Self-Training with Direct Preference Optimization Improves Chain-of-Thought Reasoning

Tianduo Wang[†] **Shichen Li**[‡] **Wei Lu**[†]
[†]StatNLP Research Group, Singapore University of Technology and Design
[‡]Soochow University
{tianduo_wang,luwei}@sutd.edu.sg scli_21@outlook.com

Abstract

Teaching small-scale language models to perform math reasoning is a valuable yet challenging task. Besides obtaining labeled data from human experts, one of the most common ways to collect high-quality data is by sampling from a larger and more powerful language model. Although previous works have demonstrated the effectiveness of this method, such a knowledge distillation paradigm can be costly and unstable, especially considering that many large language models, such as GPT-4 (OpenAI, 2023), are closed-source, proprietary, and their behaviors are unpredictable. In this work, to avoid relying on outputs from large models, we demonstrate that the reasoning abilities of small-scale language models can be enhanced through self-training, which involves training models with their own outputs. We also show that the conventional self-training can be further augmented by an alignment algorithm called Direct Preference Optimization (DPO) (Rafailov et al., 2023). We empirically found that models trained with the DPO objective are capable of making better generations that largely benefit multi-turn self-training. The experimental results show our models outperform the existing models with comparable sizes on the GSM8K benchmark with minimal resource requirements.[1]

1 Introduction

Making language models (LMs) perform mathematical reasoning is a valuable, yet challenging research objective (Hendrycks et al., 2021; Cobbe et al., 2021). Recent works focus on enhancing large-scale LMs' reasoning abilities, e.g., chain-of-thought prompting (Wei et al., 2022b; Kojima et al., 2022), continual pretraining (Azerbayev et al., 2024), or adding external verifiers (Li et al., 2023b). However, the research question of how to improve LMs with smaller sizes (e.g., less than 1 billion parameters) is still under-explored.

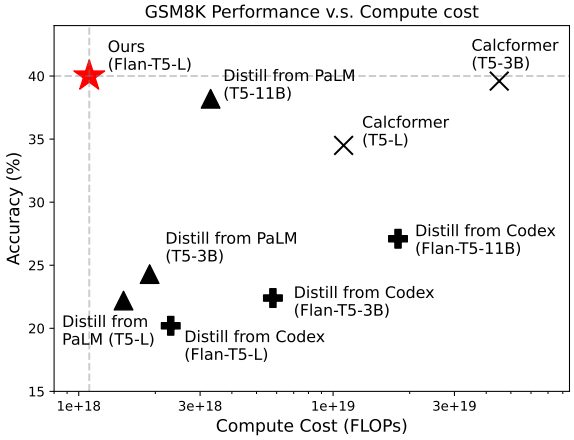

Figure 1: GSM8K performance v.s. computational cost. Our approach outperforms baseline models with comparable sizes while minimizing the required compute. Comparisons include two knowledge distillation techniques, i.e., Codex distillation (Fu et al., 2023), and PaLM distillation (Magister et al., 2023), and a data augmentation method, Calcformer (Kadlčík et al., 2023).

Recent studies (Fu et al., 2023; Ho et al., 2023; Magister et al., 2023) demonstrate that the reasoning capabilities of smaller LMs can be significantly enhanced through learning from the outputs of larger and more advanced LMs, such as Codex (Chen et al., 2021) and PaLM (Chowdhery et al., 2022). Despite the relative ease of implementation, the associated costs can be substantial. The computational demand, measured in *floating-point operations* (FLOPs), is considerably higher for large LMs. Additionally, the use of proprietary and closed-source large LMs for data annotation can incur significant economic costs. Ho et al. (2023) demonstrated that annotating multiple reasoning chains for a single question with large LMs can markedly enhance the performance of smaller models. Hence, a trade-off between cost and performance exists.

[1]Our code and data are released at https://github.com/tianduowang/dpo-st.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11917–11928

Another line of work focuses on making improvements via *self-training* (Zelikman et al., 2022; Gulcehre et al., 2023; Singh et al., 2023). Instead of using data generated by larger models, the essence of self-training methods is to make small models learn from their own generations. While these methods demonstrate the effectiveness of utilizing self-generated data, their success largely depends upon the pre-existing abilities of the models. For example, Zelikman et al. (2022) started by few-shot prompting a large model, i.e., GPT-J (Wang and Komatsuzaki, 2021), to self-generate rationales, which is an emergent ability that only comes with sufficiently large models (Wei et al., 2022a). However, it is still unclear whether small-scale LMs can benefit from self-training.

Recently, we have witnessed that *reinforcement learning from human feedback* (RLHF) has emerged as a prominent method to precisely modify LMs' behavior towards human preference (Ouyang et al., 2022; Casper et al., 2023). In this work, we propose to augment the self-training algorithm with an RLHF training process, i.e., Direct Preference Optimization (Rafailov et al., 2023), for its better performance and stability. Our experimental results demonstrate that the proposed method can significantly improve LMs' reasoning capabilities while minimizing the computational costs. We visualize the relationship between the downstream task performance and computational cost over a series of specialized models in Figure 1. The computational costs for each model are estimated following previous practice (Kaplan et al., 2020; Yuan et al., 2023). It can be observed that our method not only achieves the highest accuracy, but also minimizes the computational demand by learning from its own generations. Overall, the main contribution of this work can be summarized as follows:

- We propose a novel extension to the classic self-training framework with Direct Preference Optimization, and we show its effectiveness through standard math problem-solving tasks.
- We demonstrate that this novel extension improves the reasoning abilities of the LMs with minimal computational resource requirements.
- We propose an efficient method for integrating LMs with external tools, significantly improving the performance without sacrificing much inference speed.

Algorithm 1 Self-training for CoT reasoning tasks

Input: pre-trained language model f_θ
labeled dataset $\mathcal{L} = \{(x^i, y^i, a^i)\}_{i=1}^{l}$
unlabeled dataset $\mathcal{U} = \{(x^i, a^i)\}_{i=1}^{u}$
Output: fine-tuned model $f_{\theta'}$

Fine-tune f_θ on $\mathcal{L}$ to get $f_{\theta'}$
repeat
  Build pseudo-labeled dataset $\mathcal{S}$:
  $\mathcal{S} = \{(x^i, \hat{y}^i, \hat{a}^i)\}_{i=1}^{s}$
  where $x^i \sim \mathcal{U}$ and $\hat{y}^i, \hat{a}^i \sim f_{\theta'}(\cdot|x^i)$
  Select $\mathcal{S}^\alpha \subset \mathcal{S}$ when $\hat{a}^i = a^i$
  Update $\mathcal{L} \leftarrow \mathcal{S}^\alpha \cup \mathcal{L}$
  Train f_θ on $\mathcal{L}$ to get a new $f_{\theta'}$
until convergence or max iteration is reached

2 Background

Math word problem solving The math word problem solving task can be formulated as a sequence-to-sequence task where the input x is a question asking for an unknown value and the output y is a rationale leading to the answer a (Cobbe et al., 2021). Normally, the answers can be extracted from the rationales via some rule-based methods, e.g., regular expressions. A generated rationale $\hat{y}$ is regarded as correct if the extracted answer $\hat{a}$ matches the gold answer a. Formally, the labeled dataset for a math word problem solving task with l instances can be represented as:

$$\mathcal{L} = \{(x^i, y^i, a^i)\}_{i=1}^{l}. \quad (1)$$

A common way for specializing a LM f_θ towards math reasoning with the labeled dataset $\mathcal{L}$ is *supervised fine-tuning* (SFT). It optimizes f_θ by minimizing the negative log likelihood loss $\mathcal{L}_{\text{SFT}}(\theta)$:

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{L}} -\Big[\sum_{t=1}^{T} \log f_\theta(y_t|x, y_{1:t-1})\Big], \quad (2)$$

where T is the length of the rationale y and we use y_t to represent the t-th token in y.

Self-training Self-training is one of the earliest approaches in semi-supervised learning (Scudder, 1965; Fralick, 1967) that has risen in popularity recently (He et al., 2019; Amini et al., 2022). This method first regards a base model trained with a labeled dataset $\mathcal{L}$ as teacher, and uses it to build a pseudo-labeled dataset $\mathcal{S}$ by annotating an unlabeled dataset $\mathcal{U}$. Then, a student model is trained with the combination of $\mathcal{L}$ and $\mathcal{S}$ that are expected

to outperform the teacher model. Such a framework has been shown effective in a wide range of natural language processing tasks, e.g., natural language understanding (Vu et al., 2021) and generation (He et al., 2019). A formal description of a self-training algorithm for CoT reasoning tasks is provided in Algorithm 1.

Previous studies have demonstrated that the quality of the pseudo-labels largely impacts the overall performance of the self-training algorithm (He et al., 2019; Amini et al., 2022). For example, Gulcehre et al. (2023) proposed to select high-quality pseudo-labels with a learned reward function. Zelikman et al. (2022) filtered the generated rationales to include only the ones that lead to correct answers. Although many methods are proposed to select pseudo-labels, few works discuss how to improve the fine-tuned model $f_{\theta'}$ so that more high-quality pseudo-labels can be generated. In this paper, we present a method to enhance $f_{\theta'}$ in each iteration so that higher-quality pseudo-labeled data can be generated.

Direct Preference Optimization The Reinforcement Learning from Human Feedback (RLHF) methods align LMs with human preference (Ouyang et al., 2022; Bai et al., 2022). The standard pipeline of RLHF requires to first train a reward model from human preference data. Then, the reward model is used to fine-tune language models via reinforcement learning objective, e.g., Proximal Policy Optimization (Schulman et al., 2017). A recent study propose Direct Preference Optimization (DPO) (Rafailov et al., 2023) to avoid explicitly training a reward model so that language models can be directly tuned with human preference data.

The DPO pipeline can be described as follows. First, given some prompt x, sample several completions from the reference model π_{ref} (normally it is the model after supervised fine-tuning):

$$y_1, y_2 \sim \pi_{\text{ref}}(\cdot \mid x). \tag{3}$$

Next, construct the DPO dataset $\mathcal{D}$ from the completions based on the human preference:

$$\mathcal{D} = \{(\, x^i,\, y_w^i,\, y_l^i\,)\}_{i=1}^{N}, \tag{4}$$

where y_w^i and y_l^i represent the winning and losing completions respectively. Then, we optimize the language model π_θ to minimize $\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}})$ which can be defined as follows:

$$\mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}} \Big[-\log \sigma\Big(r(y_w|x) - r(y_l|x)\Big)\Big], \tag{5}$$

where $r(\cdot|x) = \beta \log \frac{\pi_\theta(\cdot|x)}{\pi_{\text{ref}}(\cdot|x)}$ and β is a coefficient that controls π_θ's deviation from π_{ref}.

Algorithm 2 DPO-augmented self-training

Input: pre-trained language model f_θ
labeled dataset $\mathcal{L} = \{(x^i, y^i, a^i)\}_{i=1}^{l}$
unlabeled dataset $\mathcal{U} = \{(x^i, a^i)\}_{i=1}^{u}$
Output: fine-tuned model $f_{\theta'}$

Warm-up stage
Fine-tune f_θ on $\mathcal{L}$ to get $f_{\theta'}$
repeat
  # DPO step
  Generate DPO dataset $\mathcal{D}$:
    $\mathcal{D} = \{(\, x^i,\, y_w^i,\, y_l^i\,)\}_{i=1}^{N}$
    where $x^i \sim \mathcal{U}$ and $y_w^i,\, y_l^i \sim f_{\theta'}(\cdot|x^i)$
  Tune $f_{\theta'}$ with $\mathcal{L}_{\text{DPO}}$ on $\mathcal{D}$ to get f_{θ^d}
  # SFT step
  Build pseudo-labeled dataset $\mathcal{S}$:
    $\mathcal{S} = \{(x^i, \hat{y}^i, \hat{a}^i)\}_{i=1}^{s}$
    where $x^i \sim \mathcal{U}$ and $\hat{y}^i, \hat{a}^i \sim f_{\theta^d}(\cdot|x^i)$
  Select $\mathcal{S}^\alpha \subset \mathcal{S}$ when $\hat{a}^i = a^i$
  Update $\mathcal{L} \leftarrow \mathcal{S}^\alpha \cup \mathcal{L}$
  Train f_θ on $\mathcal{L}$ to get a new $f_{\theta'}$
until convergence or max iteration is reached

3 Method

In this section, we first describe the proposed approach. Then, we demonstrate how we integrate an external calculator into the model's decoding process which significantly improves LMs' performance on the downstream tasks.

3.1 DPO-Augmented Self-Training

Our approach starts with a *warm-up stage*, and then followed by an iterative process, where each iteration is composed of two sub-steps: *DPO step* and *SFT step*. The iterative process ends when the model performance converges or reaches the maximum iteration. A formal description of the proposed method is illustrated in Algorithm 2. An illustration of our method is presented in Figure 2.

Warm-up stage Like classic self-training, we start by fine-tuning the base model f_θ to optimize $\mathcal{L}_{\text{SFT}}(\theta)$ on the labeled data $\mathcal{L}$ to get a new model

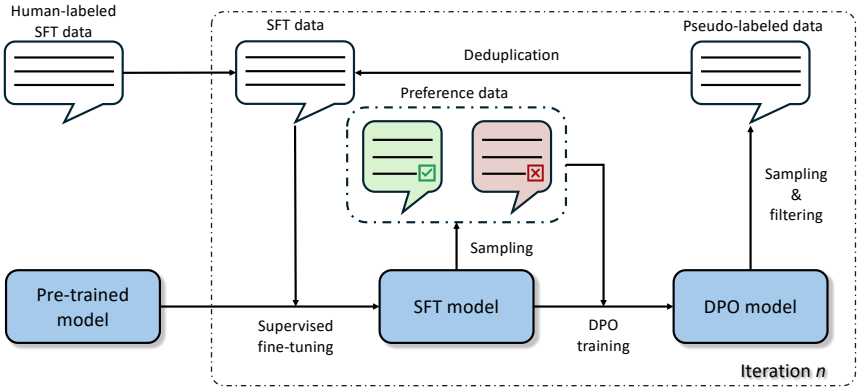

Figure 2: An illustration of the proposed DPO-augmented Self-Training framework. The conventional Self-Training method uses the SFT model to generate the pseudo-labels for the next iteration. In contrast, our method first optimize the SFT model with Direct Preference Optimization (DPO), and use the DPO model to produce the pseudo-labels.

$f_{\theta'}$. After this stage, we assume that $f_{\theta'}$ is capable of solving certain math problems. Specifically, given a math question x, $f_{\theta'}$ will generate a rationale $\hat{y}$ with answer $\hat{a}$.

Iterative step 1: DPO step In this step, we first sample rationales $\hat{y}$ from the fine-tuned model $f_{\theta'}$ given some questions x from $\mathcal{U}$. For each question x, we generate multiple rationales to build the DPO training dataset $\mathcal{D}$. As mentioned, for math problem solving tasks, it is easy to know whether a generated rationale $\hat{y}$ can be considered as correct. We label rationales with correct answers as winning completions, while consider rationales with incorrect answers as losing completions. Then, we train $f_{\theta'}$ on $\mathcal{D}$ to optimize the objective function $\mathcal{L}_{\text{DPO}}$ and get a DPO model f_{θ^d} in the end.

Iterative step 2: SFT step After obtaining f_{θ^d}, we use it to generate a new pseudo-labeled dataset $\mathcal{S}$ for the next-round supervised fine-tuning:

$$\mathcal{S} = \{(x, \hat{y}) | x \sim \mathcal{U}, \hat{y} \sim f_{\theta^d}(\cdot|x)\} \quad (6)$$

After generation, we clean $\mathcal{S}$ by eliminating rationales with incorrect answers and removing duplicates. Therefore, the pseudo-labeled dataset we obtained in the end is a subset of the original one, i.e., $\mathcal{S}^\alpha \subset \mathcal{S}$. The final training dataset is the combination of the original labeled dataset $\mathcal{L}$ and the newly-generated pseudo-labeled dataset $\mathcal{S}^\alpha$.

Notice that during this process, once we collect a new dataset, we train from the original base model f_θ instead of continually fine-tuning $f_{\theta'}$ to avoid overfitting, following previous practice (Zelikman et al., 2022; Singh et al., 2023).

```
Q: James writes a 3-page letter to 2
different friends twice a week. How many
pages does he write a year?
A: He writes each friend
3*2=<<3*2=6>>6 pages a week.
So he writes
6*2=<<6*2=12>>12 pages every week.
That means he writes
12*52=<<12*52=624>>624 pages a year.
#### 624
```

Figure 3: An example from the GSM8K dataset. The calculation annotations are highlighted in blue. All calculation steps are wrapped within special tokens <<...>>. During decoding, the calculator will be triggered when such patterns exist and the model's output tokens will be overridden by the calculator results. Following Cobbe et al. (2021), the calculation is performed with the python `eval()` function.

3.2 Batch Decoding with Calculator

We empirically observed that, in contrast to large LMs which excel at basic arithmetic calculations (Brown et al., 2020), smaller LMs like Flan-T5-Large exhibit poor performance on similar arithmetic tasks. This deficiency significantly impacts their ability to handle math reasoning tasks. To address this, various studies (Parisi et al., 2022; Schick et al., 2023; Kadlčík et al., 2023) have proposed augmenting these smaller models with an external calculator to enhance their math reasoning capabilities. However, many of these existing methods are limited to a batch size of one during decoding. This constraint substantially reduces the inference speed, which hinders their widespread adoption.

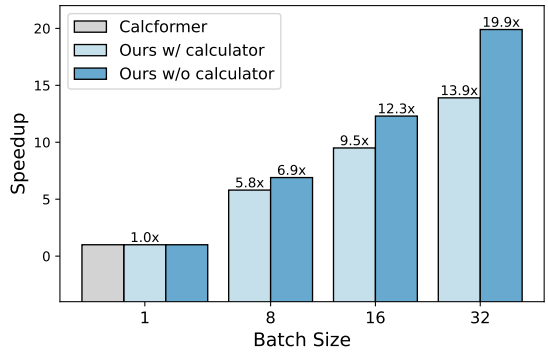

Figure 4: Inference speed comparison between our methods (w/ and w/o calculator) and Calcformer (Kadlčík et al., 2023) with varying batch sizes. The results are measured on a single NVIDIA A40 GPU.

To alleviate this issue, we propose a simple yet efficient method to enables the use of a large batch size during inference with an external calculator. We adopt the calculator annotation provided in the original GSM8K dataset (Cobbe et al., 2021). Figure 3 demonstrates an example of this annotation and describes how such annotations can be used during decoding. Our models are built with the Transformers library (Wolf et al., 2020). During generation, we adopt a customized `LogitsProcessor`[2] to override the model's generation. `LogitsProcessor` provides an interface to modify the language model's output tokens during generation.

To demonstrate the efficiency of the proposed solution, We compare the inference speed of our methods (w/ and w/o calculator) based on Flan-T5-Large against an open-source tool-using method, Calcformer (Kadlčík et al., 2023) based on T5-Large, in Figure 4. We find that when the batch size equals 1, all three methods have a similar inference speed of around 40 tokens per second. However, as the inference batch size increases, the speedup of our methods increases significantly.

4 Experiments

In this section, we first describe our experimental setup. Next, we present the performance of our models across a series of math word problem solving tasks, comparing them against a selection of competitive baselines. Finally, we empirically analyze what makes the proposed method effective.

[2] https://huggingface.co/docs/transformers/internal/generation_utils#logitsprocessor

Dataset	Split	# Data
GSM8K (Cobbe et al., 2021)	Train	6,705
	Validation	768
	Test	1,319
MultiArith (Roy and Roth, 2015)	Test	600
ASDiv (Miao et al., 2020)	Test	2,096
SVAMP (Patel et al., 2021)	Test	1,000

Table 1: Statistics of the datasets used in our experiments. The original GSM8K dataset only contains train and test split. We randomly select 768 training examples to construct the validation dataset in our experiments.

4.1 Setup

Base models We employ Flan-T5 models (Chung et al., 2024) as the base models in our experiments. Specifically, we consider two models of different sizes from the Flan-T5 model family: Flan-T5-Base (250M) and Flan-T5-Large (780M). We select Flan-T5 over the original T5 models (Raffel et al., 2019) as our backbone models based on the evidence from previous research (Chung et al., 2024; Fu et al., 2023), which demonstrated that instruction-tuned models, i.e., Flan-T5, outperform their pre-trained counterparts in math reasoning tasks. Besides Flan-T5 models, we also consider the Llama models (Touvron et al., 2023a,b; Meta, 2024) as our base models.

Datasets The labeled dataset $\mathcal{L}$ used in our experiments comes from the training split of the GSM8K dataset. Our unlabeled dataset $\mathcal{U}$ is also built upon GSM8K's training data by removing its annotated rationales y. For evaluation, we consider three other commonly used math reasoning tasks besides GSM8K: MultiArith (Roy and Roth, 2015), ASDiv (Miao et al., 2020), and SVAMP (Patel et al., 2021). Table 1 shows the statistics information of each dataset. Following previous practice (Fu et al., 2023), we only fine-tune the base models on the GSM8K training data while utilizing the rest three datasets to evaluate our models' out-of-domain performance as they do not have an official in-domain training split.

Evaluation metrics We use accuracy of the greedy decoding results as the main evaluation metric. The questions in the datasets in our experiments ask about the values of the unknown variables. The answers to these questions are real numbers that can be extracted from the model-generated rationales.

Method	Base Model	GSM8K	MultiArith	ASDiv	SVAMP
Supervised Fine-Tuning	Flan-T5-Base	18.1	54.2	26.2	19.5
Self-Training	Flan-T5-Base	25.9	73.8	28.2	**24.2**
DPO-aug Self-Training (*Ours*)	Flan-T5-Base	**27.2**	**74.3**	**29.2**	22.6
Supervised Fine-Tuning	Flan-T5-Large	30.8	77.2	38.1	33.6
Self-Training	Flan-T5-Large	35.6	86.2	42.5	34.8
DPO-aug Self-Training (*Ours*)	Flan-T5-Large	**37.4**	**89.0**	**42.8**	**36.8**

Table 2: Overall accuracies (%) over four math word problem solving tasks. Inspired by the previous practice (Fu et al., 2023), all the models in this table are only trained with the GSM8K training set (Cobbe et al., 2021). Hence, we report the in-distribution performance for GSM8K, while reporting the out-of-distribution performance for the other three datasets, i.e., MultiArith, ASDiv, and SVAMP.

Implementation details In every DPO step, we sample rationales from the SFT model $f_{\theta'}$ to build the DPO training data. We sample 5 rationales from $f_{\theta'}$ per question with a temperature of 0.7. We regard generated rationales $\hat{y}$ as winning ones y_w if it contains the correct answer, while regarding the rest as the losing ones y_l. For the SFT steps, we generate 3 rationales per question from the DPO-tuned model f_{θ^d} also with a temperature of 0.7. Only the correct generated rationales $\hat{y}$ will be selected to build the pseudo-labeled dataset $\mathcal{S}$. For both generated DPO data and SFT data, we make simple deduplication based on the Jaccard similarity scores with a threshold of 0.7.

4.2 Main Results

Baselines We mainly consider two baseline methods to compare with our method: Supervised Fine-Tuning (SFT) and Self-Training (ST). SFT baselines are trained on the original GSM8K annotations with the $\mathcal{L}_{\text{SFT}}(\theta)$ objective. The Self-Training baseline adheres to the procedure outlined in Algorithm 1. Consequently, this makes it the most directly comparable and relevant baseline for evaluating the effectiveness of our approach.

Comparison with baselines Table 2 shows the performance of our method compared with the baselines using two base models over four datasets. It can be observed that both the Self-Training baseline and our proposed method outperform the supervised fine-tuning baseline by a large margin, indicating the effectiveness of the general self-training framework. Although the Self-Training baselines make significant improvements over the SFT baselines, the proposed DPO-augmented Self-Training models consistently outperform them on both in-domain and out-of-domain tasks.

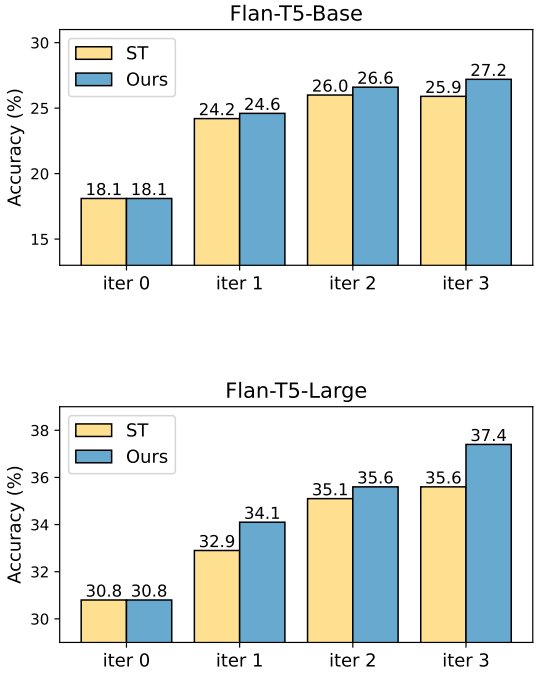

Figure 5: The performance (accuracy) of the proposed method on GSM8K over three iterations. "ST" stands for the Self-Training baseline. The "iter 0" represents the results of the SFT baselines.

Effect of iterative training In Figure 5, we demonstrate how our model improve over mulitple iterations using two base models: Flan-T5-Base and Flan-T5-Large. Both the ST baseline and our method start with a SFT step using the original GSM8K annotation at iteration 0. We can observe that, as iterations progress, our method consistently outperforms ST, which suggests that the proposed method offers a substantial improvement over ST after repeated training cycles. We also notice that the degree of improvement gradually approaches zero as the iterative process proceeds, suggesting that the model has converged in the last iteration.

Method	Base Model	# Annotations	Annotator	Tools	Acc.
Supervised fine-tuning					
CoT (Shridhar et al., 2023)	GPT-2-Large	7K	Human	✗	14.1
Self-consistency (Khalifa et al., 2023)	Flan-T5-Large	7K	Human	✓	33.3
GRACE (Khalifa et al., 2023)	Flan-T5-Large	7K	Human	✓	36.3
Calcformer (Kadlčík et al., 2023)	T5-Large	30K	Human	✓	34.2
Knowledge Distillation					
Socratic CoT (Shridhar et al., 2023)	GPT-2-Large	7K	GPT-3 175B	✗	21.1
CoT from CodeX (Fu et al., 2023)	Flan-T5-Large	100K	CodeX	✗	20.2
CoT from PaLM (Magister et al., 2023)	T5-Large	6K	PaLM 540B	✓	22.2
Ours					
DPO-aug Self-Training (K=3)	Flan-T5-Large	7K	Human	✓	37.4
DPO-aug Self-Training (K=5)	Flan-T5-Large	7K	Human	✓	39.1
DPO-aug Self-Training (K=10)	Flan-T5-Large	7K	Human	✓	**40.0**

Table 3: Detailed comparison among existing methods with comparable model sizes on the GSM8K test set. The "Annotator" column indicates how the rationales of the training data are generated. In this column, "Human" refers to the labels from the original GSM8K dataset (Cobbe et al., 2021) that are written by human annotators. The "Tools" column indicates whether external tools, e.g., calculator or code interpreter, are applied during inference.

4.3 Comparison with Existing Methods

In this section, we compare our methods with existing approaches. Additionally, we scale up the proposed method by increasing the number of sampled pseudo-labels per question, denoted by the hyperparameter K as in Yuan et al. (2023).

Table 3 presents a detailed comparison between our method and exisiting methods using a simialr base model size. The base models we considered include GPT-2-Large (Radford et al., 2019), T5-Large (Raffel et al., 2019), and Flan-T5-Large (Chung et al., 2024). All these models have approximately 770 million parameters. As shown in Table 3 our approach not only outperforms other methods on the GSM8K benchmark, but also demonstrates remarkable label efficiency by utilizing only the annotations from the original GSM8K dataset.

In Table 4, we further verify the effectiveness of the proposed method with the Llama model family (Touvron et al., 2023a,b; Meta, 2024). comparing them with several state-of-the-art closed-source models as well as open-source models with similar sizes. We observe a substantial performance gap between proprietary and open-source models. Among the open-source models, those utilizing knowledge distillation consistently outperform their counterparts without such enhancement. Notably, our models using Llama-1-7b (Touvron et al., 2023a) and Llama-2-7b (Touvron et al., 2023b) base models surpass other open-source alternatives that do not employ knowledge distillation, achieving accuracies of 44.7% and 54.7% respectively. Furthermore, our model employing the latest Llama-3-8b (Meta, 2024) matches or exceeds the performance of earlier models with knowledge distillation, demonstrating a significant accuracy of 68.8%.

Method	Base Model	Acc.
Closed-source models		
Claude-3-Opus (Anthropic, 2024)	-	95.0
Claude-2 (Anthropic, 2023)	-	88.0
GPT-4 (OpenAI, 2023)	-	92.0
Flan-PaLM-2 (Anil et al., 2023)	-	84.7
PaLM-2 (Anil et al., 2023)	-	80.7
Models w/ knowledge distillation		
RFT-U13B (Yuan et al., 2023)	Llama-1-7b	49.3
RFT-U33B (Yuan et al., 2023)	Llama-2-7b	51.2
MAmooTH-CoT (Yue et al., 2023)	Llama-2-7b	50.5
LEMA (An et al., 2023)	Llama-2-7b	54.1
WizardMath (Luo et al., 2023)	Llama-2-7b	54.9
MetaMath (Yu et al., 2024)	Llama-2-7b	66.5
MuggleMath (Li et al., 2023a)	Llama-2-7b	68.4
ToRA (Gou et al., 2024)	Llama-2-7b	68.8
Models w/o knowledge distillation		
SFT (Yuan et al., 2023)	Llama-1-7b	35.9
RFT (K=12) (Yuan et al., 2023)	Llama-1-7b	41.6
RFT (K=100) (Yuan et al., 2023)	Llama-1-7b	41.7
SFT (Yuan et al., 2023)	Llama-2-7b	41.6
RFT (K=12) (Yuan et al., 2023)	Llama-2-7b	45.3
RFT (K=100) (Yuan et al., 2023)	Llama-2-7b	47.5
Ours		
DPO-ST (K=10)	Llama-1-7b	44.7
DPO-ST (K=10)	Llama-2-7b	54.7
DPO-ST (K=10)	Llama-3-8b	**68.8**

Table 4: Comparison with the state-of-the-art close-source models and open-source models based on Llama model family (Touvron et al., 2023a,b; Meta, 2024).

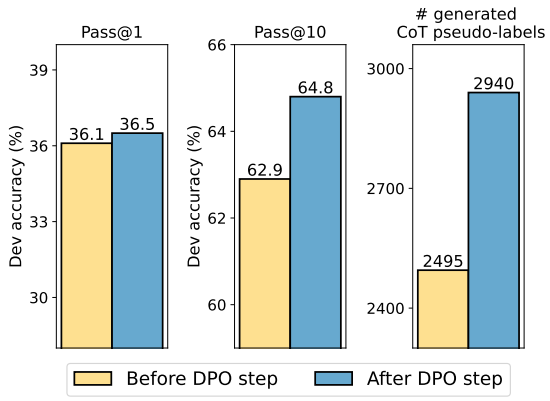

Figure 6: Effects of the DPO step. **Left:** we report the greedy decoding results for Pass@1. **Middle:** For Pass@10, the solutions are sampled with temperature 0.7. **Right:** We count the number of generated pseudo-labels after deduplication.

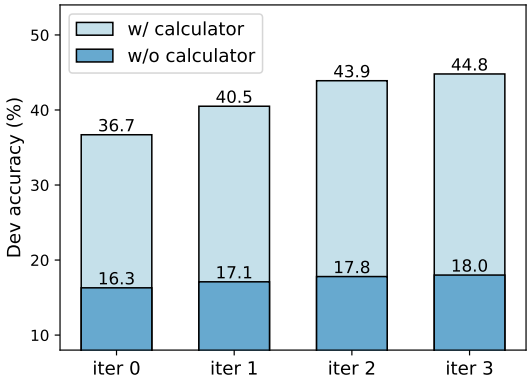

Figure 7: GSM8K development set accuracy of Flan-T5-Large with and without the use of an external calculator during inference.

4.4 Effects of the DPO Step

As mentioned earlier, the main difference between the proposed method and the classic self-training is the DPO step in every iterative process. We now analyze how the DPO steps improve self-training. Figure 6 compares the performance of models before and after the DPO step in the first iteration on the Pass@K metrics. Pass@K measures the probability that at least one of the K generated solutions for a problem is correct, which serves as a gauge for both the quality and the variety of the model-generated solutions. The models we investigate here are fine-tuned from the Flan-T5-Large.

As shown in Figure 6, the DPO step yields only marginal improvements over the SFT model in the Pass@1 performance on the development set. However, the performance significantly improves when multiple rationales, i.e., 10 solutions per question, are sampled with temperature 0.7 (measured with the Pass@10 metric). This indicates that the DPO training objective makes language models inclined to generate rationales of both high quality and diversity. We also compare the number of generated rationales on the training set $\mathcal{L}$ for models with and without the DPO step. Figure 6 (right) clearly shows that the model after the DPO step can produce more SFT data for the next iteration.

4.5 Effects of External Calculator

Driven by the observation that small-scale LMs frequently make basic calculation errors, we develop a simple yet efficient method that integrates an external calculator into the models' decoding process. To evaluate the impact of this integration, we conduct an ablation study by omitting the calculator and present the findings in Figure 7. Our results indicate that decoding without the calculator markedly reduces accuracy across all iterations. We believe that this is because models will generate large amount of false positive pseudo-labels without calculator, that is, the generated pseudo-labels may have correct final answers but make errors in the intermediate reasoning steps.

5 Related Work

Learning from pseudo-labels Supervised fine-tuning (SFT) is prevalent technique employed to enhance the performance of pre-trained language models on specific downstream tasks (Ouyang et al., 2022; Chung et al., 2024). However, this method heavily depends on the availability of high-quality labeled data, which can be both expensive and labor-intensive to procure (Brown et al., 2020). To address this limitation, various strategies have been developed to generate high-quality pseudo-labels using either unlabeled or synthetic data for a wide range of applications, including text classification (Xie et al., 2020), sentence representation learning (Wang and Lu, 2022), instruction tuning (Honovich et al., 2022), and math reasoning (Wang and Lu, 2023). Recent advancements in this area primarily focus on two directions: self-training and knowledge distillation. The key difference between these methods lies in the source of the pseudo-labels: self-training uses the model's own predictions on unlabeled data, while knowledge distillation utilizes the insights from larger, more powerful models.

Self-training in language model Recently, we have witnessed a large number of works focusing on self-training algorithms for language models (He et al., 2019; Zelikman et al., 2022; Yuan et al., 2023). Most of such methods are built upon the classic self-training framework (Scudder, 1965). He et al. (2019) empirically studied the effectiveness of self-training in natural language generation tasks, e.g., summarization and translation. Zelikman et al. (2022) proposed *self-taught reasoner* (STaR), which demonstrated that language models can be iteratively improved from its own generation, even there are no gold rationales provided. Yuan et al. (2023) proposed *rejection sampling fine-tuning* to improve language models' math reasoning abilities. This method can be interpreted as only executing one iteration of the self-training algorithm. Singh et al. (2023) proposed ReST^{EM}, a self-improving algorithm based on expectation-maximization framework. This method demonstrates significant improvements in problem-solving tasks, e.g., math reasoning and code generation.

Knowledge distillation from LLMs Many of the recent research efforts demonstrated large language models (LLMs) are capable of doing math reasoning (Wei et al., 2022b; Gao et al., 2022; OpenAI, 2023; Anil et al., 2023; Anthropic, 2023, 2024). Therefore, a recent line of work focuses on improving smaller language models' reasoning abilities by distilling chain-of-thought pseudo-labels from LLMs (Ho et al., 2023; Magister et al., 2023; Fu et al., 2023). For example, Luo et al. (2023) proposed Reinforcement Learning from Evol-Instruct Feedback built upon Evol-Instruct (Xu et al., 2023), which requires ChatGPT to provide the training signals. An et al. (2023) demonstrated that language models can effectively learn from the mistakes that can be corrected by larger models, e.g., GPT-4 (OpenAI, 2023), during supervised fine-tuning. Yu et al. (2024) proposed a novel question bootstrapping method with the help of larger models to augment the existing training dataset. Although these methods are shown to have promising experimental results, they are costly to implement as large models cost more FLOPs during inference. Our work demonstrates that small-scale language models can also learn from their own generations like the larger ones (Zelikman et al., 2022), which is more resource-efficient compared with the knowledge distillation methods.

6 Conclusion

We present an effective and resource-efficient method called DPO-augmented Self-Training (DPO-ST), which augments the original Self-Training algorithm with Direct Preference Optimization (Rafailov et al., 2023). Unlike previous studies that improve small-scale language models' reasoning abilities by distilling a larger and more powerful model, we argue that small models that are trained merely on the limited human-labeled data can improve themselves significantly. We also empirically find that models trained with DPO loss can generate pseudo-labeled data with higher quality and diversity. Our experiments demonstrate that the proposed method not only outperforms existing methods with comparable model sizes on the GSM8K benchmark, but also achieves remarkable resource efficiency in terms of both computational cost and the requirements of human-labeled data.

Limitations

Use of unlabeled data Our method is built upon the classic self-training algorithm, which provides an effective semi-supervised learning framework that makes good use of unlabeled data. However, this work doesn't explore the use of unlabeled data explicitly. Future research efforts can be made to explore how to collect high-quality unlabeled data for math word problem solving. In other words, we need to find an efficient method for collecting unlabeled data $\mathcal{U} = \{(x_i, a_i)\}_{i=1}^{u}$ that for each math question x_i, there is a corresponding ground-truth answer a_i.

Generalization to other tasks One of the limitations of this work is the narrow scope of our experiments, which were exclusively conducted on math reasoning tasks. The primary reason for this limitation is the lack of appropriate training data for other reasoning tasks. As our method requires a set of training data with chain-of-thought labels, many existing reasoning tasks lack such annotations, making it challenging to extend our experiments beyond the current scope. Future research may focus on identifying and developing suitable datasets for a wider range of reasoning tasks to fully evaluate the applicability and effectiveness of our method across different reasoning tasks.

Acknowledgements

This work was done when Shichen Li was a visiting student at the StatNLP Research Group of SUTD. We would like to thank the anonymous reviewers, our meta-reviewer, and senior area chairs for their constructive comments and support on this work. This research/project is supported by Ministry of Education, Singapore, under its Academic Research Fund (AcRF) Tier 2 Programme (MOE AcRF Tier 2 Award No: MOET2EP20122-0011), the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Program (AISG Award No: AISG2-RP-2020-016), and Ministry of Education, Singapore, under its Tier 3 Programme (The Award No.: MOET320200004). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the funding agencies.

References

Massih-Reza Amini, Vasilii Feofanov, Loïc Pauletto, Emilie Devijver, and Yury Maximov. 2022. Self-training: A survey. *arXiv preprint arXiv:2202.12040*.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Anthropic. 2023. Claude 2. `https://www.anthropic.com/news/claude-2`. Accessed: 2024-05-06.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. Accessed: 2024-05-06.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *Proceedings of ICLR*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, et al. 2020. Language models are few-shot learners. In *Proceedings of NeurIPS*.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, J'er'emy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, et al. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research*.

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

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. *Journal of Machine Learning Research*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Stanley C. Fralick. 1967. Learning to recognize patterns without a teacher. *IEEE Trans. Inf. Theory*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *Proceedings of ICML*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *Proceedings of ACL*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2019. Revisiting self-training for neural sequence generation. *arXiv preprint arXiv:1909.13788*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of NeurIPS*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of ACL*.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.

Marek Kadlčík, Michal Štefánik, Ondřej Sotolář, and Vlastimil Martinek. 2023. Calc-x and calcformers: Empowering arithmetical chain-of-thought through interaction with symbolic systems. In *Proceedings of EMNLP*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv*.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Grace: Discriminator-guided chain-of-thought reasoning. In *Findings of EMNLP*.

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

Chengpeng Li, Zheng Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. 2023a. Query and response augmentation cannot help out-of-domain math reasoning generalization. *arXiv preprint arXiv:2310.05506*.

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 ACL*.

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. *arXiv preprint arXiv:2308.09583*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of ACL*.

Meta. 2024. Llama 3. `https://llama.meta.com/llama3/`. Accessed: 2024-06-01.

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 ACL*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of NAACL*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

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. In *Proceedings of NeurIPS*.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of EMNLP*.

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. In *Proceedings of NeurIPS*.

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

H. J. Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of ACL*.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, et al. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Tu Vu, Minh-Thang Luong, Quoc Le, Grady Simon, and Mohit Iyyer. 2021. STraTA: Self-training with task augmentation for better few-shot learning. In *Proceedings of EMNLP*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Tianduo Wang and Wei Lu. 2022. Differentiable data augmentation for contrastive sentence representation learning. In *Proceedings of EMNLP*.

Tianduo Wang and Wei Lu. 2023. Learning multi-step reasoning by solving arithmetic tasks. In *Proceedings of ACL*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Proceedings of NeurIPS*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Proceedings of 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. *arXiv preprint arXiv:2304.12244*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *Proceedings of ICLR*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv: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. *arXiv preprint arXiv:2309.05653*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Proceedings of NeurIPS*.