Targeted Training for Math Word Problems with Large Language Models

Anonymous EMNLP submission

Abstract

After recent gains achieved by large language models (LLMs) on numerical reasoning tasks, it has become of interest to have LLMs teach small models to improve on math word problems (MWPs). Instructing LLMs to generate Chains of Thought (CoTs) to fine-tune small models is an established approach. However, small models are passive in this line of work, and may not be able to exploit the provided training data. In this paper, we propose a novel targeted training strategy to match LLM’s assistance with small models’ capacities. The small model will proactively request LLM’s assistance when it sifts out confusing training data. Then, LLM refines such data by successively revising reasoning steps and reducing question complexity before feeding the small model. Experiments show that this targeted training approach remarkably improves the performance of small models on a range of MWP datasets by 12–25 percent, making small models even competitive with some LLMs.

1 Introduction

The rapid advancement of LLMs has yielded several noteworthy outcomes, including a notable enhancement in the numerical reasoning proficiency of LLMs, as evidenced by solving MWPs (Frieder et al., 2023). Evaluating the multi-step reasoning capability of LLM with MWPs appears to have gained popularity, and recently proposed approaches like CoT (Wei et al., 2022; Kojima et al., 2022) have substantially enhanced LLM’s performance on multi-step numerical reasoning. A MWP example is shown in Figure 1.

Motivation. However, some challenging MWP datasets like GSM8k (Cobbe et al., 2021), are still tough for small models to solve due to their limited capacities to comprehend complex questions and perform numerical reasoning (Muffo et al., 2022; Yuan et al., 2023). To address these limitations, there is a trend in employing LLM to strengthen small models by distilling CoT capabilities (Fu et al., 2023; Magister et al., 2022), question decomposition capabilities (Shridhar et al., 2023) or code generation capabilities (Zhu et al., 2023) to small models, with the aim of improving the multi-step numerical reasoning capabilities of small models. These methods, however, place small models in a passive role and may fail to match LLM’s assistance with small models’ capacities. By contrast, we propose that small models should be more proactive, e.g., to sift out training data that confuses them and then request the assistance of LLM to tailor training data to match their capacities.

Our Work. To implement this idea, we present an approach called targeted training, which is made up of two stages successively targeted on tailoring the reasoning steps and question complexity. Different from existing approaches, we place small models in a more proactive state of receiving feeds from LLM. We let the small model actively sift out the training subset that is confusing. The LLM then refines such data by revising reasoning steps and reducing question complexity, and eventually the small model selectively accepts the refined data as training data, achieving the goal of tailoring training data to meet the capacities of the small model and improving its performance. To summarize, our technical contributions include

- a novel targeted training framework where training data is sifted and refined with the help of LLM to match small model’s capacities,
• a thoughtful design of a two-stage targeted training process where both reasoning steps and question complexity are tailored.

Our code and data are packed in the supplementary material and will be open source on GitHub after acceptance.

2 Related Work

2.1 Knowledge Distillation from LLMs

Transferring the capabilities of large models to small models is a long-standing research challenge, known as knowledge distillation (Buciluă et al., 2006; Hinton et al., 2015; Gou et al., 2021). Fu et al., 2023 and Magister et al., 2022 observed that small models will have partial CoT capability by using LLMs to generate CoTs for questions and then using them to fine-tune small models. By distilling the question decomposition capability of LLMs into small models, Shridhar et al., 2023 enable small models to decompose a complex question into simple questions and subsequently solve them. In addition, efforts have been made to extract rationales generated by LLMs in order to reduce the reliance of small models on large-scale data (Hsieh et al., 2023). Through program-aided distilling, Zhu et al., 2023 reduce the reasoning errors in CoTs caused by the hallucination of LLMs.

By contrast, our targeted training allows small models to be more proactive, via multiple phases of targeted requests for assistance from LLMs. Small models sift the results returned by LLMs, as opposed to passively using all the feeds.

2.2 Math Word Problem Solving

One of the most important indicators of a model’s capacity for numerical reasoning is its ability to solve MWPs. MWP solving has been present in the research of NLP for years (Hosseini et al., 2014; Huang et al., 2016). The task provides a text describing a question and asks for an answer that can be derived from arithmetic expressions. The evolution of approaches to solving MWPs has progressed from statistical learning methods (Hosseini et al., 2014; Kushman et al., 2014) and rule/template-based methods (Shi et al., 2015; Wang et al., 2019) to deep learning based methods (Huang et al., 2021; Wu et al., 2021; Qin et al., 2021; Shen et al., 2021; Jie et al., 2022).

Recent attention has been given to LLMs and their superior performance on reasoning (Frieder et al., 2023). In-context learning and prompting strategies with LLMs have demonstrated their potential to solve MWPs, represented by CoT prompting (Wei et al., 2022) and its variations (Lyu et al., 2023; Yao et al., 2023). Interestingly, Kojima et al., 2022 have found that adding the phrase “Let’s think step by step” after the question can improve LLMs’ numerical reasoning ability in a zero-shot setting. Employing external tools (Gao et al., 2022; Schick et al., 2023) and the majority voting approach (Wang et al., 2023) used with LLMs can also benefit MWP solving.

While the aforementioned methods mainly focus on the reasoning steps, recent research has suggested that improving the question side by question polishing/refining (Xi et al., 2023) or question decomposition (Zhou et al., 2023) can also be advantageous to LLMs’ reasoning capability. In this regard, our approach improves both the question- and reasoning-sides. Improvements on the reasoning side help train small models with reasoning steps that match their capacities, and improvements on the question side enable small models to be trained only with questions that they can comprehend.

3 Approach

A MWP dataset \( D \) is a set of triples \( \langle q, e, a \rangle \), where \( q \) is a question text and \( e \) is an arithmetic expression that can be computed to get an answer \( a \). Given \( q \), the objective is to generate \( e \) and compute \( a \).

We assume that the arithmetic expression \( e \) is described in a formal domain specific language (DSL), including four binary operations
The objective of our targeted training is to sift and refine the original training set \( \mathcal{D} \) by \( \hat{\mathcal{D}} \) out the training data that causes confusion, denoted expressions in \( \mathcal{D} \) by \( \hat{\mathcal{D}} \). In the first stage, reasoning steps (i.e., arithmetic expressions) are targeted. The small model sifts and refines the original training set \( \hat{\mathcal{D}} \). This, in turn, will fit small model’s capacities. This, in turn, will facilitate the training of a better small model.

Our process of tailoring \( \mathcal{D} \) to small model’s capacities consists of two stages, covering both the reasoning steps (Section 3.2) and the question complexity (Section 3.3) because both sides may confuse the small model during training. We will explain the pipeline shown in Figure 2.

### 3.1 Overview of Targeted Training

The objective of our targeted training is to sift and refine the original training set \( \mathcal{D} \), resulting in a high-quality and small model-friendly training set \( \hat{\mathcal{D}} \) that fits small model’s capacities. This, in turn, will facilitate the training of a better small model.

Table 2: A MWP with different arithmetic expressions.

<table>
<thead>
<tr>
<th>Question</th>
<th>Expression e₁</th>
<th>Expression e₂</th>
<th>Expression e₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noah is a painter. He paints pictures and sells them at the park. He charges $60 for a large painting and $30 for a small painting. Last month he sold eight large paintings and four small paintings. If he sold twice as much this month, how much is his sales for this month?</td>
<td>multiply(60, 8); multiply(30, 4); add(#0, #1); multiply(#2, 2)</td>
<td>multiply(60, 2); multiply(#0, 8); multiply(30, 2); multiply(#2, 4); add(#1, #3)</td>
<td>multiply(30, 4); add(480, #0); multiply(#2, 2)</td>
</tr>
</tbody>
</table>

**Example.** Considering the question in Table 2, an ideal arithmetic expression (e₁) first calculates the sales of large paintings for the previous month (multiply(60, 8)) and the sales of small paintings (multiply(30, 4)), adds the two results together (add(#0, #1)) and multiplies the sum by two to get this month’s sales (multiply(#2, 2)). However, the actual arithmetic expression in the original training data may be like \( e₂ \) which follows a complicated procedure and might be challenging and counter-intuitive to understand, or like \( e₃ \) which compresses certain steps. Such expressions can be computed to obtain the correct answer but may confuse the small model during training. Therefore, both \( e₂ \) and \( e₃ \) are expected to be revised to \( e₁ \).

**Procedure.** We consider the training samples that the small model can learn quickly with high confidence to be advantageous. We initially identify the training samples that the small model finds confusing. Specifically, as the model is trained to a preset number of steps, we are motivated by Shen and Sanghavi, 2019 to keep track of the cross-entropy loss (averaged over all the tokens) for each training sample. \( \mathcal{D}^c \) is formed by sifting out the top \( \lambda \) percentage of training samples in \( \mathcal{D} \) that have the highest loss, where \( \lambda \) is a hyperparameter.

Benefiting from the outstanding capabilities of CoT and few-shot learning in LLM, we employ the LLM to refine all the arithmetic expressions in \( \mathcal{D}^c \) by re-parsing the questions into arithmetic expressions in a CoT-based few-shot way. Specifi-
Prompts for refining expressions in Stage 1

**Question:** Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May? 

**Explanation:** 1. We first need to find out how many clips Natalia sold in May. We know that she sold half as many clips in May as she did in April, so we can divide the number of clips sold in April (48) by 2. The arithmetic expression for this step is: \#0 = divide(48, 2). 

2. Next, we need to find the total number of clips sold by adding the number of clips sold in April (48) to the number of clips sold in May (\#0). The arithmetic expression for this step is: \#0 = divide(48, 2); \#1 = add(48, \#0).

**Arithmetic expression:** \#0 = divide(48, 2); \#1 = add(48, \#0) ...

**Example.** Considering the question in Table 4, the small model for an extra sifting.

**Procedure.** We write eight demonstration examples manually and employ few-shot prompting or removing some of the descriptions from the original question in \(D^c\). LLM sifts out the questions in \(D^c\) and parses them into the small model for an extra sifting. It is important to note that, due to the LLM’s hallucination (Bang et al., 2023), we do not immediately trust the expressions provided by the LLM. Rather, we will later (at the end of Stage 2) give them to the small model for an extra sifting.

### 3.3 Stage 2: Targeted Question Complexity

In this stage, question complexity is targeted. The LLM sifts out the questions in \(D^c\) that are too complex for the LLM and hence also for the small model, denoted by \(D^f \subseteq D^c\), refines to make these questions less challenging, and parses them into arithmetic expressions to form \(D^f\).

**Example.** Considering the question in Table 4, the small model may get confused by the overly complex original question in \(D^c\). LLM can reduce the complexity of the original question by modifying or removing some of the descriptions from the original question, resulting in a simpler question with a shorter arithmetic expression.

**Procedure.** We write eight demonstration examples manually and employ few-shot prompting (prompts are shown in Table 3) to instruct the LLM to simplify these questions so that the complexity of the simplified questions may better fit the small model’s capacities. Specifically, the LLM recognizes the questions that do not produce any valid arithmetic expressions in Stage 1 and sifts them out into \(D^f \subseteq D^c\); such unsuccessful attempts in the previous stage may indicate complex questions. Each question in \(D^f\) is refined by the LLM into three simplified variants. The LLM then parses these simplified questions into arithmetic expressions, using the same prompting method as described in Section 3.2. The LLM attempts to parse each simplified question ten times and selects an arithmetic expression by majority voting with a minimum of three votes. Finally we collect these simplified questions and their corresponding arithmetic expressions to form \(D^f\).

<table>
<thead>
<tr>
<th>Original question</th>
<th>Simplified question</th>
<th>Arithmetic expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>James creates a media empire. He creates a movie for $2000. Each DVD cost $6 to make. He sells it for 2.5 times that much. He sells 500 movies a day for 5 days a week. How much profit does he make in 20 weeks?</td>
<td>James creates a movie for $2000. Each DVD cost $6 to make. He sells it for 2.5 times that much. How much profit does he make if he sells 500 movies?</td>
<td>multiply(6, 2.5); subtract(#0, 6); multiply(#1, 500); multiply(#2, 5); multiply(#3, 20); subtract(#4, 2000)</td>
</tr>
</tbody>
</table>

Table 4: Original and simplified questions.
So far, we have $\hat{D}^c$ and $\hat{D}^t$ which have gone through two stages of sifting-refining. To reduce the errors caused by the hallucination of LLM, the small model will again sift all the refined data. Specifically, in a manner similar to that described in Section 3.2, we track the loss of each sample in $\hat{D}^t$ and $\hat{D}^c$ when training on $\bar{D}^c \cup \bar{D}^t \cup (D/D^c) \cup D^t$. We set a hyperparameter $\alpha$ to specify the proportion of samples in $\hat{D}^c$ and $\hat{D}^t$ with the lowest loss that will be retained. We denote the $\alpha$-percentage of data retained from $\hat{D}^t$ and $\hat{D}^c$ by $\hat{D}^{ts} \subseteq \hat{D}^t$ and $\hat{D}^{cs} \subseteq \hat{D}^c$, respectively. The new training set we obtain after this last sifting is denoted by $\bar{D} = \hat{D}^{ts} \cup \hat{D}^{cs} \cup (D/D^c) \cup D^t$. We purposefully keep the complex questions $\hat{D}^c$ to improve the diversity of training data and hence the model’s robustness. We will finally train our model on $\bar{D}$.

4 Experiments

4.1 Datasets and Evaluation Metrics

We conducted experiments on six MWP datasets: GSM8k (Cobbe et al., 2021), MultiArith (Roy and Roth, 2015), ASDiv (Miao et al., 2020), SVAMP (Patel et al., 2021), SingleOp (Roy and Roth, 2018), and AddSub (Koncel-Kedziorski et al., 2016). We followed the train-valid-test splits of these datasets in Lila (Mishra et al., 2022). We used GSM8k in an in-distribution (ID) setting, i.e., our model was trained only on the training set of GSM8k, and the other datasets were considered as out-of-distribution (OOD) data for validation and testing due to their small sizes. For each OOD dataset, we added its training set to the test set to improve the reliability of the experimental results.

We measured accuracy, that is, the proportion of questions for which an arithmetic expression resulting in the correct answer was generated.

4.2 Backbone Models

We used four generative models as our small models to solve MWP by predicting an arithmetic expression, including:

- two T5 models (Raffel et al., 2020), T5-base (250M) and T5-large (780M), and
- two instruction-tuned Flan-T5 models (Chung et al., 2022), Flan-T5-base (250M) and Flan-T5-large (780M).

The small models all used greedy decoding in the prediction phase. We used GPT-3.5-turbo via OpenAI APIs as our LLM.

4.3 Baselines

We compared with four state-of-the-art fine-tuned small models: Flan-T5-large (Khalifa et al., 2023), LLaMA-7B (Khalifa et al., 2023), CodeT5 (Zhu et al., 2023), and LLaMA-7B (Hu et al., 2023).

We also compared with five knowledge distillation methods: Flan-T5-11B (Fu et al., 2023), T5-11B (Magister et al., 2022), GPT-2 (Shridhar et al., 2023), GPT-3-6.7B (Ho et al., 2023), and CodeT5 (Zhu et al., 2023).

Moreover, we fine-tuned our backbone model with CoTs containing arithmetic expressions as another knowledge distillation baseline to be compared. CoTs were obtained from the LLM by submitting each question ten times and retaining the returned CoTs containing an arithmetic expression resulting in the correct answer.

4.4 Implementation Details

We experimented on NVIDIA V100 (32GB) GPUs and Ubuntu 18.04. Our implementations were based on PyTorch 1.13.1 and HuggingFace Transformers 4.28.1. Note that GSM8k did not provide an official validation set, so we used the validation set of the OOD datasets to select the model. We set learning rate = $3e-5$ selected from $\{3e-5, 5e-5, 9e-5\}$, batch size = 27 selected from $\{9, 18, 27\}$, random seed = 8, maximum sequence length = 192, epoch = 300, $\lambda = 65\%$ selected from $\{20\%, 35\%, 50\%, 65\%, 80\%\}$ and $\alpha = 80\%$ selected from $\{60\%, 80\%, 100\%\}$.

4.5 Effectiveness of Targeted Training

As shown in Table 5, in both ID and OOD settings, we observed considerable improvements brought by targeted training, with increases ranging from 4.52 to 56.13 points. Larger improvements were observed on smaller models, with an average improvement of 25.39 points for T5-base, 18.57 points for T5-large, 20.74 points for Flan-T5-base, and 12.71 points for Flan-T5-large.

4.6 Comparison with Baselines

As shown in Table 6, our Flan-T5-large model with targeted training performed better than all the fine-tuned small models on GSM8k (46.79 points for Flan-T5-large vs. 45.97 points for Flan-T5-base).
Table 5: Effectiveness of targeted training.

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params.</th>
<th>GSM8k</th>
<th>MultiArith</th>
<th>ASDiv</th>
<th>SVAMP</th>
<th>SingleOp</th>
<th>AddSub</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-base</td>
<td>250M</td>
<td>22.90</td>
<td>68.10</td>
<td>18.12</td>
<td>14.16</td>
<td>13.44</td>
<td>10.27</td>
</tr>
<tr>
<td>w/ targeted training</td>
<td>250M</td>
<td>35.50</td>
<td>80.17</td>
<td>26.07</td>
<td>30.83</td>
<td>36.57</td>
<td>36.30</td>
</tr>
<tr>
<td>T5-large</td>
<td>780M</td>
<td>27.56</td>
<td>77.80</td>
<td>30.75</td>
<td>21.43</td>
<td>35.61</td>
<td>31.51</td>
</tr>
<tr>
<td>w/ targeted training</td>
<td>780M</td>
<td>40.99</td>
<td>86.85</td>
<td>53.73</td>
<td>36.59</td>
<td>74.06</td>
<td>43.84</td>
</tr>
<tr>
<td>Flan-T5-base w/ target</td>
<td>250M</td>
<td>27.10</td>
<td>79.96</td>
<td>23.35</td>
<td>16.79</td>
<td>21.93</td>
<td>15.07</td>
</tr>
<tr>
<td>Flan-T5-large w/ target</td>
<td>780M</td>
<td>32.06</td>
<td>80.52</td>
<td>50.00</td>
<td>36.59</td>
<td>74.06</td>
<td>55.14</td>
</tr>
</tbody>
</table>

Fine-Tuned Small Models

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params.</th>
<th>GISMBk</th>
<th>MultiArith</th>
<th>ASDiv</th>
<th>SVAMP</th>
<th>SingleOp</th>
<th>AddSub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flan-T5-large (Khalifa et al., 2023)</td>
<td>780M</td>
<td>36.30</td>
<td>-</td>
<td>-</td>
<td>68.60*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LLaMA (Khalifa et al., 2023)</td>
<td>7B</td>
<td>94.60*</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CodeT5 (Zhu et al., 2023)</td>
<td>770M</td>
<td>39.10</td>
<td>-</td>
<td>51.20</td>
<td>48.20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LLaM (Hu et al., 2023)</td>
<td>7B</td>
<td>21.90</td>
<td>88.30*</td>
<td>-</td>
<td>44.50*</td>
<td>-</td>
<td>78.50*</td>
</tr>
</tbody>
</table>

Knowledge Distillation from LLMs

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params.</th>
<th>GISMBk</th>
<th>MultiArith</th>
<th>ASDiv</th>
<th>SVAMP</th>
<th>SingleOp</th>
<th>AddSub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flan-T5 (Fu et al., 2023)</td>
<td>11B</td>
<td>27.10</td>
<td>63.00</td>
<td>37.60</td>
<td>35.60</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T5 (Magister et al., 2022)</td>
<td>11B</td>
<td>21.90</td>
<td>-</td>
<td>42.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GPT-2 (Shridhar et al., 2023)</td>
<td>774M</td>
<td>21.08</td>
<td>-</td>
<td>-</td>
<td>23.60</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GPT-3 (Ho et al., 2023)</td>
<td>6.7B</td>
<td>33.33*</td>
<td>-</td>
<td>12.67</td>
<td>-</td>
<td>-</td>
<td>21.01*</td>
</tr>
<tr>
<td>CodeT5 (Zhu et al., 2023), Flan-T5, fine-tuned w/ CoTs</td>
<td>774M</td>
<td>39.20</td>
<td>79.20</td>
<td>51.20</td>
<td>48.20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Flan-T5-large, fine-tuned w/ CoTs</td>
<td>780M</td>
<td>30.30</td>
<td>90.52</td>
<td>50.27</td>
<td>44.49</td>
<td>64.86</td>
<td>51.03</td>
</tr>
</tbody>
</table>

Our Targeted Training Approach

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params.</th>
<th>GISMBk</th>
<th>MultiArith</th>
<th>ASDiv</th>
<th>SVAMP</th>
<th>SingleOp</th>
<th>AddSub</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-base w/ targeted training</td>
<td>250M</td>
<td>35.50</td>
<td>80.17</td>
<td>30.83</td>
<td>36.57</td>
<td>36.30</td>
<td>-</td>
</tr>
<tr>
<td>T5-large w/ targeted training</td>
<td>780M</td>
<td>40.99</td>
<td>86.85</td>
<td>53.73</td>
<td>36.59</td>
<td>74.06</td>
<td>43.84</td>
</tr>
<tr>
<td>Flan-T5-base w/ targeted training</td>
<td>250M</td>
<td>37.25</td>
<td>84.48</td>
<td>48.45</td>
<td>36.59</td>
<td>74.06</td>
<td>55.14</td>
</tr>
<tr>
<td>Flan-T5-large w/ targeted training</td>
<td>780M</td>
<td>46.79</td>
<td>95.04</td>
<td>58.10</td>
<td>81.37</td>
<td>81.37</td>
<td>55.14</td>
</tr>
</tbody>
</table>

Table 6: Comparison with baselines. The best and second best results are in bold and underlined, respectively. We use * to indicate the results obtained in an ID setting.

vs. 39.10), MultiArith (95.04 vs. 94.60), and AS-Div (58.10 vs. 51.20). However, it was not the best-performing method on SVAMP and AddSub, which was not surprising since the best baseline results were achieved in an ID setting.

Among the methods with knowledge distillation from the LLM, Fu et al., 2023, Magister et al., 2022, and Ho et al., 2023 fine-tuned small models with CoTs generated by the LLM. Shridhar et al., 2023 distilled the capabilities of LLM to small models through question decomposition and Zhu et al., 2023 distilled the code generation capability of LLM to train small model as a code generator. Our Flan-T5-large model with targeted training outperformed these methods on all the datasets except for SVAMP. In particular, our Flan-T5-base model (250M) with targeted training also outperformed most of these models (up to 11B).

Comparing the two variants of Flan-T5-large, our targeted training largely exceeded CoT-based fine-tuning on GSM8k (46.79 vs. 30.30), while their difference was relatively small on the simple MultiArith dataset (95.04 vs. 90.52). It suggested that it might be difficult for small models to understand and exploit the CoTs for complex questions.

4.7 Ablation Study

To analyze the effectiveness of the two stages of targeted training, we conducted an ablation study by disabling each stage. As shown in Table 7, we noticed a substantial performance drop (12.13–27.74 points) on ASDiv, SVAMP, SingleOp, and AddSub when Stage 2 was disabled. On GSM8k and Multi-Arith, a major performance decrease (5.17–13.74) was seen when Stage 1 was further disabled. The two stages showed complementary usefulness on different datasets. Therefore, our model using both stages enabled small models to achieve substantial improvements in performance on all the datasets, exhibiting its robustness and generalizability.
### 4.8 Comparison with LLMs

From the literature we collected the results of well-known LLMs on MWP datasets to help position our results achieved with a small model. Depending on whether CoT was used and whether demonstration examples were provided (i.e., zero-shot or few-shot), we divided existing LLM-based methods into three categories as shown in Table 8.

On GSM8k, our model outperformed CoT-based zero-shot PaLM (540B), despite PaLM having ~700 times more model parameters. Our model also exceeded GPT-3.5, Codex, LaMDA, and PaLM in the standard few-shot setting by 27.09–40.29 points. However, our model generally lagged behind CoT-based few-shot LLMs like GPT-4. On MultiArith, our model outperformed PaLM under all the settings and was only slightly behind (95.04 vs. 96.2) CoT-based few-shot Codex. On the AS-Div, SVAMP, SingleOp, and AddSub datasets, our model outperformed LaMDA under both the standard and the CoT-based few-shot settings, but it could not match PaLM under the few-shot setting.

### 4.9 Analysis of Sifting Ratio ($\lambda$ and $\alpha$)

We conducted experiments to examine the influence of various sifting ratios on the performance.

As shown in Figure 3, the accuracy generally increased with $\lambda$, with the exception of MultiArith where the performance fluctuated within a narrow range. On most datasets, accuracy did not change much when $\lambda \geq 50\%$, and relatively good results were often achieved when $\lambda = 65\%$. The situation was similar for $\alpha$, and the accuracy mostly peaked when $\alpha = 80\%$.  

### 4.10 Sizes of Sifted and Refined Training Data

Table 9 shows the size of training data processed in each stage.

The small model sifting out 65% of the raw training set ($|\hat{D}^{cs}|/|D|$) indicated a large gap between the raw training data and the high-quality and friendly data that fits the small model’s capacities, showing the necessity of targeted training.

We observed that $|\hat{D}^{cs}| = 5,934$ exceeded 5,361.
Figure 3: Analysis of sifting ratio. We reported accuracy on the validation set, except for GSM8k on the test set.

$|D_c| = 4,621$, indicating that for some questions, the LLM returned multiple different expressions in Stage 1. As a result, during training, the small model could see different solutions to a question, thus alleviating over-fitting and enhancing the small model’s generalizability.

The percentage of questions that the LLM could correctly answer in at least one of ten attempts was $((|D_c| - |D_t|)/|D_c|) = (4,621 - 1,062)/4,621 = 77\%$. This considerably high accuracy suggested the LLM’s potential to solve complex MWPs such as those in GSM8k.

$\hat{D}_{ts}$ only took a small fraction (8.5%) of the final training data ($|\hat{D}_{ts}|/|\hat{D}|$) but brought an improvement of 12.13–27.74 points on ASDiv, SVAMP, SingleOp, and AddSub in Stage 2 as shown in Table 7, possibly because it helped extend question distribution in the training data.

### 4.11 Error Analysis

In order to identify the causes of model failures, we randomly selected 50 questions from the test set of GSM8k and the validation set of other datasets. As shown in Table 10, we categorized the identified errors into the following groups:

1. **Operator error**: Although the model predicted the correct operands, the operators were incorrect.
2. **Operand error**: Operators were correct while operands were incorrect.
3. **#steps error**: The model predicted an expression having a similar structure to the gold expression but missing some steps or adding unnecessary steps.
4. **Misunderstanding error**: The model provided an expression that was entirely unrelated to the gold expression.
5. **Syntax error**: The expression structure did not adhere to the DSL definition.

On GSM8k and other datasets, we observed a similar distribution of error categories, with a percentage of 50% and 56%, respectively, belonging to the first three types of errors which are relatively minor errors and would be the focus of our future work.

### Table 10: Error Analysis.

<table>
<thead>
<tr>
<th>Source of error</th>
<th>GSM8k</th>
<th>Other Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator error</td>
<td>12%</td>
<td>20%</td>
</tr>
<tr>
<td>Operand error</td>
<td>10%</td>
<td>6%</td>
</tr>
<tr>
<td>#steps error</td>
<td>28%</td>
<td>30%</td>
</tr>
<tr>
<td>Misunderstanding error</td>
<td>50%</td>
<td>40%</td>
</tr>
<tr>
<td>Syntax error</td>
<td>0%</td>
<td>4%</td>
</tr>
</tbody>
</table>

5 Conclusion

We proposed a targeted training approach to solving MWPs. With the help of LLM, our small model goes through two stages of proactive sifting and refinement of reasoning steps and question complexity to tailor training data to meet small model’s capacities. We have demonstrated the effectiveness of targeted training on multiple MWP datasets.

In the future, we will consider other possibilities of refining training data. Moreover, we believe that our targeted training represents a general strategy and has the potential to be extended to other tasks beyond MWPs, such as logical reasoning (Clark et al., 2021; Dalvi et al., 2021) and reasoning over tabular data (Chen et al., 2021; Zhu et al., 2021).
Limitations

We were unable to use a stronger LLM like GPT-4 for cost and API restriction reasons, while such a stronger model might bring a greater improvement to the performance of our small models. Our current method of using sorted losses to sift out the training data that confuses the small model is straightforward and can be polished. In our future work, we will explore better sifting strategies.

References


Implementations of the chain of thought (CoT) reasoning in language models. The four operations (Chen et al., 2021) include addition (+), subtraction (−), multiplication (×), and division (÷) that are described in natural language. The authors of Lila (Mishra et al., 2022) used customized scripts to parse the GSM8k raw output into Python code. Lila is a benchmark that includes various datasets, and the authors stated in their paper that some of the datasets used custom scripts to convert raw output to Python code and others were manually annotated. We confirmed via email with Lila’s authors that for GSM8k, Python code was generated automatically by scripts. We further transformed Lila’s Python code into a domain-specific language (DSL) representation of an arithmetic expression.

Our DSL is similar to that used in other works (Chen et al., 2021). The four operations for addition (add), subtraction (subtract), multiplication (multiply), and division (divide) are included in the DSL that we defined. Each operator is binary, and we use #0, #1, etc. to indicate the intermediate results of calculations in previous steps. A semicolon (“;”) separates steps. One benefit of utilizing the DSL is that the computational process can be represented formally to be generated by models. Once such a representation has been obtained, an answer can be calculated using a calculator, which avoids relying on the model to compute numerical results which is error-prone.

A.2 Prompts in Stage 1

Prompts are shown in Table 12 and Table 13.
Question: Noah is a painter. He paints pictures and sells them at the park. He charges $60 for a large painting and $30 for a small painting. Last month he sold eight large paintings and four small paintings. If he sold twice as much this month, how much is his sales for this month?

GSM8k raw output (natural language): Noah earned $60/large painting x 8 large paintings = $480 for the large paintings. He also earned $30/small painting x 4 small paintings = $120 for the small paintings. So, his total sales last month were $480 + $120 = $600. If he sold twice as much this month, then his sales would be $1200.

Lila’s processed expression (Python): a=60; b=8; c=30; d=4; e=c*a+b*d; answer=e*2; print(answer)

Our processed expression (DSL): multiply(60, 8); multiply(30, 4); add(#0, #1); multiply(#2, 2)

Question: Each bird eats 12 beetles per day, each snake eats 3 birds per day, and each jaguar eats 5 snakes per day. If there are 6 jaguars in a forest, how many beetles are eaten each day?

GSM8k raw output (natural language): First find the total number of snakes eaten: 5 snakes/jaguar x 6 jaguars = 30 snakes. Then find the total number of birds eaten per day: 30 snakes x 3 birds/snake = 90 birds. Then multiply the number of snakes by the number of beetles per snake to find the total number of beetles eaten per day: 90 snakes x 12 beetles/snake = 1080 beetles.

Lila’s processed expression (Python): a=5; b=6; c=a*3; d=b*12; e=c+d; answer=e*1080; print(answer)

Our processed expression (DSL): multiply(5, 6); multiply(#0, 3); multiply(#1, 12)

Question: Rachel and Sara want to attend a beauty and modeling contest. They both want to buy new pairs of shoes and dresses. Sara buys a pair of shoes which costs $50 and a dress which costs $200. How much should Rachel budget if she wants to spend twice as much as what Sara spent on the pair of shoes and dress?

GSM8k raw output (natural language): The cost Rachel should budget for her pair of shoes is $50 * 2 = $100. The cost Rachel should budget for her dress is $200 * 2 = $400. So, the total Rachel should budget is $100 + $400 = $500.

Lila’s processed expression (Python): a=50; b=200; c=a*b; d=100+400; e=c+d; answer=e*2; print(answer)

Our processed expression (DSL): multiply(50, 2); multiply(200, 2); add(#0, #1)

Table 11: GSM8k raw output, Lila’s processed expression, and our processed expression.

A.3 Prompts in Stage 2

Prompts are shown in Table 14.
Explanation:

1. We first need to find out how many minutes Emily's hair is. We do this by subtracting the number of minutes Logan's hair is (20) from Emily's. The arithmetic expression for this step is: #1=subtract(50,20).

2. Finally, we can calculate how many inches of Emily's hair is by calculating how long Emily's hair is by dividing by 6 (as Emily's hair is 6 inches longer than Logan's). The arithmetic expression for this step is: #0=divide(#1,6).

Arithmetic expression:

#0=divide(#1,6); #1=subtract(50,20).

Explanation:

1. We first need to find out how many minutes Emily's hair is. We do this by subtracting the number of minutes Logan's hair is (20) from Emily's. The arithmetic expression for this step is: #1=subtract(50,20).

2. Finally, we can calculate how many inches of Emily's hair is by calculating how long Emily's hair is by dividing by 6 (as Emily's hair is 6 inches longer than Logan's). The arithmetic expression for this step is: #0=divide(#1,6).

Arithmetic expression:

#0=divide(#1,6); #1=subtract(50,20).

Question: Brandon sold 86 geckos last year. He sold twice that many the year before. How many geckos has Brandon sold in the last two years?

Explanation:

1. We first need to find out how many geckos Brandon sold the year before. We do this by multiplying the number of geckos sold last year (86) by 2. The arithmetic expression for this step is: #0=multiply(86,2).

2. Next, we can find the total number of geckos sold in the last two years by adding the number of geckos sold last year (86) to the number of geckos sold the year before (86 × 2). The arithmetic expression for this step is: #1=add(#0,#0).

Arithmetic expression:

#0=multiply(86,2); #1=add(#0,#0).

Explanation:

1. We first need to find out how many minutes Emily's hair is. We do this by subtracting the number of minutes Logan's hair is (20) from Emily's. The arithmetic expression for this step is: #1=subtract(50,20).

2. Finally, we can calculate how many inches of Emily's hair is by calculating how long Emily's hair is by dividing by 6 (as Emily's hair is 6 inches longer than Logan's). The arithmetic expression for this step is: #0=divide(#1,6).

Arithmetic expression:

#0=divide(#1,6); #1=subtract(50,20).

Question: Lisa is looking to attempt a World Record. She has decided to try and match Joey Chestnut’s record of eating 75 full hotdogs, buns included, in 10 minutes. Halfway through the time Lisa has eaten 20 hotdogs. How many hotdogs will she have to eat per minute to at least tie Joey Chestnut’s record?

Explanation:

1. Since Lisa has eaten 20 hotdogs halfway through the 10-minute time limit, we know that there are only 5 minutes left for her to eat the remaining hotdogs. To find out the number of minutes Lisa has left to eat hotdogs, we divide the total time limit of 10 minutes by 2, which gives us 5 minutes. The arithmetic expression for this step is: #0=divide(10,2).

2. Next, we need to calculate how many hotdogs Lisa still needs to eat to at least tie Joey Chestnut’s record. We do this by subtracting the number of hotdogs Lisa has already eaten (20) from Joey Chestnut’s record of 75. The arithmetic expression for this step is: #1=subtract(75,20).

3. Finally, we can find out how many hotdogs Lisa needs to eat per minute during the remaining 5 minutes to at least tie Joey Chestnut’s record. We divide the number of hotdogs that Lisa still needs to eat (#1) by the number of minutes left for her to eat them (#0). The arithmetic expression for this step is: #2=divide(#1,#0).

Arithmetic expression:

#0=divide(10,2); #1=subtract(75,20); #2=divide(#1,#0).

Explanation:

1. We first need to find out how many minutes Emily's hair is. We do this by subtracting the number of minutes Logan's hair is (20) from Emily's. The arithmetic expression for this step is: #1=subtract(50,20).

2. Finally, we can calculate how many inches of Emily's hair is by calculating how long Emily's hair is by dividing by 6 (as Emily's hair is 6 inches longer than Logan's). The arithmetic expression for this step is: #0=divide(#1,6).

Arithmetic expression:

#0=divide(#1,6); #1=subtract(50,20).

Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Explanation:

1. First, we need to find out how many clips Natalia sold in May. We do this by multiplying the number of clips sold in April (48) by 2. The arithmetic expression for this step is: #0=multiply(48,2).

2. Next, we need to find the total number of clips sold by adding the number of clips sold in April (48) to the number of clips sold in May (#0). The arithmetic expression for this step is: #0=multiply(48,2); #1=add(#0,#0).

Arithmetic expression:

#0=multiply(48,2); #1=add(#0,#0).

Explanation:

1. We first need to find out the length of Emily’s hair by adding 6 inches to Logan’s hair length (20). The arithmetic expression for this step is: #0=add(20,6).

2. Next, we need to find out how long Kate’s hair is by dividing the length of Emily’s hair (#0) by 2 because Kate’s hair is half as long as Emily’s hair. The arithmetic expression for this step is: #1=divide(#0,2).

Arithmetic expression:

#0=add(20,6); #1=divide(#0,2).

Question: On Blacks, Martha goes to the mall to buy clothes on sale. For every 2 jackets she buys, she gets 1 free jacket. For every 3 t-shirts she buys, she gets 1 free t-shirt. Martha decides to buy 4 jackets and 9 t-shirts. How many clothes will she take home?

Explanation:

1. First, we need to find out how many jackets Martha will get for buying 4 jackets. Since for every 2 jackets bought, she gets 1 free jacket, we can divide the number of jackets (4) by 2 to get the number of free jackets. The arithmetic expression for this step is: #0=divide(4,2).

2. Using the result from #0, we can now add the number of free jackets (#0) to the number of jackets (4) Martha bought to get the total number of jackets she will take home. The arithmetic expression for this step is: #0=divide(4,2); #1=add(#0,4).

3. Next, we need to calculate how many free t-shirts Martha will get for buying 9 t-shirts. Since for every 3 t-shirts she buys, she gets 1 free t-shirt, we can divide the number of t-shirts (9) she bought by 3 to get the number of free t-shirts. The arithmetic expression for this step is: #2=divide(9,3).

4. Using the result from #2, we can now add the number of free t-shirts (#2) to the number of t-shirts (9) Martha bought to get the total number of t-shirts she will take home. The arithmetic expression for this step is: #3=add(#2,9).

5. Finally, we can add the total number of jackets and t-shirts Martha will take home to get the total number of clothes she will take home. The arithmetic expression for this step is: #4=add(#1,#3).

Arithmetic expression:

#0=divide(4,2); #1=add(#0,4); #2=divide(9,3); #3=add(#2,9); #4=add(#1,#3).

Question: Kate’s hair is half as long as Emily’s hair. Emily’s hair is 6 inches longer than Logan’s hair. If Logan hair is 20 inches, how many inches is Kate’s hair?

Explanation:

1. We first need to find out the length of Emily’s hair by adding 6 inches to Logan’s hair length (20). The arithmetic expression for this step is: #0=add(20,6).

2. Next, we need to find out how long Kate’s hair is by dividing the length of Emily’s hair (#0) by 2 because Kate’s hair is half as long as Emily’s hair. The arithmetic expression for this step is: #1=divide(#0,2).

Arithmetic expression:

#0=add(20,6); #1=divide(#0,2).
Question: Ron ate pizza with his friends the other day. If they ordered a 12-slice pizza and each of them ate 4 slices, how many friends were there with Ron?

Explanation: 1. We start with the total number of slices in the pizza (12) and divide it by the number of slices each person ate (4) to find the total number of people who ate pizza. The arithmetic expression for this step is: \( #0=\text{divide}(12, 4) \). 2. But the question asks for the number of friends with Ron, so we need to subtract Ron himself from the total number of people who ate pizza. We can subtract 1 from the total we got in \( #0 \). The final arithmetic expression is: \( #1=\text{subtract}(#0, 1) \).

Arithmetic expression: \( #0=\text{divide}(12, 4); #1=\text{subtract}(#0, 1) \).

Question: Jerry’s two daughters play softball on different teams. They each have 8 games this season. Each team practices 4 hours for every game they play. If each game lasts for 2 hours, how many hours will Jerry spend at the field watching his daughters play and practice altogether?

Explanation: 1. We first need to find out the total number of practice hours for one of Jerry’s daughters by multiplying the number of games each team has (8) by the practice hours per game (4). The arithmetic expression for this step is: \( #0=\text{multiply}(8, 4) \). 2. Then, we can find out the total number of game hours for one of Jerry’s daughters by multiplying the number of games each team has (8) by the game hours per game (2). The arithmetic expression for this step is: \( #1=\text{multiply}(8, 2) \). 3. Then, we can get the total number of hours Jerry will spend watching one of his daughters by adding the total practice and game hours together. The arithmetic expression for this step is: \( #2=\text{add}(#0, #1) \). 4. Finally, we note that Jerry has two (2) daughters play softball on different teams, so we need multiply the result from last step (#2) with 2: \( #3=\text{multiply}(#2, 2) \) to get the final result.

Arithmetic expression: \( #0=\text{multiply}(8, 4); #1=\text{multiply}(8, 2); #2=\text{add}(#0, #1); #3=\text{multiply}(#2, 2) \).

Question: {{ QUESTION }}

Explanation:

Table 13: Few-shot prompts for revising reasoning steps in Stage 1 (continued from Table 12).
Original complex question: Olaf collects colorful toy cars. At first, his collection consisted of 150 cars. His family, knowing his hobby, decided to give him some toy cars. Grandpa gave Olaf twice as many toy cars as the uncle. Dad gave Olaf 10 toy cars, 5 less than Mum. Auntie gave Olaf 6 toy cars, 1 more than the uncle. How many toy cars does Olaf have in total, after receiving all these gifts?

Rewrite original complex question into a simplified version: Olaf collects colorful toy cars. At first, his collection included 150 cars. His family knew about his hobby and decided to give him some toy cars. Grandpa gave Olaf twice as many toy cars as his uncle. Uncle gave him 10 toy cars. After receiving all these gifts, how many toy cars does Olaf have in total?

Original complex question: Mr. Finnegan has 3 tanks with a capacity of 7000 gallons, 5000 gallons, and 3000 gallons, respectively. If he fills the first tank up to 3/4 full, the second tank with water up to 4/5 of its capacity, and the third tank up to half of its capacity, how many gallons in total are in the tanks?

Rewrite original complex question into a simplified version: Mr. Finnegan has 2 tanks with a capacity of 7000 gallons, 5000 gallons, respectively. If he fills the first tank up to 3/4 full, the second tank with water up to 4/5 of its capacity, how many gallons in total are in the tanks?

Original complex question: Denmark wants to order pizza. For toppings, he has 3 cheese, 4 meat and 5 vegetable options, one of which is peppers. He can have only one selection from each topping category (one cheese, one meat and one vegetable). However, if he chooses to have pepperoni, he cannot have peppers. How many topping combinations does he have total?

Rewrite original complex question into a simplified version: Denmark wants to order pizza. For toppings, he has 3 cheese, 4 meat and 5 vegetable options. He can have only one selection from each topping category (one cheese, one meat and one vegetable). How many topping combinations does he have total?

Original complex question: 8 years ago James was twice Janet’s age. In 15 years James will turn 37. Susan was born when Janet turned 3. How old is Janet?

Rewrite original complex question into a simplified version: 8 years ago James was twice Janet’s age. In 15 years James will turn 37. How old is Janet?

Original complex question: There are three trees in the town square. The tallest tree is 150 feet tall. The middle height tree is 2/3 the height of the tallest tree. The shortest tree is half the size of the middle tree. How tall are all 4 buildings put together?

Rewrite original complex question into a simplified version: There are three trees in the town square. The tallest tree is 150 feet tall. The middle height tree is 20 feet shorter than the tallest tree. The shortest tree is 20 feet shorter than the tallest tree. How tall are all 4 buildings put together?

Original complex question: It’s Meghan’s turn to pick up her team’s coffee order. She needs 2 drip coffees that are $2.25 each and one double shot espresso that’s $3.50. She needs 2 lattes that are $4.00 and needs to add vanilla syrup to one of those for an additional $0.50. She also needs 2 cold brew coffees that are $2.50 each and 1 cappuccino for $3.50. How much is the coffee order?

Rewrite original complex question into a simplified version: It’s Meghan’s turn to pick up her team’s coffee order. She needs 2 drip coffees that are $2.25 each and one double shot espresso that’s $3.50. She needs 2 lattes that are $4.00 each. How much is the coffee order?

Original complex question: The tallest building in the world is 100 feet tall. If the second tallest is half that tall, and the third tallest is half as tall as the second, and the fourth is one-fifth as tall as the third, how tall are all 4 buildings put together?

Rewrite original complex question into a simplified version: The tallest building in the world is 100 feet tall. If the second tallest is half that tall, and the third tallest is half as tall as the second, how tall are all 4 buildings put together?

Original complex question: A factory decides to stop making cars and start making motorcycles instead. When it made cars, per month, it cost $100 for materials, they could make 4 cars, and they sold each car for $50. Now that they make motorcycles it costs $250 for materials, but they sell 8 of them for $50 each. How much more profit do they make per month selling motorcycles instead of cars?

Rewrite original complex question into a simplified version: A factory decides to start making motorcycles. Now they make motorcycles it costs $250 for materials, but they sell 8 of them for $50 each. How much more profit do they make per month selling motorcycles instead of cars?

Original complex question: [{ QUESTION }]

Rewrite original complex question into a simplified version:

Table 14: Few-shot prompts for reducing question complexity in Stage 2.