, integrating this preference into reinforcement learning to improve model performance.

We ask the question *Can we leverage the intrinsic properties of model-generated data to address the issue of data scarcity?* We observe that there exists an inherent preference here, that a rationale capable of generating the correct answer should be superior to a rationale that generates an incorrect one. The implicit information in this statement is that proper reasoning should lead to correct re-

[*] Corresponding author.

LREC-COLING 2024, pages 8840–8852
20-25 May, 2024. © 2024 ELRA Language Resource Association: CC BY-NC 4.0

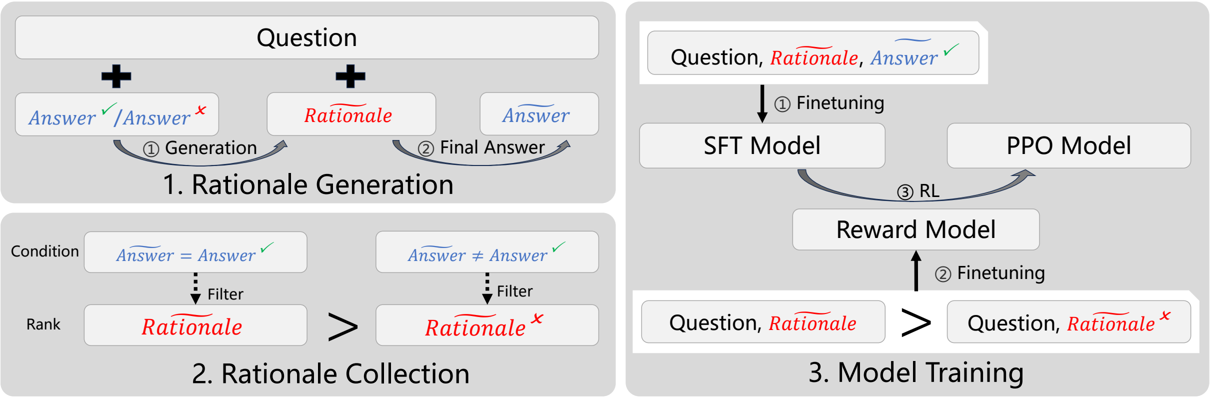


Figure 2: Overview of our method. As shown in the figure, our method can be divided into three steps in general: (1) **Rationale Generation**: We first generate a rationale using *Few-shot-CoT* (Kojima et al., 2022). Specifically, we first generate rationales for different answers (called *Given Answer*). We then use these generated rationales to generate the *Final Answer*. This allows us to later filter the rationales using both the original and generated answers. (2) **Rationale Collection**: We filter the rationales by determining whether the *Given Answer* and the *Final Answer* match the correct answer. This helps us identify relatively better and worse rationales. (3) **Model Traning**: We use the better data to train a model and get a *Supervised Fine-tuning Model (SFT Model)*. We then use data of varying quality to build a *Reward Model (RM)*. Finally, we utilize the previously acquired *SFT Model* and *Reward Model* for reinforcement learning with PPO.

sults, while improper reasoning should yield incorrect ones. To illustrate, consider the question "Sam had 9 dimes in his bank. His father gave him 7 more. How many dimes does Sam possess now?" The rationale "Sam has 9 + 7 = 16 dimes." results in the correct answer "16", while the rationale "Sam had 9 - 7 = 2 dimes remain." produces an incorrect answer "2". It is evident that the former rationale should be better than the latter.

This preference information can be used to filter or evaluate the quality of rationale. Specifically, we can use a model to generate a large number of rationales, and then utilize this preference to filter them, obtaining relatively high-quality and low-quality rationales. Subsequently, we can use this preference information to train a reward model that assesses the quality of rationales generated by the model. By integrating current reinforcement learning algorithms, this reward model can be used to optimize the model, enabling it to discern which rationales are superior, thereby enhancing its performance. Through this approach, we can leverage existing datasets to generate rationales and construct a rank, solely relying on the model itself, without the need for extrenal large models or manual annotations. We call it Self-motivated Learning.

In summary, our contributions are as follows:

- We point out an inherent preference in rationales, that is, a *rationale* capable of generating correct answers should be superior to a *rationale* generating incorrect answers. This preference reflects the quality of the *rationale*.
- By using this preference, we alleviate data scarcity. We utilize the model and existing datasets to generate *rationale*, integrating this preference into reinforcement learning to improve model performance.
- We conducted experiments using *Llama 2 7B* and multiple datasets. The results demonstrate that our method can significantly improve the performance of the model. Without resorting to large models, our approach surpasses the performance of models fine-tuned with *rationales* generated by *text-davinci-002*. In some tasks, it even outperforms *text-davinci-002*.

Rationale Generation	Final Answer Generation
[Instruction and Question]	[Instruction and Question]
$<q_i>$	$<q_i>$
[Answer]	[Rationale]
$<a'>$	$<\hat{r}_i>$
[Rationale]	[Answer]
$<\hat{r}_i>$	$<\hat{a}_i>$

Table 1: Prompt templates for generation. We first use the question $<q_i>$ and given answer $<a'>$ to generate the rationale $<\hat{r}_i>$, and then use the question $<q_i>$ and generated rationale $<\hat{r}_i>$ to generate the final answer $<\hat{a}_i>$.

2. Method

We propose Self-motivated Learning, a task-agnostic approach to further improve the performance of reasoning in LMs. The core idea is to generate correct/incorrect rationales with in-context learning, and then use them to do fine-tuning and

Prompt For Better Rationale	Prompt For Worse Rationale
[Instruction and Question] Would a pear sink in water? [Answer] No [Rationale] *The density of a pear is about 0.6g/cm3, which is less than water. Objects less dense than water float. Thus, a pear would float.*	[Instruction and Question] Would a pear sink in water? [Answer] Yes [Rationale] *The density of a pear is about 0.6g/cm3, which is less than water. Objects less dense than water float. Thus, a pear would sink.*

Table 2: Example of the prompt for better and worse rationale. The correct answer of the question is "No". We use the correct answer and incorrect answer to generate the *rationales*.

reinforcement learning. To filter the generated rationales, we use the model to generate the answer based on the generated rationales and compare it with the given answer and ground truth. After filtering, we can get relatively high-quality and low-quality rationales to train the models.

2.1. Training Process

Step 1. Rationale Generation. As table 1 shows, we first utilize a model to generate rationales for a given task $\mathcal{T}$. Consider a standard sample S_i consisting of a question q_i and its true answer a_i. Using Few-shot-CoT (Kojima et al., 2022), we prompt the model to generate a rationale $\hat{r}_i$ based on the given answer a_i', where a_i' is the correct answer or incorrect answer. Then we utilize a model to generate a final answer $\hat{a}_i$ for $\hat{r}_i$ with greedy decoding. We can construct correct answers or incorrect answers from the source dataset. For example, in the question "Would a pear sink in water?", the correct answer is "No". When generating a rationale, we place the given answer before the rationale, as table 2 shown. Besides the correct answer provided by the dataset, we can generate incorrect answers based on the given options or randomly. In this way, we can produce rationales for both the correct and incorrect answers. This provides some information for our subsequent rationale selection.

Step 2: Rationale Collection. Once we've generated the CoT rationales and the final answers, our next step is to filter these rationales based on their quality. Our objective is to distinguish between high-quality and low-quality rationales. We employ the following filtering criteria:

- **Answer Consistency Check:** Evaluate the correctness of the provided answer a_i' and the final answer $\hat{a}_i$ by comparing them with the true answer a_i. When both a_i' and $\hat{a}_i$ are correct, we categorize the corresponding rationale as a high-quality rationale. Conversely, if both are incorrect, the rationale is deemed low-quality. We throw away the rationales that do not fall into either of these categories.
- **Rationale Content Check:** We filter rationales that include the correct answer but exclude incorrect answers as high-quality rationales. In contrast, we discard the rationales that do not contain the correct answer as low-quality rationales.
- **Label Reference Check:** When dealing with multiple-choice questions, the given rationale should reference the label of the chosen answer. So, if "C. Paris" is the selected option, the word "Paris" should be incorporated in the rationale content.
- **Numerical Accuracy Check:** For numerical solutions, the answers are transformed into a floating-point format for consistency. If the absolute difference between two answers is less than $1e-6$, they are treated as identical. Moreover, the answer should be present within the rationale.

Step 3. Model Training. After the generation and filtration processes in the initial two steps, we have obtained rationales of both relatively high and low quality. Then we can train the models with them.

1. **Supervised Fine-Tuning Model (SFT Model):** After collecting the rationale data, we fine-tune the model in the assembled high-quality rationales to get a SFT model $\mathcal{M}_{\text{sft}}$. The training objective employed for this fine-tuning remains consistent with the pre-training phase, specifically utilizing the autoregressive language modeling objective or next-token prediction (Radford et al., 2018). Mathematically, the objective is to minimize the language modeling loss below:

$$\mathcal{L} = -\sum_{t=1}^{T} \log p(x_t | x_1, \ldots, x_{t-1}; \theta) \quad (1)$$

   To be noticed, we use the format "question, rationale, answer" for training, and only calculate the loss of *rationale* and *answer*.
2. **Reward Model (RM):** To train the Reward Model $\mathcal{M}rm$, we utilize both high-quality and low-quality rationales from the same question. The training objective for the reward model is captured by the following loss function:

$$\mathcal{L} = -E_{(x,y_j,y_k)\in D}[\log(\sigma(r_\theta(x,y_j) - r_\theta(x,y_k)))] \quad (2)$$

   In this function, r represents the model's score, and y_j is the preferred choice. The equation ensures that the RM assigns a higher score to the high-quality rationale y_j compared to the low-quality one y_k. The second term of the loss acts as a regularizer, penalizing extreme values of the scores.
3. **Reinforcement Learning:** Finally, we employ the fine-tuned model $\mathcal{M}_{sft}$, and the reward

Dataset	Choices	Training Samples	Test Samples	Data Split	License	References
SingleEq	-	356	152	70:30	None	Koncel-Kedziorski et al. (2015a)
AddSub	-	276	119	70:30	Unspecified	Hosseini et al. (2014)
MultiArith	-	420	180	70:30	Unspecified	Roy and Roth (2016)
SVAMP	-	700	300	70:30	MIT	Patel et al. (2021a)
GSM8K	-	7473	1319	Original	MIT	Cobbe et al. (2021)
Date Understanding	5–6	258	111	70:30	Apache-2.0	Srivastava et al. (2022)
CommonSenseQA	5	9741	1221	Original	Unspecified	Talmor et al. (2018)
StrategyQA	2	1603	687	70:30	Apache2.0	Geva et al. (2021a)

Table 3: Description of datasets used in our study.

model $\mathcal{M}_{rm}$ to perform reinforcement learning in the training dataset utilizing the PPO algorithm. The SFT Model serves as a backbone, guiding the initial stages of the learning, while the RM provides the necessary feedback for refining the policy. A common issue with training the language model with RL is that the model can learn to exploit the reward model by generating complete gibberish, which causes the reward model to assign high rewards. To balance this, we add a penalty to the reward: we keep a reference of the model that we don't train and compare the new model's generation to the reference one by computing the KL-divergence:

$$R(x, y) = r(x, y) - \beta KL(x, y) \tag{3}$$

where r is the reward from the reward model and $KL(x, y)$ is the KL-divergence between the current policy and the reference model.

2.2. Strategy of Reward

We already have a rank preference and need to design a reward strategy. We have devised three strategies to investigate the impact of the reward strategy on the model. Next, we introduce these three RL reward strategies.

- **Simple RL:** During the training process, we predefined the output format, allowing us to extract the model's final answer from its output. As we train in the dataset, we can compare the model's output with the correct answer. If the output matches the correct answer, we confer a positive reward score for the output; otherwise, a negative reward score is given. This represents the simplest scenario based on our ranking preference.
- **Model RL:** In Simple RL, we directly compare the model's output with the correct answer. However, this approach solely discerns correct from incorrect outputs without assessing the quality of rationales that are both correct or incorrect. To address this limitation, we propose training a reward model using rationales generated from the training set, both correct and incorrect. This empowers the model to implicitly discern the quality of a rationale.
- **Correction RL:** We integrate the approaches of both Simple RL and the Reward Model. If the model's predicted Final Answer is incorrect, we confer a negative reward score. However, if the answer is correct, we compare the results of Simple RL with the Reward Model and allocate the greater one from the two methods. In this way, we can avoid some errors in the Reward Model in some contexts.

3. Experiments

3.1. Tasks and datasets

Following the split of Ho et al. (2023), we evaluate our method in 8 datasets pertaining to three categories of complex reasoning, which are shown in table 3. These include SingleEq (Koncel-Kedziorski et al., 2015a), AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2016), SVAMP (Patel et al., 2021a), GSM8K (Cobbe et al., 2021), Date Understanding (Srivastava et al., 2022), CommonSenseQA (Talmor et al., 2018) and StrategyQA (Geva et al., 2021a).

3.2. Comparison methods

We present a comparison of our methods alongside several baseline methods.

Open/Close-Source Models : We prompt the open-source models and the close-source models to generate the rationales and final answers.

- **Open-Source Models**: It takes the instruction-tuning format of StableVicuna and LLama2-Chat with the final answer generation.
- **Close-Source Models**: It takes the Zero-shot-CoT format following Kojima et al. (2022): "Q: <$\hat{q}_i$>. A: Let's think step by step. <$\hat{r}^i$> Therefore, the answer is <$\hat{a}_i$>".

Methods on Llama2 7b : We compare our method with the following methods on Llama2 7B.

- **Few-shot-CoT**: This method employs few-shot prompting, as outlined in (Wei et al., 2022). And the Few-shot-CoT$^{SC=8}$ means the self-consistency and the number of samples is 8.

Method	Param	Single Eq	Add Sub	Multi Arith	SVAMP	GSM8K	Date Understanding	Common SenseQA	Strategy QA
Close-Source Models									
text-davinci-003	175B	86.4	81.3	83.7	73.6	59.5	77.0	70.0	61.1
text-davinci-002	175B	82.24	78.99	78.89	64.67	40.26	73.87	61.75	53.57
Open-Source Models									
StableVicuna	13B	62.50	57.14	43.33	46.67	40.26	45.95	58.64	41.34
LLama2-Chat	7B	73.03	68.91	67.22	53.67	28.35	35.14	56.67	38.14
Methods on Llama2 7B									
Few-shot-CoT	7B	63.82	54.62	35.00	39.00	14.60	53.15	50.61	61.28
Few-shot-CoT$^{SC=8}$	7B	67.76	67.23	55.56	44.67	15.09	35.13	48.40	62.45
Fine-tune	7B	71.05	63.87	11.67	45.67	12.58	64.87	76.58	65.21
Fine-tune-CoT (text-davinci-002)	7B	70.39	72.27	76.67	47.33	–	73.88	–	58.95
Fine-tune-CoT (STaR)	7B	75.66	67.23	72.78	44.33	17.29	81.98	63.63	64.63
Fine-tune-CoT (Llama2)	7B	71.05	65.55	53.33	40.67	13.72	83.78	69.53	60.84
Self-motivated Learning	7B	**76.32**	**76.47**	**80.00**	**55.33**	**18.88**	**87.39**	**77.97**	**66.08**

Table 4: Accuracy (%) in 8 tasks under our different models and methods. Note that the methods based on the LLama2 7B are trained in different datasets seperately.

- **Fine-tune**: The model is fine-tuned in the training dataset without any rationales.
- **Fine-tune-CoT**: This model is fine-tuned with rationales generated by different methods.
  - **Fine-tune-CoT (text-davinci-002)**: This model is fine-tuned with diverse reasoning data generated by text-davinci-002 from Ho et al. (2023). Due to limited training resources, Ho et al. (2023) did not generate diverse reasoning data for all datasets (e.g. GSM8K, CommonsenseQA).
  - **Fine-tune-CoT (STaR)**: This model is fine-tuned with the rationales generated following STaR (Zelikman et al., 2022).
  - **Fine-tune-CoT (Llama2)**: This model is fine-tuned with the filtered rationales generated with Few-shot-CoT by Llama2 7B.
- **Self-motivated Learning (Ours)**: The method, which implements reinforcement learning with PPO for the "Fine-tune-CoT (Llama2)" model, is described in Section 2.

Due to certain policy restrictions, we cannot access OpenAI's API. Consequently, we trained the Fine-tune-CoT (text-davinci-002) model using data from Ho et al. (2023). Additionally, the text-davinci-002 performance results are sourced from Ho et al. (2023).

3.3. Experiments Setting

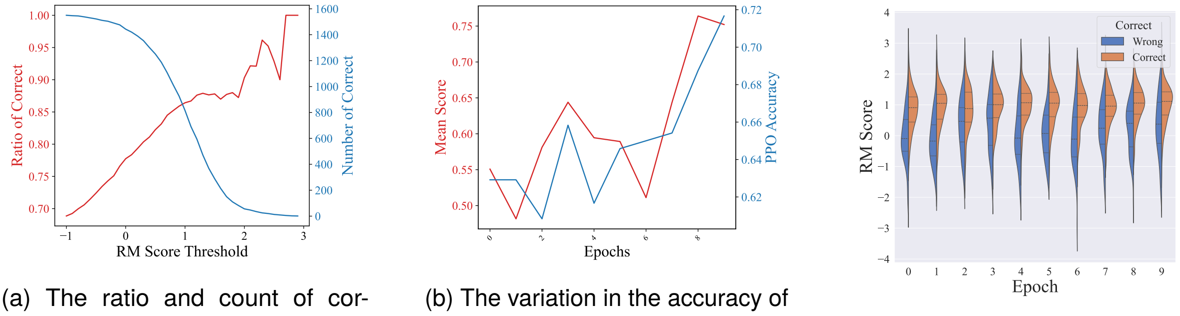
Implementation details. All experiments were conducted using the Llama2 7B model (Touvron et al., 2023) with Lora (Hu et al., 2021). We employ Lora to train the SFT model and Reward Model under half-precision to obtain their Lora weights W_{SFT} and W_{RM}. Subsequently, we use the weight W_{SFT} to initialize the Lora weight of the Policy. In Reinforcement Learning, we utilize PPO to optimize the Policy's LoRA weights W_{Policy}, only requiring switching between W_{RM}, W_{Policy}, and W_{SFT} during the training process. It is important to note that we did not merge the Lora weight W_{SFT} with the original model during the training, ensuring that the resulting Policy weight is relatively small, which aids in memory conservation. By adopting this approach, we can significantly save GPU memory and storage space. The format of the training data for SFT will be formatted like the "final answer generation" task shown in table 1.

Generation. Following Wei et al. (2022); Kojima et al. (2022); Ho et al. (2023), we employ greedy decoding to evaluate performance. For *rationale generation*, we apply Few-shot-CoT with temperature sampling configured with parameters: $T = 0.8$, $TopP = 0.95$, and $Max\ Length = 512$, and then use greedy decoding for *final answer generation*. To optimize memory usage, we incorporate temperature sampling, but adjust the parameters to $TopP = 1.0$ and $Max\ Length = 150$ during the reinforcement learning process. Detailed templates for *rationale generation* and *final answer generation* are shown in table 1.

Rationale Data. Due to resource limitation, Ho et al. (2023) did not produce diverse reasoning for CommonsenseQA. For analogous reasons, we generated merely 5 instances of diverse reasoning for CommonsenseQA. For mathematical problems, we randomly generated an incorrect answer between 0 and 100. For other datasets, we generate 8 instances of diverse reasoning and 2 instances of rationale for each incorrect answer and limit the rationales from text-davinci-002 to 8 instances.

3.4. Results

In this section, we present the results of our experiments. We first present the results of the methods and then analyze the results of our method.



(a) The ratio and count of correct reasoning vary with the reward score threshold.

(b) The variation in the accuracy of the PPO/RM model and the average score.

(c) The score distribution of the RM model in every epoch during PPO.

Figure 3: The analysis in the SingleEq dataset. The "RM Score Threshold" is what we use to filter the rationales whose score is greater than the threshold during PPO. The "PPO Accuracy" means the accuracy in the training dataset during PPO.

Method	SingleEq	AddSub	MultiArith
Fine-tune-CoT (text-davinci-002)	70.39	72.27	76.67
+RL	71.71	78.16	80.56
Increase	+1.32	+5.89	+3.89

Table 5: The performance comparison of Fine-tune-CoT (text-davinci-002) before and after reinforcement learning (RL) implementation. The reward model used in RL is constructed based on data generated by Llama2 7B. It can be observed that the reward model trained with Llama2 7B can be effectively transferred to Fine-tune-CoT (text-davinci-002) to enhance its performance.

Reasoning generation with small models is not bad. We employ Llama2 7B combined with Few-shot-CoT to generate rationales. After fine-tuning Llama2 7B with filtered rationales, the performance of Fine-tune-CoT surpasses the Fine-tune-CoT (text-davinci-002) in some datasets. For example, the improvements in SingleEq, Date Understanding, and StrategyQA are 0.66%, 9.90%, and 1.69%, respectively. This indicates that small models, when filtered, can provide valuable reasoning data. Furthermore, generating reasoning with smaller models is cost-effective.

Fine-tune-CoT's effectiveness is limited In commonsense reasoning. The gains from Fine-tune-CoT in commonsense reasoning tasks are relatively modest. It can be observed that whether using CoT generated from Llama2 or InstructionGPT, the enhancements introduced by Fine-tune-CoT are limited. Both show a performance drop compared to the direct use of Fine-tuning. In our experiments, the performance decreased by 7% to 15% compared to direct fine-tuning.

Reinforcement learning can significantly improve performance. The Self-motivated Learning applies PPO for reinforcement learning in the training dataset using the Fine-tune-CoT model, resulting in an average increase of 10.68%. This demonstrates its superior performance in various tasks, with an average accuracy of 74.22%. Notably, our method surpasses the performance of the model fine-tuned with CoT generated by text-davinci-002 in all datasets. Specifically, in MultiArith, Date Understanding, CommonSenseQA, and StrategyQA, our approach outperforms text-davinci-002.

	State		Single	Add	Multi	SVAMP	Date	Common	Strategy
FS	SFT	RL	Eq	Sub	Arith		Understanding	SenseQA	QA
✓	✓	✓	83	5	43	75	46	511	261
✓	✗	✓	10	59	13	19	7	46	45
✗	✓	✓	19	3	46	28	40	279	101
✗	✗	✓	4	25	42	44	4	116	47
✓	✓	✗	2	0	5	5	3	22	39
✓	✗	✗	2	1	2	18	3	39	76
✗	✓	✗	4	0	2	14	4	37	17
✗	✗	✗	28	26	27	97	4	171	101

Table 6: We present the fluctuation in the number of samples of different states using three methods on the test set: Few-shot (FS), Supervised Fine-tuning (SFT), and Reinforcement Learning (RL). For instance, the second sequence "✓ ✗ ✓" indicates that the sample is correct under FS and RL, but incorrect under SFT. We can see that RL can rectify the errors introduced during SFT in the different datasets.

3.5. Analysis

In this section, we conduct an analysis of our method using the SingleEq dataset. Our primary focus includes examining the relationship between reward scores and the quality of rationales, the correlation between given answers and rationales, the transferability of the reward model, and the impact of reinforcement learning.

Method	Param	Teacher Param	Single Eq	Add Sub	Multi Arith	SVAMP	Date Understanding	Common SenseQA	Strategy QA
Fine-tune-CoT	7B	7B	71.05	65.55	53.33	40.67	83.78	69.53	60.84
+ Simple RL	7B	7B	75.00	73.11	76.11	50.67	**87.39**	77.64	65.21
+ Model RL	7B	7B	74.34	68.06	68.33	54.00	86.49	76.33	64.77
+ Correction RL	7B	7B	**76.32**	**76.47**	**80.00**	**55.33**	**87.39**	**77.97**	**66.08**

Table 7: Accuracy (%) in different datasets with different Reinforcement Learning (RL) strategies. Our proposed strategy, "Correction RL," shows the highest improvement. The method “Fine-tune-CoT + Correction RL” is our proposed method, which is also called “Self-motivated Learning”.

The score of the reward model reflects the quality of the rationale. As fig. 3a shows, we use the score threshold to filter the rationales during the PPO in the dataset SingleEq. We can see that the ratio of correct rationales increases with the score threshold. This means that the higher the reward model score, the more likely the rationale is correct. This shows that the reward model score can reflect the quality of the rationale to some extent.

The score and performance of the model are positively correlated. As depicted in fig. 3b, the average score and the performance of the model show a trend of improvement over time, although they do not always align perfectly. This misalignment might be attributed to imperfections in the reward model. fig. 3c illustrates the distribution of the reward score during PPO training. It is evident that the score distribution is dispersed, particularly for incorrect rationales. This dispersion suggests that the reward model might occasionally assign high scores to incorrect rationales, reinforcing the need for our reward score corrections. Such modifications effectively address the *reward hacking* challenge prevalent in RL algorithms.

Wrong answer leads to wrong rationales. We use the wrong answer prompt to generate some wrong rationales, as shown in table 2. We successfully induced the model to output wrong rationales with logical errors. This may be because when the model generates rationales with in-context learning, it will try to explain the wrong label as much as possible, resulting in errors.

The reward model trained using this rank information exhibits a certain degree of generalization. As depicted in table 5, we conducted reinforcement learning on Fine-tune-CoT (text-davinci-002) using the reward model trained with data generated by Llama 2 7B. This resulted in enhanced performance over the original model. Specifically, there was an improvement of 1.32 in SingleEq, 5.89 in AddSub, and 3.89 in MultiArith, with an average enhancement of 3.70 in performance. This indicates that the reward model trained with rank information possesses a degree of generalizability. This implies that even if we use different models to generate data for training the reward model, it can still be transferred to other models for reinforcement learning to improve their performance.

RL rectified some errors introduced during Supervised Fine-tuning. We assessed the accuracy variations of samples in the test set under three methods: Few-shot (FS), Supervised Fine-tuning (SFT), and Reinforcement Learning (RL). As shown in table 6, it is evident that, compared to introducing new errors, RL generally corrects the original errors brought about by SFT. Specifically, in the AddSub dataset, RL corrected the 59 errors that resulted from SFT without introducing any new errors. This indicates that RL can, to some extent, correct the errors introduced during SFT.

3.5.1. Strategy of Reward

Simple RL is Strong. As table 7 shows, the Simple RL strategy can improve the performance of the model in all datasets. The performance of the model in the SingleEq dataset is improved by 4.95%, and the performance in the AddSub dataset is improved by 7.61%. The performance in the MultiArith dataset is improved by 22.78%. This shows that the Simple RL strategy can effectively improve the performance of the model. It shows that the model can learn from the correct answer and the wrong answer rank to know what rationale is correct.

Model RL and the Challenge of Reward Hacking. As depicted in table 7, while the model reward strategy enhances performance, it still falls short of the achievements demonstrated by the simple RL strategy. A potential explanation for this disparity is the imperfection inherent in the reward model, which can lead to *reward hacking* as described by Skalse et al. (2022). This phenomenon may result in erroneously high scores for some rationales. A plausible cause for this could be the limited size of the training data used for the reward model. To mitigate the effects of reward hacking with limited data, we introduce the Correction RL strategy.

Prevent reward hacking with score correction. Based on the two strategies, we propose a new strategy called Correction RL. Compared to the Simple RL, we use the reward model to give better fine-grained feedback to the positive example. Compared to the Model RL, we use the method of comparing the answer to avoid the error of the reward model, which significantly eliminates the impact of reward hacking. As table 7 shows, the Correction RL strategy can improve the performance of the model in all datasets and is superior to the other two strategies.

4. Related Works

Reasoning Skills. Researchers in the literature have proposed many benchmarks requiring various reasoning skills, including commonsense reasoning (Zellers et al., 2018; Talmor et al., 2019; Bhagavatula et al., 2019; Geva et al., 2021b) numerical reasoning (Dua et al., 2019), multi-hop reasoning (Yang et al., 2018), arithmetic reasoning (Koncel-Kedziorski et al., 2015b; Roy and Roth, 2015; Miao et al., 2020; Patel et al., 2021b; Cobbe et al., 2021), logical reasoning (Liu et al., 2020; Yu et al., 2020), inductive reasoning (Sinha et al., 2019) and tabular reasoning (Chen et al., 2020; Zhu et al., 2021).

Advancements in Reasoning with Language Models. Language models (LMs), particularly large LMs (LLMs), have shown significant potential in addressing reasoning tasks with Chain-of-Thought Wei et al. (2022); Kojima et al. (2022). This technique necessitates the model to first generate a rationale, subsequently leading to an answer. An insightful observation by Wang et al. (2022b) reveals that incorporating a majority vote over multiple rationales can compensate for the shortfalls inherent in an LLM generating a singular, potentially incomplete rationale. However, the effectiveness of CoT Prompting diminishes with smaller LMs (Chung et al., 2022). Several studies have ventured into enhancing the reasoning abilities of LMs through diverse methodologies. For instance, Deng et al. (2021) utilized internet-crawled data for training LMs. Techniques like logic-guided data augmentation were introduced by Asai and Hajishirzi (2020). On the other hand, Shen et al. (2021); Cobbe et al. (2021); Li et al. (2023b) advocated for the training of a verifier, the task of which is to rank solutions drawn from fine-tuned LMs. An alternate approach is to endow LMs with reasoning skills by devising training samples through human-crafted templates, a method endorsed by researchers such as Geva et al. (2020); Yoran et al. (2022); Campagna et al. (2020); Wang et al. (2022a). Taking a step further, Pi et al. (2022) proposed the integration of reasoning faculties into LMs via continual pre-training on program execution data. There have been attempts to generate explanations for datasets using boosting methods by Zelikman et al. (2022) with hint. Lastly, fostering the CoT capabilities of smaller models by leveraging large models' rationale generation in diverse datasets is a concept Kim et al. (2023); Ho et al. (2023); Wang et al. (2023); Li et al. (2023a). While many focus on enhancing the reasoning capabilities of large language models or rely on extra language models or manual efforts to generate data to improve the performance of the smaller models, our objective is to fully tap into the potential of the model itself, reduce dependency on large-scale models and manual annotations, thereby improving the reasoning prowess of these compact models.

Reinforcement learning from human feedback Reinforcement learning from human feedback (RLHF) involves training models to perform tasks through feedback obtained from human evaluators, as opposed to traditional reward signals from an environment (Stiennon et al., 2020). Such methods have proven effective in refining models' behaviors, especially when environment rewards are sparse or ambiguous. In recent studies, such as Nakano et al. (2021); Ouyang et al. (2022), RLHF has been utilized to fine-tune large language models by collecting comparison data, where multiple model responses are ranked by quality. There are some methods do RLHF with large-scale models and extensive human feedback (Ouyang et al., 2022; Lightman et al., 2023; Uesato et al., 2022; Luo et al., 2023). And most methods in RLHF necessitate large models or extensive manually annotated data and typically focus on value alignment and safety alignment (Bai et al., 2022; Ganguli et al., 2022; Dai et al., 2023). In contrast, we leverage similar techniques to enhance the model's reasoning capabilities while reducing the dependence on large models and manual annotation.

5. Conclusion

We propose "Self-motivated Learning", a task-agnostic approach designed to enhance reasoning performance in LMs while decreasing reliance on large models and manual annotations. This framework is grounded in the idea that a rationale leading to the correct answer is superior to one leading to an incorrect answer. We conducted experiments across 8 datasets encompassing three categories of complex reasoning, demonstrating that our method can significantly enhance model performance without external annotation.

Acknowledgments

We gratefully acknowledge the support of the National Natural Science Foundation of China (NSFC) via grant 62236004 and 62206078.

6. Bibliographical References

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.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online. Association for Computational Linguistics.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2023. Safe rlhf: Safe reinforcement learning from human feedback.

Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. ReasonBERT: Pretrained to reason with distant supervision. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6112–6127, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, Andy Jones, Sam Bowman, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Nelson Elhage, Sheer El-Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Danny Hernandez, Tristan Hume, Josh Jacobson, Scott Johnston, Shauna Kravec, Catherine Olsson, Sam Ringer, Eli Tran-Johnson, Dario Amodei, Tom Brown, Nicholas Joseph, Sam McCandlish, Chris Olah, Jared Kaplan, and Jack Clark. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021a. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies.

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

Yuxian Gu, Pei Ke, Xiaoyan Zhu, and Minlie Huang. 2022. Learning instructions with unlabeled data for zero-shot cross-task generalization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1617–1634, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning.

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. Curran Associates, Inc.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015a. Parsing algebraic word problems into equations.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015b. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. 2022. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023a. Symbolic chain-of-thought distillation: Small models can also “think” step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2665–2679, Toronto, Canada. Association for Computational Linguistics.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023b. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning.

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

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

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.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

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

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021b. 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*, pages 2080–2094.

Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Yan Gao, Qiang Fu, Jian-Guang Lou, and Weizhu Chen. 2022. Reasoning like program executors. *arXiv preprint arXiv:2201.11473*.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

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

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward hacking.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge.

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.

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

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

Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023. SCOTT: Self-consistent chain-of-thought distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5546–5558, Toronto, Canada. Association for Computational Linguistics.

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Zhou, and Nan Duan. 2022a. Logic-driven context extension and data augmentation for logical reasoning of text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1619–1629, Dublin, Ireland. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. WorldTree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5456–5473, Marseille, France. European Language Resources Association.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Seonghyeon Ye, Hyeonbin Hwang, Sohee Yang, Hyeongu Yun, Yireun Kim, and Minjoon Seo. 2023. In-context instruction learning. *arXiv preprint arXiv:2302.14691*.

Ori Yoran, Alon Talmor, and Jonathan Berant. 2022. Turning tables: Generating examples from semi-structured tables for endowing language models with reasoning skills. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6016–6031, Dublin, Ireland. Association for Computational Linguistics.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance.

7. Language Resource References

Geva, Mor and Khashabi, Daniel and Segal, Elad and Khot, Tushar and Roth, Dan and Berant, Jonathan. 2021. *Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies*. MIT Press.

Hosseini, Mohammad Javad and Hajishirzi, Hannaneh and Etzioni, Oren and Kushman, Nate. 2014. *Learning to solve arithmetic word problems with verb categorization.* Citeseer.

Koncel-Kedziorski, Rik and Hajishirzi, Hannaneh and Sabharwal, Ashish and Etzioni, Oren and Ang, Siena Dumas. 2015. *Parsing algebraic word problems into equations*. MIT Press.

Patel, Arkil and Bhattamishra, Satwik and Goyal, Navin. 2021. *Are NLP Models really able to Solve Simple Math Word Problems?*

Roy, Subhro and Roth, Dan. 2016. *Solving general arithmetic word problems*.

Srivastava, Aarohi and Rastogi, Abhinav and Rao, Abhishek and Shoeb, Abu Awal Md and Abid, Abubakar and Fisch, Adam and Brown, Adam R and Santoro, Adam and Gupta, Aditya and Garriga-Alonso, Adrià and others. 2022. *Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models*.

Talmor, Alon and Herzig, Jonathan and Lourie, Nicholas and Berant, Jonathan. 2018. *Commonsenseqa: A question answering challenge targeting commonsense knowledge*.