Distilling Reasoning Capabilities into Smaller Language Models

Kumar Shridhar* **Alessandro Stolfo*** **Mrinmaya Sachan**
Department of Computer Science, ETH Zürich
{shkumar, stolfoa}@ethz.ch

Abstract

Step-by-step reasoning approaches like chain of thought (CoT) have proved to be very effective in inducing reasoning capabilities in large language models. However, the success of the CoT approach is fundamentally tied to the model size, and billion parameter-scale models are often needed to get CoT to work. In this paper, we propose a knowledge distillation approach that leverages the step-by-step CoT reasoning capabilities of larger models and distills these abilities into smaller models.

In this work, we propose an alternative reasoning scheme, SOCRATIC COT that learns a decomposition of the original problem into a sequence of subproblems and uses it to guide the intermediate reasoning steps. We use SOCRATIC COT to train a combination of two small distilled models: a *problem decomposer* and a *subproblem solver*. In practice, given a new problem, the two distilled models work in sync to decompose and solve complex problems. On multiple reasoning datasets (GSM8K, StrategyQA, and SVAMP), our proposed distillation strategies boost the performance of smaller models over 70% compared to the baselines. Finally, we investigate when SOCRATIC COT is an effective alternative to CoT, demonstrating cases where a much smaller model (GPT-2 large) can outperform a 10X larger model (GPT-3 6B). Our code is available here.

1 Introduction

Large language models (LLMs) have demonstrated strong performance on a variety of reasoning tasks (Brown et al., 2020; Hoffmann et al., 2022; Chowdhery et al., 2022, *inter alia*). One particularly interesting strategy for prompting these models is chain-of-thought (CoT), which has been shown to elicit reasoning abilities in LLMs by asking the model to incorporate intermediate reasoning steps while solving a problem (Nye et al., 2021; Wei et al., 2022b; Wang et al., 2022). However, CoT has been shown to work primarily on models with hundreds of billions of parameters (Wei et al., 2022b,a) or those tuned on a wide range of tasks (Chung et al., 2022; Iyer et al., 2022).

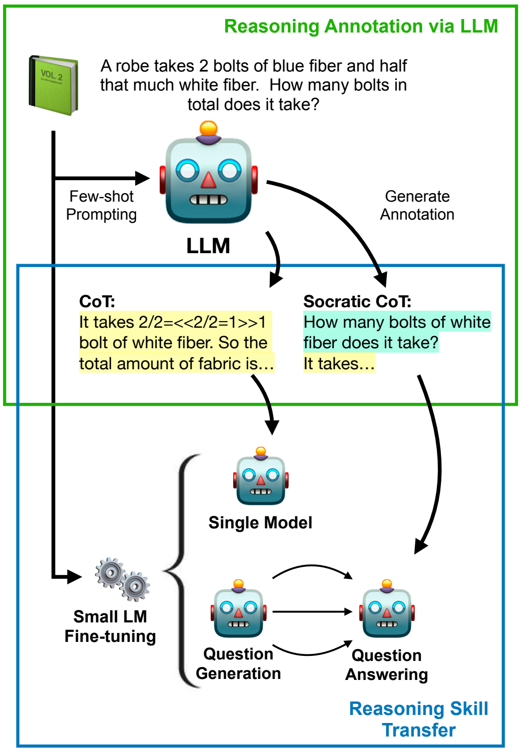

Reasoning Annotation via LLM
VOL 2
A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Few-shot Prompting
LLM
Generate Annotation
CoT:
It takes 2/2=<<2/2=1>>1 bolt of white fiber. So the total amount of fabric is…
Socratic CoT:
How many bolts of white fiber does it take?
It takes…
Single Model
Small LM Fine-tuning
Question Generation
Question Answering
Reasoning Skill Transfer

Figure 1: Illustration of the proposed framework. First, an LLM is prompted to decompose a multi-step problem providing annotation for the intermediate steps leading to the final solution. Then, the generated annotation is used to provide additional supervision when fine-tuning smaller models.

Due to the significant computational resources or expensive API calls required to access CoT-capable LLMs, we ask whether it is possible to elicit such reasoning capabilities in smaller models.[1]

* Equal contribution.

[1] Following Li et al. (2022), we argue that *small* and *large* models are relative terms and context-dependent. We consider models with billions of parameters to be large, and models with millions of parameters to be small.

Findings of the Association for Computational Linguistics: ACL 2023, pages 7059–7073
July 9-14, 2023 ©2023 Association for Computational Linguistics

Small-sized, non-fine-tuned language models are known to be poor reasoners (Stolfo et al., 2023). Therefore, a possible approach to induce CoT-like reasoning abilities in smaller models would be fine-tuning them on step-by-step examples.

In our work, we propose a framework for leveraging the reasoning capabilities of LLMs to supervise the training of smaller models. This approach can be thought of as a form of *knowledge distillation* (Hinton et al., 2015), where a larger teacher model transfers knowledge to a smaller student model. However, unlike standard knowledge distillation, our method transfers the reasoning abilities of the teacher model only using its generated solutions as a proxy, i.e., we do not assume access to the teacher model parameters. Our approach consists of prompting an LLM to produce step-by-step annotations leading to the answer for a set of problems. This annotation is then used as supervision to fine-tune the student model. A high-level illustration of the process is provided in Figure 1.

Within this framework, we study three different types of *annotation structure* for supervising our distillation approach: (i) We consider fine-tuning on the *gold* step-by-step solution procedure for datasets where the step-by-step solutions are available. (ii) We study whether procedural supervision, coming from the chain of thought (CoT) of the teacher model can improve upon the baseline. (iii) We propose a third type of supervision structure, which we call SOCRATIC COT. This approach relies on learning a semantic decomposition of the original problem into a sequence of subproblem-solution pairs using two models – a) a question generator that learns to decompose the problem into a sequence of subproblems, and b) a question-answering model that solves the various generated subproblems (more details are in section 3.2). This approach can be thought of as an extension of the typical chain of thought reasoning where, unlike CoT, the intermediate steps are now decomposed into subquestion-solution pairs; the subquestions guide the generation of intermediate steps that lead to the final answer to the problem.

We train distilled student models with the various annotation structures mentioned above. Depending on the annotation available for the given data, we use the teacher model to generate either a CoT-like solution to a problem or, if the step-by-step annotation is available, a set of subquestions leading to the solution of the problem, or both (examples of different annotations are shown in Figure 2).

We perform our analyses on three multi-step reasoning datasets: GSM8K (Cobbe et al., 2021), StrategyQA (Geva et al., 2021), and SVAMP (Patel et al., 2021). We consider data with various types of annotation to cover a range of realistic data scenarios. Our results show that supervision by CoT-decomposed examples helps smaller models perform better, and subquestioning introduced by SOCRATIC COT can provide further improvement. We observe performance gains of up to 40% with LLM-generated step-by-step annotations – this validates the effectiveness of our distillation framework (detailed analysis in Section 5).

2 Related Work

Decomposing Multi-Step Reasoning Tasks Solving multi-step reasoning tasks like MWPs has been a popular area of research for the last couple of years (Kushman et al., 2014; Hosseini et al., 2014; Roy et al., 2015; Amini et al., 2019; Zhang et al., 2020; Shridhar et al., 2022; Opedal et al., 2023). However, the majority of the modern approaches for these problems are shifting towards using large language models, often relying on approaches involving prompting or in-context learning (Cobbe et al., 2021; Kojima et al., 2022; Wei et al., 2022b; Chowdhery et al., 2022; Lewkowycz et al., 2022; Srivastava et al., 2022). One such prompting approach is the chain of thought prompting (Wei et al., 2022b), which prompts the language model to generate a series of intermediate steps that improve the reasoning capabilities in LLMs. Wang et al. (2022) took another step forward and sampled multiple reasoning paths and selected the most relevant output using majority voting. Huang et al. (2022) used the most voted outputs to further fine-tune the model for better performance. Kojima et al. (2022) further improved the reasoning of LLM in a zero-shot manner by appending "Let's think step by step" to the prompt. In contrast, our work does not propose prompting solutions; instead, we explicitly guide the student model reasoning using sub-questions at each step. Most similar to our work is the work by Zhou et al. (2022) which decomposes questions into sub-questions and asks the language model to solve each sub-question sequentially. However, this work is also restricted to prompting and only works with LLMs with billions of parameters.

Knowledge Distillation Our approach is reminiscent of knowledge distillation (Ba and Caruana, 2014; Hinton et al., 2015) in that we use a student network to mimic the large teacher language model. Snell et al. (2022) demonstrated the usefulness of providing instruction that can help models achieve better reasoning skills. Similar to our hypothesis, Eisenstein et al. (2022) argued that question-answering systems should focus not only on the final answer, but also on the rationale that justifies their reasoning, to help them reason better. We go beyond this; in our work, in addition to the question-answering system, we also focus on what questions need to be asked at each step that can help to learn that reasoning step better. Finally, similar to our hypothesis of injecting reasoning capabilities into smaller models, Li et al. (2022) used CoT-like reasoning from LLMs to train smaller models on a joint task of generating the solution and explaining the generated solution. We, on the other hand, use the LLM to generate subquestions and solution pairs and use them together to inject reasoning capabilities into smaller models.

Subquestioning as supervision The idea of inquiring or asking information-seeking questions for discovery learning has been studied well in the past (Bruner, 1961). Rao and Daumé III generated clarification questions based on Stack Exchange questions as supervision, Klein and Nabi (2019) used a joint question answering model to ask questions from a given span of text and later answer them, and (Rajani et al., 2019; Shwartz et al., 2020) asked questions to improve common sense QA models. In contrast, our work focuses on multistep reasoning tasks where intermediate clarifying questions and reasoning steps may not always be available and may need to be extracted from a teacher model.

3 Methodology

The setting we consider consists of a data set $\mathcal{D}$, where each problem P_i is accompanied by a final answer a_i that can be reached by several steps of reasoning. The task of solving the problem using a model ψ is to predict an answer $\hat{a} = \psi(P)$ such that $\hat{a} = a$. We consider different data scenarios where intermediate annotations of the solution may be available in different forms (e.g., step-by-step, as a semantic decomposition by subquestions) or may not be present. Depending on the availability of annotations, we propose different approaches to augment the training of a small model on $\mathcal{D}$ by using LLMs.

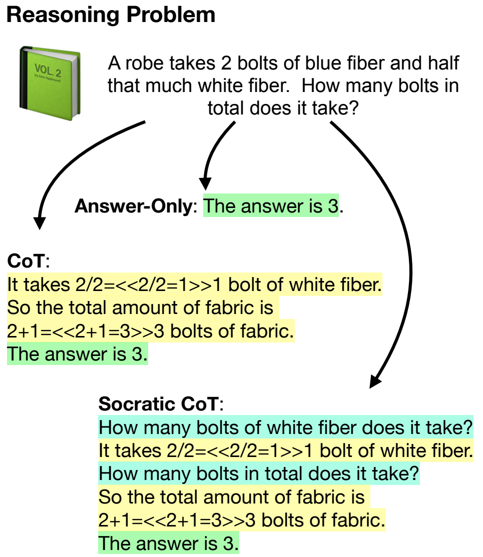

Reasoning Problem
VOL. 2
A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Answer-Only: The answer is 3.
CoT:
It takes 2/2=<<2/2=1>>1 bolt of white fiber.
So the total amount of fabric is
2+1=<<2+1=3>>3 bolts of fabric.
The answer is 3.
Socratic CoT:
How many bolts of white fiber does it take?
It takes 2/2=<<2/2=1>>1 bolt of white fiber.
How many bolts in total does it take?
So the total amount of fabric is
2+1=<<2+1=3>>3 bolts of fabric.
The answer is 3.

Figure 2: Illustration of the three different kinds of annotation structure. Our proposed approach, SOCRATIC COT, augments the typical chain-of-thought step-by-step solution with subquestioning.

3.1 Distilling step-by-step reasoning via CoT

A data set may present an annotation that contains intermediate reasoning steps that lead to the answer a_i (i.e., a chain-of-thought annotation). This intermediate annotation can be used directly to fine-tune a small model. However, in cases where such step-by-step information is not available, we use a LLM to generate the reasoning steps that might improve the performance of the small model.

To achieve this, we consider a small subset of the dataset $\mathcal{D}$ and decompose each problem P_i into n_i intermediate reasoning steps. We construct these intermediate reasoning steps manually, since we only need a few examples as prompts (examples are provided in Appendix Table 6).

For each remaining problem $P \in \mathcal{D}$, we then prompt a large language model $\mathcal{M}$ to generate the intermediate reasoning steps. We make sure that the chain of reasoning steps is meaningful by checking whether the last solution matches the ground truth answer, i.e. whether $a_i^{(n_i)} = a_i$, where $a_i^{(n_i)}$ represents the answer corresponding to the last reasoning step. If this is not the case, we discard the problem and sample a new chain by prompting the model again (for a maximum of 3 times). In this way, we obtain an augmented dataset $\mathcal{D}^*$ in which a subset of problems is paired with a sequence of reasoning steps leading to the correct result. Finally, we can distill the reasoning capabilities into smaller models by fine-tuning them with the generated intermediate steps.

3.2 Distilling step-by-step reasoning through Socratic CoT

In this section, we describe how CoT can be enhanced through subquestioning. An illustration of our approach is shown in Figure 3.

3.2.1 Extracting the Reasoning Capability from the Teacher

In Section 3.1, we detailed how an LLM can be used to generate the intermediate annotation of a problem P_i as a chain of steps leading to the answer a_i. We now extend this procedure to include a subquestion at each step of the solution. Following a similar procedure as described in Section 3.1, we prompt the LLM with few exemplars of problems decomposed as a set of intermediate subquestion-solution pairs (the prompts are reported in Appendix Table 6). This way, we obtain an intermediate annotation that includes subquestioning. In particular, each of the n_i steps constituting the overall solution is a subquestion-solution pair, denoted $q_i^{(j)}, s_i^{(j)}$, $j \in \{1, \dots, n_i\}$ (an example is shown in Figure 2). We refer to the ordered list of subquestion-solution pairs for problem P_i as $(q_i^{(1)}, s_i^{(1)}), \dots, (q_i^{(n_i)}, s_i^{(n_i)})$.

3.2.2 Transferring the Reasoning Capability into the Student

We present two strategies to distill the reasoning annotation provided by the LLM into smaller models.

In the first strategy, a single *unified* student is trained to generate the subquestion-solution pairs simultaneously, while in the second strategy, the question generation and question-answering tasks are assigned to two separate models. We call this second strategy *iterative* because the question-answering model is trained to solve each subquestion iteratively.

Unified. Using the problems in $\mathcal{D}$ that contain the chain of intermediate questions and solutions, we train a *unified* student model $\mathcal{M}_{uni}$ that learns to generate the sequence of subquestion-solution pairs $\{(q^{(1)}, s^{(1)}), (q^{(2)}, s^{(2)}), \dots\}$ that lead to the solution of a given problem. We use a pre-trained transformer-based model (Vaswani et al., 2017) and train it on the chain of subquestion-solution pairs for each problem P. Given a step j of problem P (i.e., the concatenation of $q^{(j)}$ and $s^{(j)}$) consisting of a sequence of m_j tokens $\{x_j^{(1)}, \dots, x_j^{(m_j)}\}$, we use a typical auto-regressive language modeling loss, $\mathcal{L}$:

$$\mathcal{L}_j(P) = \quad -\sum_{k=1}^{m_j} \log \mathbb{P}_{uni}\ (x_j^{(k)} | x_j^{:(k-1)}, P) \quad (1)$$

where $\mathbb{P}_{uni}(x|c)$ is the probability assigned by $\mathcal{M}_{uni}$ to token x given context c, and $x^{:(y)}$ indicates the sequence $\{x^{(1)}, \dots, x^{(y)}\}$. The loss $\mathcal{L}_j$ is computed for each problem P_i and for each pair $(q^{(j)}, s^{(j)})$ leading to the final answer a_i.

Iterative. The *iterative* version of the student separates the tasks of generating the subquestions and providing an intermediate answer to each subquestion into two distinct models: a question generation (QG) model and a question answering (QA) model. Both the QG and QA models are implemented using a Transformer-based language model (Vaswani et al., 2017). In particular, the QA model $\mathcal{M}_{qa}$ is iteratively trained to answer the teacher-generated sub-questions. The learning objective is computed at the token level for each intermediate solution:

$$\mathcal{L}(P, s^{(j)}) = -\sum_{k=1}^{l_j} \log \mathbb{P}_{\mathcal{QA}}\ (y_j^{(k)} | y_j^{:(k-1)}, q^{:(j)}, s^{:(j-1)}, P)$$

where l_j and the y_j's represent, respectively, the length and the tokens of the intermediate solution $s^{(j)}$. $s^{:(j-1)}$ consists of the previous solution generated by the QA model iteratively in the past iterations.

Similarly, the QG model is trained to acquire the ability of the teacher model to decompose the problem's main question into a series of sub-steps, each of which corresponds to a subquestion. The loss for this model is analogous to Equation 1, with the only difference being that the intermediate solutions are not considered for the QG model. During training, the previous intermediate solutions generated by the QA model are replaced with the teacher-generated solutions using teacher forcing (Cho et al., 2014). However, the intermediate solutions generated by the model are used at inference time.

3.3 Inference-time Predictions

Given an unseen problem P, the unified student model can directly predict a solution as a sequence

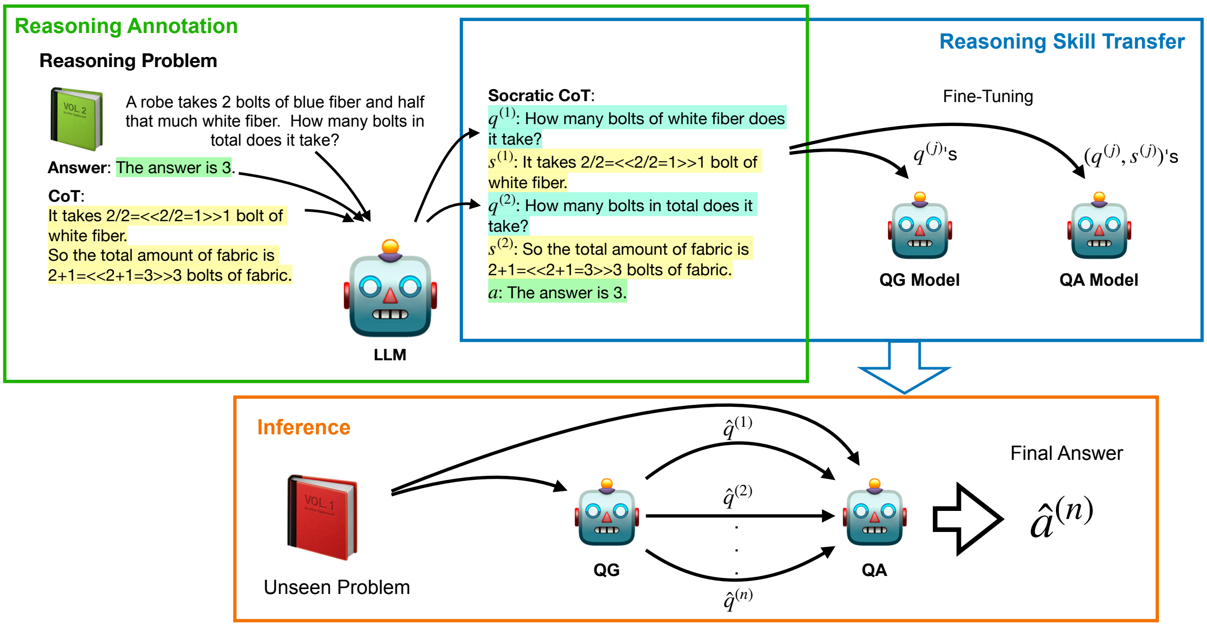

Reasoning Annotation
Reasoning Problem
VOL. 2
A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Answer: The answer is 3.
CoT:
It takes 2/2=<<2/2=1>>1 bolt of white fiber.
So the total amount of fabric is 2+1=<<2+1=3>>3 bolts of fabric.
LLM
Socratic CoT:
$q^{(1)}$: How many bolts of white fiber does it take?
$s^{(1)}$: It takes 2/2=<<2/2=1>>1 bolt of white fiber.
$q^{(2)}$: How many bolts in total does it take?
$s^{(2)}$: So the total amount of fabric is 2+1=<<2+1=3>>3 bolts of fabric.
a: The answer is 3.
Reasoning Skill Transfer
Fine-Tuning
$q^{(j)}$'s
$(q^{(j)}, s^{(j)})$'s
QG Model
QA Model
Inference
VOL. 1
Unseen Problem
QG
$\hat{q}^{(1)}$
$\hat{q}^{(2)}$
$\hat{q}^{(n)}$
QA
Final Answer
$\hat{a}^{(n)}$

Figure 3: Detailed illustration of our framework. First, a LLM is prompted to decompose the input problem P into a series of subquestion-solution pairs $(q_i^{(j)}, s_i^{(j)}, j \in \{1, \dots, n_i\})$ with an answer at each step $a_i^{(j)}$. The generated subquestions-solutions are used to train two student models: a) the QG model which learns to mimic the LLM's sub questioning capability and b) the QA model, which learns to solve each subquestion. At the bottom, the inference process is depicted for an unseen problem and no LLM is involved. The QG model breaks the unseen problem into simpler subquestions and the QA model solves each one of them eventually leading to the final answer $a_i^{(n_i)}$.

of subquestions and answers. In the iterative approach, we first generate the subquestions conditioning the generation of the QG model on P. After these questions are generated, they are provided to the QA model one by one, decoding the intermediate solution $\hat{s}^{(j)}$ at step j token by token according to the model's probability distribution over its vocabulary:

$$\mathbb{P}_{\mathcal{QA}}\left(y_j^{(k)} | y_j^{:(k-1)}, \hat{q}^{:(j)}, \hat{s}^{:(j-1)}, P\right), \qquad (2)$$

where $y_j^{(k)}$ is the k-th token being decoded in greedy fashion.

After the last solution $\hat{s}^{(n)}$ has been generated, the numerical prediction $\hat{a}^{(n)}$ is parsed from the text using simple heuristics.

4 Empirical Analysis

4.1 Datasets

We study how smaller models can learn to reason better on three multi-step reasoning datasets: GSM8K (Cobbe et al., 2021), StrategyQA (Geva et al., 2021), and SVAMP (Patel et al., 2021). GSM8K consists of 8.5K grade school math word problems, each requiring 2 to 8 steps of reasoning to solve. The solutions primarily involve a sequence of elementary calculations using basic arithmetic operations ($+$, $-$, $\times$, $\div$). The dataset is divided into 7.5K training problems and 1K test problems. To evaluate the model on SVAMP, we train the model on 761 multi-step math word problems taken from the ASDiv (Miao et al., 2020) training set and evaluate it on 237 multi-step SVAMP problems. For StrategyQA, the test set with facts is not available, so we split the data into 80% training, 10% as validation data, and the last 10% as test data. We do not shuffle the data to maintain reproducibility.

4.2 Experimental Setup

We use three kinds of annotation, corresponding to the three datasets that we consider.

Step-by-step solution. The GSM8K dataset falls into this category and includes a Socratic version where intermediate subquestion-solution pairs are provided for each MWP. While the intermediate step-by-step solutions were manually annotated, the authors report that the subquestions were generated by prompting GPT-3. We reproduced a subset of these subquestions using a GPT-3 model with prompts, and we observed a high similarity between the questions provided and the ones gen-

Unified	
Input: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?	Output: How many bolts of white fiber does it take? It takes 2/2 = <<2/2=1>> 1 bolt of white fiber. How many bolts in total does it take? So the total amount of fabric is 2+1 = <<2+1=3>> 3 bolts of fabric. The answer is 3.
Iterative	
Iteration 1	
Input: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?	Output: **QG**: How many bolts of white fiber does it take? **QA**: It takes 2/2 = <<2/2=1>> 1 bolt of white fiber.
Iteration 2	
Input: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? How many bolts of white fiber does it take? It takes 2/2 = <<2/2=1>> 1 bolt of white fiber.	Output: **QG**: How many bolts in total does it take? **QA**: So the total amount of fabric is 2+1 = <<2+1=3>> 3 bolts of fabric. The answer is 3.

Table 1: Example demonstraing the input-output format for unified vs iterative setup. QG represents the question generation model and QA is the question answerer model. Note that the QA model uses the QG output to answer it as shown in Figure 3.

erated by us (BERT F_1 score of 95%). For Socratic CoT, we thus use the subquestioning annotation already provided.

Supporting facts. We study the StrategyQA dataset, which falls in this category. Strategy QA consists of a factual question with binary True/False as the final answer. Additional supporting facts and decomposed questions are provided. However, the set of facts and the decomposed questions provided with a given question are not always aligned (i.e., a fact is not necessarily the answer to one subquestion). Therefore, having a setup similar to the one for GSM8K is not possible. We thus consider two versions of the data. One in which the supporting facts are used as CoT and the corresponding questions are generated by prompting a GPT-3 model, and a second in which we take the provided questions and generate the facts (this time aligned with the questions) using GPT-3.

Final answers only. AsDiv/SVAMP falls in this category and for training, we use GPT-3 to generate both intermediate subquestions and solutions. Intermediate solutions are used as CoT and the generated subquestion-solution pairs for Socratic CoT.

4.3 Implementation Details

We use GPT-2 variants (Radford et al., 2019) as student models. GPT-3 175B (Brown et al., 2020) served as the teacher model for decomposing complex problems into a series of simpler substeps (we report the prompts used in Appendix Table 6).

All models were trained using the Huggingface library (Wolf et al., 2020) on an NVIDIA Tesla A100 GPU with 40 GB of memory. Each experiment was run for the same number of iterations to ensure fairness with periodic evaluation over the validation set. Teacher forcing was used during training to replace the generated responses with ground truth answers from the training dataset.

Evaluation Metric. To evaluate the question-answering performance on the GSM8K, SVAMP, and StrategyQA datasets, we compute the accuracy based on the final answer provided by the student model.

5 Results and Discussion

Can our framework improve the reasoning capabilities of smaller models? Table 2 demonstrates that leveraging LLMs reasoning capabilities using our framework can improve the reasoning results for all dataset types.

Step-by-Step Solution. When human-annotated step-by-step solutions are available, training smaller models with LLM-generated CoT is not advantageous, as shown on GSM8K. This is to be expected since the annotation generated by an LLM is likely to be noisier and of lower quality than human-annotated data. However, the ground-truth step-by-step annotation can be leveraged to prompt an LLM to generate subquestions for the Socratic CoT approach, giving a performance

						Iterative		Unified
Dataset	**Model**	**Answer Only**	**GT Steps**	**GT Facts**	**CoT**	**Soc**$_{CoT}$	**Soc**$_{GT}$	**Soc**$_{CoT}$
GSM8K	Small (124M)	1.45	5.05	-	4.70	5.98	**6.44** (↑ 20%)	5.10
	Medium (355M)	2.90	7.88	-	7.10	11.57	**12.74** (↑ 38%)	7.90
	Large (774M)	4.62	14.10	-	12.85	17.89	**21.08** (↑ 33%)	13.25
	GPT-3 (6B)	-	21.00	-	-	-	-	-
StrategyQA	Medium (355M)	54.10	-	52.02	55.01	52.05	**60.31** (↑ 13%)	52.05
	Large (774M)	61.10	-	62.80	55.90	61.32	**66.40** (↑ 5%)	59. 45
	XL (1.5B)	60.51	-	**66.30**	58.07	62.30	63.56 (↓ 4%)	62.05
SVAMP	Small (124M)	2.15	-	-	5.35	**6.79**	-	5.82
	Medium (355M)	4.80	-	-	17.30	**18.99**	-	17.62
	Large (774M)	7.40	-	-	**23.60**	18.14	-	17.45

Table 2: Accuracy comparison (in %) on the three considered datasets. We consider three human-annotated baselines: final answers only (Answer Only), ground-truth step-by-step solution (GT Steps), and supporting facts (GT Facts). We compare the different supervision strategies for fine-tuning the small models: **CoT** represents the case where the chain of intermediate reasoning steps is generated by GPT-3, **Soc**$_{CoT}$ represents the case where both the chain of intermediate solutions and the subquestions are generated by LLM and used to fine-tune small models. **Soc**$_{GT}$ represents the case where GT solutions/facts are used when prompting GPT-3 to generate the subquestions. Iterative and Unified represent the two **Soc**$_{CoT}$ strategies described above. All models are GPT-2 versions and their size is reported within parentheses. All experiments were run at least 3 times and the average is reported. GPT-3 6B results are taken from Cobbe et al. (2021).

boost of up to 38% when the LLM-generated subquestions are used at inference time. When the subquestions are learned by the QG model (Iterative **Soc**$_{CoT}$), the accuracy of the student model decreases slightly but still improves over the step-by-step annotation without subquestions (17.89 vs. 14.10). Figure 5 shows a comparison of predictions generated by **Soc**$_{CoT}$ models and a model trained on the GT step-by-step annotation. Unified SOCRATIC COT performs similarly to training with the step-wise ground-truth annotation. We additionally include the score produced by GTP-3 6B to show that training with SOCRATIC COT can help a small model (GPT-2 large with 774M parameters) perform as well as a nearly 10x larger model fine-tuned with human annotated data.

Supporting facts. On StrategyQA, we observe that the inclusion of ground-truth supporting facts in the fine-tuning procedure improves the performance of the small models. However, surprisingly, when the supporting facts are generated by GPT-3, their inclusion actually hurts performance (58.07 vs 60.51 for GPT-2 Large). We hypothesize that this is likely due to the imperfect factual knowledge provided by the LLM, which mars the quality of the supervision. We have observed that the GT supporting facts provided often do not represent a logical sequence of propositions leading to the final answer. This is likely the reason why decomposing the problem through subquestions based on such facts actually harms accuracy (see **Soc**$_{CoT}$ column in Table 2). Instead, using the provided subquestions and using an LLM to generate the answers (representing coherent facts leading to the final answer) proves to be an effective strategy (60.31 vs. 52.02 for GPT-2 Medium). A more detailed comparison between our proposed approaches is presented in Figure 4. However, GPT-2 XL mod-

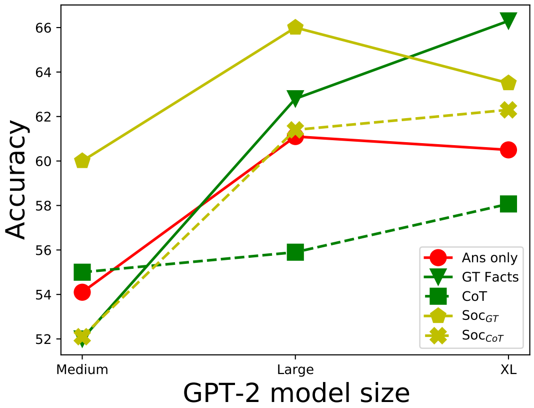
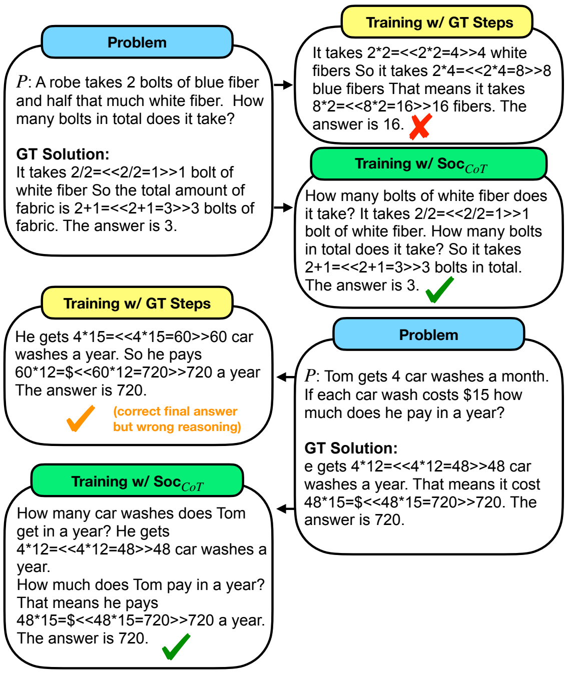


Figure 4: Accuracy comparison for different supervision strategies on StrategyQA. The baseline method consists of fine-tuning on final answers only (Ans only), and it is compared to fine-tuning with: ground-truth supporting facts (GT Facts), GPT-3-generated supporting facts (CoT), ground-truth supporting facts with GPT-3-generated subquestions (**Soc**$_{CoT}$), and LLM-generated facts with human-annotated subquestions (**Soc**$_{GT}$).



P: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?

GT Solution:
It takes 2/2=<<2/2=1>>1 bolt of white fiber So the total amount of fabric is 2+1=<<2+1=3>>3 bolts of fabric. The answer is 3.

It takes 2*2=<<2*2=4>>4 white fibers So it takes 2*4=<<2*4=8>>8 blue fibers That means it takes 8*2=<<8*2=16>>16 fibers. The answer is 16. ✗

How many bolts of white fiber does it take? It takes 2/2=<<2/2=1>>1 bolt of white fiber. How many bolts in total does it take? So it takes 2+1=<<2+1=3>>3 bolts in total. The answer is 3. ✓

P: Tom gets 4 car washes a month. If each car wash costs $15 how much does he pay in a year?

GT Solution:
e gets 4*12=<<4*12=48>>48 car washes a year. That means it cost 48*15=$<<48*15=720>>720. The answer is 720.

He gets 4*15=<<4*15=60>>60 car washes a year. So he pays 60*12=$<<60*12=720>>720 a year The answer is 720. ✓

How many car washes does Tom get in a year? He gets 4*12=<<4*12=48>>48 car washes a year.
How much does Tom pay in a year? That means he pays 48*15=$<<48*15=720>>720 a year.
The answer is 720. ✓

Figure 5: Example of predictions generated by a GPT-2 Large model fine-tuned with GT steps and SOCRATIC COT on GSM8K dataset.

Models	Methodology	Accuracy
GPT-3 (1-shot)	CoT	27.5
(175B)	Sub-ques	**47.1** (↑ 41%)

Table 3: Accuracy comparison (in %) of using CoT vs SOCRATIC COT (Sub-ques) on the GSM8K dataset for GPT-3 model with prompting.

els perform well when trained on facts as unlike smaller models, larger models can encode more facts at once in their parameters, which assists in answering a factual question.

Answers only. On the SVAMP dataset, which includes only final answers and no intermediate annotation, LLMs can be used to generate both the intermediate steps and the subquestions. Both the consideration of intermediate solutions without subquestions (**CoT**) and the consideration of intermediate solutions with subquestions (**Soc**$_{CoT}$) lead to an improvement in performance. The trend here is similar to what was observed for StrategyQA, with SOCRATIC COT being more effective for the two smaller models but falling back to **CoT** for the larger model.

Can SOCRATIC COT be used as a prompting strategy? We experimented with SOCRATIC COT as a prompting strategy. First, we prompted GPT-3 (175B) to decompose the main problem into simpler steps by formulating subquestions. Then, GPT-3 is used again to solve the sequence of subproblems in a single-shot setting with a problem decomposed into intermediate subquestions and solutions included in the prompt. The introduction of subquestioning boosts accuracy by over 40% compared to standard CoT prompting (Table 3). Other work (e.g., Wei et al. 2022b) has used a larger number of exemplars in the few-shot prompt, achieving higher overall accuracy. We limited our experiments to single-shot prompts due to budget constraints.

6 Ablation Studies

In this Section, we describe additional analyses regarding specific components of the framework we propose, as well as negative results that we obtained with alternative strategies.

How good are the sub-questioning capabilities of a smaller model? We investigate in more detail the ability of a small model to decompose a problem by generating meaningful subquestions. We fine-tuned GPT-2 Large on the GPT-3 generated subquestions provided in the GSM8K dataset. We then evaluated the quality of the generated questions in terms of BLEU score (Post, 2018), BERT F_1 score (Zhang et al., 2019), and by measuring for how many problems the number of questions generated by GPT-2 (#Q) matches the number of GPT-3 annotated questions for a given problem.

We found that the fine-tuned GPT-2 predicted an incorrect number of subquestions for the majority of problems (see Table 4, first row). Thus, following previous work on subquestion generation (Shridhar et al., 2022), we introduced a *guidance mechanism* that conditions the generation of subquestions for a problem P on the equations describing the intermediate solutions of P. This strategy improved the quality of the generated questions for all three metrics considered (Table 4, second row). To avoid the dependence on the step-by-step annotation of the equations for each problem P at inference time, we train an additional sequence-to-sequence model to predict, given P, the set of equations that lead to the solution of the problem. At inference time, the predictions for the guidance model are used to condition the generation by the QG model. Although the predicted equations often do not lead to the correct solution of the problem, they help the QG model to generate more meaning-

Methodology	BLEU	BERT F_1	# Q
No-guidance	51.5	0.78	0.42
Guidance	**58.8**	**0.81**	**0.80**

Table 4: BLEU, BERT F_1 and the number of questions (# Q) comparison between the question generator model and the Socratic subquestions present in the GSM8K dataset using GPT2-large model.

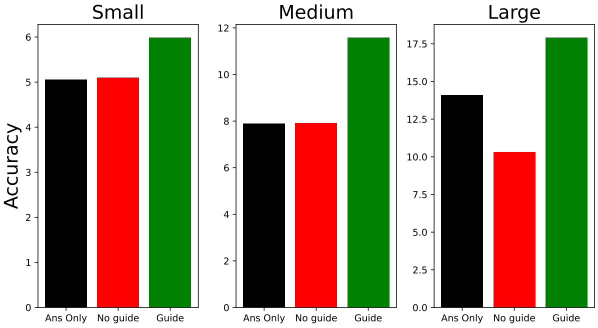


Figure 6: Accuracy of student models (QA + QG) when the question generation is conditioned using the guidance model (Guide) and with non-guided question generation (No guide). Ans only represents the baseline. All models are GPT-2 versions.

ful sub-questions. Figure 6 shows the overall accuracy of the GPT-2 student models (QA + QG) fine-tuned with SOCRATIC COT on the GSM8K data with and without equation conditioning provided by the guide model. We have extended this guidance mechanism to StrategyQA and SVAMP, where the generation of subquestions is conditioned on the number of facts (StrategyQA) or steps (SVAMP) needed to answer the problem.

Eliminating the need for a subquestion module. We have experimented with an alternative training solution that does not involve a question-generation model. This strategy aims to improve the supervision for fine-tuning a small model through subquestioning, but without relying on the presence of subquestions at test time. The procedure consists of training the student model to generate the entire chain of steps leading to an intermediate answer. That is, when the sub-question $q^{(1)}$ is asked, the model is trained to generate the answer $s^{(1)}$, but when $q^{(j)}$ is asked, the model is trained to generate the chain of thought reasoning $\{s^{(1)}, s^{(2)}, \ldots, s^{(j)}\}$ (instead of just $s^{(j)}$). This eliminates the need for the intermediate subquestions at inference time, as the model is trained to *implicitly* decompose the main problem into smaller reasoning steps. However, this method leads to significant performance degradation (results are reported in Table 5), highlighting the need for subquestions at inference time.

GPT-2	No SubQ	SubQ with QG
Small	2.70	**5.98**
Medium	7.20	**11.57**
Large	8.18	**17.89**

Table 5: Accuracy comparison (in %) of student models trained with (SubQ with QG) and without (No SubQ) question generation model on GSM8K.

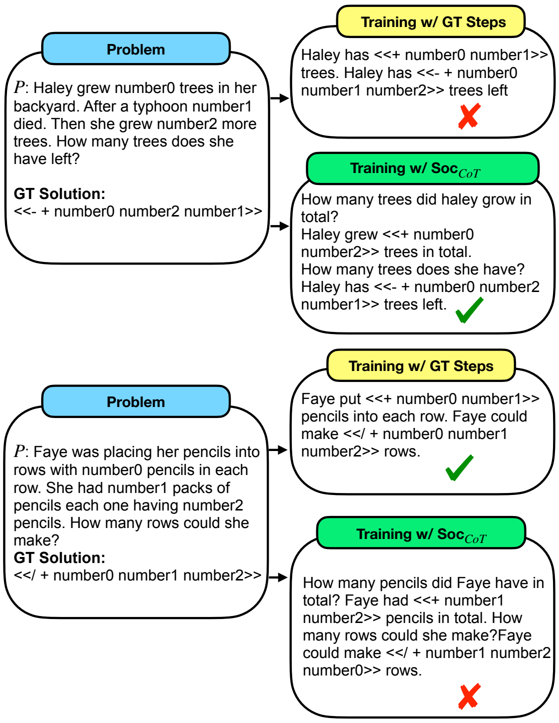

Problem
P: Haley grew number0 trees in her backyard. After a typhoon number1 died. Then she grew number2 more trees. How many trees does she have left?
GT Solution:
<<- + number0 number2 number1>>
Training w/ GT Steps
Haley has <<+ number0 number1>> trees. Haley has <<- + number0 number1 number2>> trees left
✗
Training w/ Soc$_{CoT}$
How many trees did haley grow in total?
Haley grew <<+ number0 number2>> trees in total.
How many trees does she have?
Haley has <<- + number0 number2 number1>> trees left.
✓
Problem
P: Faye was placing her pencils into rows with number0 pencils in each row. She had number1 packs of pencils each one having number2 pencils. How many rows could she make?
GT Solution:
<</ + number0 number1 number2>>
Training w/ GT Steps
Faye put <<+ number0 number1>> pencils into each row. Faye could make <</ + number0 number1 number2>> rows.
✓
Training w/ Soc$_{CoT}$
How many pencils did Faye have in total? Faye had <<+ number1 number2>> pencils in total. How many rows could she make?Faye could make <</ + number1 number2 number0>> rows.
✗

Figure 7: Example of predictions generated by a GPT-2 Medium model fine-tuned with GT steps and SOCRATIC COT on the SVAMP dataset.

Example outputs In Figures 5 and 7, we report example outputs predicted by GPT-2 models for a set of GSM8K and SVAMP problems.

7 Conclusion

The chain-of-thought style of step-by-step reasoning has proven to be very effective for reasoning in LLMs. In this work, we propose ways to distill these reasoning capabilities into smaller models and suggest ways to further improve them by explicitly asking stepwise questions. We demonstrate the effectiveness of our proposed methodology on three popular multi-step reasoning datasets, and discuss cases where one method should be preferred over the other for different datasets.

Limitations

In our work, we use only one solution from the LLM to distill information into the student model, and according to Wang et al. (2022), multiple subquestion-solution pairs can be sampled, and using majority voting, all pairs leading to the most frequent answer can be used to distill knowledge into the student models. Also, due to computational budget, we used a single prompt to compare the CoT and SOCRATIC COT and using more prompts (up to 8) might lead to a fairer comparison and better results (Wei et al., 2022b). We leave these experiments for the future.

Ethical Considerations

Although this work improves the reasoning capabilities of smaller models, the models are still not powerful enough to be used in sensitive settings such as education. We plan to release our code and model checkpoints, but the models must be used carefully by users, as many generative models, including ours, are prone to hallucination.

Acknowledgements

Alessandro Stolfo is supported by Armasuisse Science and Technology through a CYD Doctoral Fellowship.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27.

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

Jerome S Bruner. 1961. The act of discovery. *Harvard educational review*, 31:21–32.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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

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

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

Jacob Eisenstein, Daniel Andor, Bernd Bohnet, Michael Collins, and David Mimno. 2022. Honest students from untrusted teachers: Learning an interpretable question-answering pipeline from a pretrained language model. *arXiv preprint arXiv:2210.02498*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Dániel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*.

Tassilo Klein and Moin Nabi. 2019. Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv preprint arXiv:1911.02365*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. 2022. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*.

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

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

Andreas Opedal, Niklas Stoehr, Abulhair Saparov, and Mrinmaya Sachan. 2023. World models for math story problems. In *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

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

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain Yourself! Leveraging Language Models for Commonsense Reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Sudha Rao and Hal Daumé III. Answer-based Adversarial Training for Generating Clarification Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Kumar Shridhar, Jakub Macina, Mennatallah El-Assady, Tanmay Sinha, Manu Kapur, and Mrinmaya Sachan. 2022. Automatic generation of socratic subquestions for teaching math word problems. *arXiv preprint arXiv:2211.12835*.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Charlie Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *arXiv preprint arXiv:2209.15189*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schölkopf, and Mrinmaya Sachan. 2023. A causal framework to quantify the robustness of mathematical reasoning with language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

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. *arXiv preprint arXiv:2206.07682*.

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. *arXiv preprint arXiv:2201.11903*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625.

Let’s generate sub-questions for these problems. Use exactly one operation per step.

—
Q: Zoe was unboxing some of her old winter clothes . She found number0 boxes of clothing and inside each box there were number1 scarves and number2 mittens . How many pieces of winter clothing did Zoe have total ?

SQ1: How many pieces of winter clothing did Zoe have in each box?
A1: Zoe had <<+ number1 number2>> pieces of winter clothing in each box.

SQ2: How many pieces of winter clothing did Zoe have total ?
A2: Zoe had <<* number0 + number1 number2>> pieces of winter clothing in total.

—
Q: Katie picked number0 tulips and number1 roses to make flower bouquets . If she only used number2 of the flowers though , how many extra flowers did Katie pick ?

SQ1: How many flowers did Katie pick in total?
A1: Katie picked <<+ number0 number1>> flowers in total.

SQ2: How many extra flowers did Katie pick ?
A2: Katie picked <<- + number0 number1 number2>> extra flowers.

—
Q: Conner has number0 dollars in his bank account . Every month he spends number1 dollars . He does not add money to the account . How much money will Conner have in his account after number2 months ?,

SQ1: How much money does Conner spend in total? A1: Conner spends <<* number1 number2>> dollars.
SQ2: How much money will Conner have in his account after 8.0 months ? A2: After 8.0 months, Conner will have ¡¡- number0 * number1 number2>> dollars.

For each of the following topics, generate intermediate answers to the subquestions leading to the final answer.

—
Topic: Albany, Georgia (City in Georgia, United States)
Will the Albany in Georgia reach a hundred thousand occupants before the one in New York?

Albany, GA has around 75,000 people.
Albany, NY has almost 100,000 people.
The difference is 100,000-75,000=25,000
The difference is 100,000-100,000=0
No, 25,000 is not smaller than 0.
The final answer is NO.

—
Topic: The Police (English rock band)
Could the members of The Police perform lawful arrests?

Only law enforcement officers can perform lawful arrests.
No, the members of The Police (rock band) are not law enforcement officers.
The final answer is NO.

—
Topic: Wonder Woman (2017 film) (American superhero film directed by Patty Jenkins) Is a Boeing 737 cost covered by Wonder Woman (2017 film) box office receipts?

The average cost of a US Boeing 737 plane is 1.6 million dollars.
Wonder Woman (2017 film) grossed over 800 million dollars at the box office.
Yes, 800 is larger than 1.6.
The final answer is YES.

Table 6: Exemplars included in the few-shot prompt for the decomposition of the problems from the ASDiv (upper row) and StrategyQA (lower row) datasets.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitation

☑ A2. Did you discuss any potential risks of your work?
Ethical considerations

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3, Methodology

☑ B1. Did you cite the creators of artifacts you used?
Section 4.2

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Conclusion

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Our models are free to be used by anyone. We mention the limitations of our approach

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We used standard open source datasets

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4.2

C ☑ Did you run computational experiments?

Section 4.3

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4.3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4.3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4.3, Table 1

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.3

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.