THE ART OF LLM REFINEMENT: Ask, Refine, and Trust

Kumar Shridhar [◇ *] **Koustuv Sinha** [♠] **Andrew Cohen** [♠] **Tianlu Wang** [♠] **Ping Yu** [♠]
Ram Pasunuru [♠] **Mrinmaya Sachan** [◇] **Jason Weston** [♠] **Asli Celikyilmaz** [♠]

[◇] ETH Zurich [♠] Meta AI

ABSTRACT

Large Language Models (LLMs) have demonstrated remarkable generative abilities, but can they judge the quality of their own generations and self-improve? A popular concept, referred to as *self-refinement*, postulates that LLMs can detect and correct the errors in their generations when asked to do so. However, recent empirical evidence points in the opposite direction, suggesting that LLMs often struggle to accurately identify errors when reasoning is involved. To address this, we propose a reasoning with a refinement strategy called `ART: Ask, Refine, and Trust`, which *asks* necessary questions to decide when an LLM should *refine* its output, and uses it to affirm or deny *trust* in its refinement by ranking the refinement and the initial prediction. On two multistep reasoning tasks of mathematical word problems (GSM8K) and question answering (StrategyQA), `ART` achieves a performance gain of +5 points over self-refinement baselines, while using a much smaller model as the decision maker. We believe that `ART` with smaller models, making refinement decisions can be a cost-effective alternative to fine-tuning LLMs.

1 INTRODUCTION

The ability of Large Language Models (LLMs) to generate coherent and meaningful text has significantly improved over the years OpenAI (2023). However, LLMs still exhibit inaccuracies in their generations, and it has been posited that iterative refinement of generations, also using the same LLMs, can help rectify these errors Madaan et al. (2023); Shridhar et al. (2023a); Welleck et al.; Zheng et al. (2023). Madaan et al. (2023) demonstrated the potential of *self-refinement* for tasks such as dialogue response generation and sentiment reversal; however, this approach proves less effective on other tasks such as mathematical reasoning. Shridhar et al. (2023a) and Huang et al. (2023) have also demonstrated the challenges LLMs face in identifying errors in reasoning tasks. While developing models that can self-evaluate and self-correct their errors is certainly a vital step towards reliable LLMs, building such models is quite challenging.

Through empirical observation on two multistep reasoning datasets, we find that *self-refinement* does not reliably improve initial generations, validating the previous findings of Huang et al. (2023). In fact, in the majority of cases, *self-refinement* has a detrimental effect on performance. On the other hand, fine-tuning language models usually improves their performance on a given task by facilitating better adaptation to the task objectives Yuan et al. (2023). Smaller models can be trained on LLMs' data to improve their performance, which can further serve as cost-effective alternatives to LLMs for the given task Magister et al. (2023); Shridhar et al. (2023b); Hsieh et al. (2023). Thus, in this paper, we explore the possibility of training a smaller model as the decision maker for refinement, which can be used to determine *when to refine* the output of a LLM, or use the original output.

In our work, we propose a refinement approach called `ART: Ask, Refine, and Trust`, which, given an initial LLM response, works in the following three stages: (a) evaluating whether the initial generation requires refinement by asking a series of questions (`Ask`); (b) executing the *refinement* step based on the evaluation (`Refine`); and finally (c) selecting either the refined result or the initial prediction (`Trust`). On two multistep reasoning tasks, mathematical reasoning, and question

[*] Work done during internship at Meta AI; correspondence at `shkumar@ethz.ch`

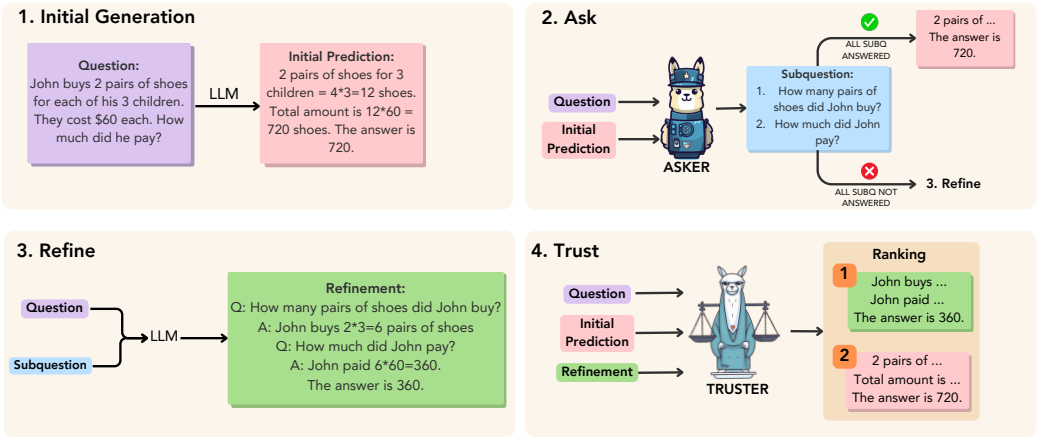

Figure 1: Our proposed objective: ART: Ask, Refine, and Trust during inference. Given a problem, an LLM first generates an initial prediction which is sent to an Asker that asks relevant questions (sub-questions) to decide whether refinement is needed or not. If all sub-questions are answered, it returns the initial prediction and no refinement is needed. If not, the model refines the initial prediction using the subquestions. Finally, the initial prediction and the refined response are sent to the Truster, which ranks them to decide if refinement was needed or if the initial prediction was better.

answering, we illustrate the effectiveness of ART by training refiners of different sizes. We observe that a much smaller model (LLaMA 7B; Touvron et al., 2023) trained to decide *when to refine*, can outperform a 10x larger model (LLaMA 70B) in a *self-refinement* setup (by up to 5 points). We evaluate the cost and accuracy tradeoffs of training a smaller model with ART to make the refinement decision for a pretrained LLM vs fine-tuning the LLM. In many cases, we illustrate the cost-effectiveness of ART as a viable alternative to fine-tuning LLMs. We show that our trained models (Asker and Truster) can work seamlessly across a wide range of LLMs (LLaMA 70B Touvron et al. (2023), ChatGPT Brown et al. (2020) and GPT-4 OpenAI (2023)) without requiring additional modifications.

2 ART: Ask, Refine, and Trust

In this section, we describe our proposed methodology ART: Ask, Refine, and Trust in detail. Given a query and an initial prediction generated by the LLM, ART uses a trainable pipeline for refinement as follows: (a) evaluate whether the initial generation requires refinement by asking a series of questions (Ask); (b) perform the *refinement* step based on the evaluation (Refine); and finally (c) choose either the refined result or the initial prediction (Trust).

2.1 Initial Prediction

Given a task query $\mathtt{x}$, the LLM ψ generates an initial prediction $\mathtt{y} = \psi(\mathtt{x})$. For pre-trained LLMs, the query $\mathtt{x}$ is augmented with several examples of the task as few-shot prompts, while for fine-tuned models the query is provided directly without any examples. Due to the multi-step reasoning nature of the tasks where intermediate steps are beneficial for the model to arrive at the final answer, we use Chain of Thought (CoT; Wei et al., 2022) and Subquestion Decomposition (Decomp; Shridhar et al., 2022; Zhou et al., 2023) as our main methods for initial prediction.

2.2 Ask

Once the initial prediction is generated, the next step is to decide when to refine the output. Refining every sample often leads to much worse performance Huang et al. (2023). Thus, we train an Asker to determine whether a prediction is correct or not, and then refine only the samples about which

the Asker is uncertain about. However, before a smaller model can determine whether a generated answer is correct or whether refinement is needed, it is important to align the model with task-specific knowledge and the expected outcome. We fine-tune the smaller model in CoT style (intermediate steps with the final answer, as shown by the "Initial Prediction" in Figure 1) on the training data.

We first describe how we create the dataset for training the Asker model. We use the LLM ψ to generate k predictions per example on the training set, and then label them "Yes" or "No" for refinement based on whether the prediction was correct or incorrect (dataset statistics are presented in Table 1). For each prediction, we append the subquestions present in the datasets[1] prior to the "Yes" or "No" decision to train the fine-tuned model. In this way the Asker learns to first ask the relevant questions, map them to the prediction, and then decide whether all its questions are answered in the prediction or not, leading to the refinement decision. An example is presented in the appendix (Figure 5).

Dataset	Train Samples		
	Fine-tune	**Asker**	**Truster**
GSM8K	7473	35000	15000
StrategyQA	1832	9000	2300

Table 1: Comparison of different data sizes used for fine-tuning, and training the Asker and Truster.

2.3 Refine

If the Asker predicts "Yes" (refinement is needed), then the LLM ψ is used to refine the output given the input and the subquestions from the Asker model, $\mathsf{y}_{\text{ref}} = \psi(\mathsf{x}; \mathsf{subq})$. Similar to Shridhar et al. (2023a), for the StrategyQA dataset, additional facts (facts) are also provided to the model ψ during refinement ($\mathsf{y}_{\text{ref}} = \psi(\mathsf{x}; \mathsf{subq}; \mathsf{facts})$). An example is presented in the appendix (Figure 6).

2.4 Trust

Finally, to decide whether the refinement output should be preferred over the original generation, we train a Truster that takes two candidates (y, y_{ref}) for the task query x and decides which one to prefer among the two. An example is presented in the appendix (Appendix A). However, we noticed that in 80% of the cases, the final answer of the refinement y_{ref} and the initial prediction y were the same. Our goal is to make Truster learn to identify the reasoning chain with the correct final answer and not a particular styled intermediate reasoning chain. Thus, to create an appropriately-sized balance training data set, we used the same training data collected for the Asker model (Table 1) and selected the prediction samples that have both a correct and an incorrect prediction. We construct preferred (correct predictions) vs. non-preferred (incorrect predictions) pairs and train Truster with the following text classification objective:

$$\mathcal{L}_\theta = -\mathbb{E}_{\mathsf{x}, y_j, y_k \sim \mathcal{D}} \left[\log(\sigma(r_\theta(\mathsf{x}, y_j) - r_\theta(\mathsf{x}, y_k)))\right]$$

where r is the score of the Truster model, y_j is the preferred candidate (correct prediction) and y_k is the non-preferred candidate (incorrect prediction) from the dataset $\mathcal{D}$. Based on the score for each sample, we select the best-scored output.

3 Experiments

3.1 Dataset

We test the ART refinement strategy on two multi-step reasoning tasks, GSM8K (Cobbe et al., 2021) and StrategyQA Geva et al. (2021). The GSM8K dataset is a grade school math word problem

[1]Note that the subquestions are available for these datasets and we directly used them to train the Asker model. However, LLMs can also be used to generate subquestions when not available. Previous work has shown that the quality of subquestions generated by LLMs on these datasets is quite close to the ground truth Magister et al. (2023).

dataset with a training set of 7473 samples and a test set of 1319 samples, each requiring two to eight steps to solve. The dataset also consists of sub-questions that correspond to the steps in a given correct solution. StrategyQA, on the other hand, is a question-answering benchmark that focuses on open-domain questions, requiring step-wise reasoning to solve it. StrategyQA consists of 2290 training examples, of which the first 20% were used as the test set and the remaining 80% as the training set, following previous work Magister et al. (2023); Shridhar et al. (2023a). Each question is accompanied by its decomposed questions and the correct factual knowledge required to answer it. An example of each dataset is presented in the appendix (Figure 6).

3.2 Experimental Setup

We use LLaMA 70B (pre-trained and chat) Touvron et al. (2023), ChatGPT (turbo (`gpt-3.5-turbo`) and turbo-instruct (`gpt-3.5-turbo-instruct`)) Brown et al. (2020), and GPT-4 (`gpt-4`) OpenAI (2023) as base models ψ due to their popularity and state-of-the-art performance. Next, we fine-tuned variants of the LLaMA model (7B, 13B, and 70B) on the GSM8K dataset and 7B and 13B on the StrategyQA dataset using the CoT strategy. All fine-tuned variants were further trained to ask relevant questions and decide when to refine to get the `Asker` model. Finally, we fine-tuned the LLaMA 13B model to get the `Truster` that decides between the original and refined output. All pre-trained and fine-tuned LLaMA models were used with greedy decoding during testing (temperature = 0 and top p = 1). To collect data for training, different temperatures were used to collect diverse samples (temperature = {0, 0.3, 0.4, 0.7, 0.8}), and k was set to 5 to generate 5 predictions on the train set. All training was done on a cluster of 8 A100 80GB GPUs (except for the LLaMA 70B fine-tuning, which required 4 clusters of 8 A100s each).

3.3 Results

Self-Refinement is not enough Table 2 shows the results of initial prediction, refinement, and trust. In general, the performance of LLaMA 70B is much lower than the ChatGPT turbo model for the GSM8K dataset (59 compared to 77 for CoT and 55 compared to 78 for Subquestion Decomposition). The Subquestion Decomposition (Decomp) approach performs better than CoT for ChatGPT, but the opposite is true for LLaMA 70B. Since the training data and the model architecture of ChatGPT are not public, it is difficult to understand the performance gap. While, *self-refinement* improves performance in some cases, it leads to worse performance in others (Red colored boxes in Table 2 show the comparison). However, combining refinement with the trust module consistently improves performance over the initial prediction in almost all cases. This demonstrates the usefulness of the different components of our proposed `ART` methodology. Note that our baselines of the `Self` modules of refinement and trust use the same prompts as presented in Shridhar et al. (2023a) for a fair comparison.

Importance of `Asking` Table 2 also demonstrates the effectiveness of training an `Asker` that decides when to refine the outputs. Compared to the self-refinement (`Self`) strategy, a much smaller model like LLaMA 7B ($\texttt{Asker}_{\text{7B}}$) outperforms much larger LLMs like ChatGPT self-refinement (`Self`) by over 2 points (80.89 vs. 78.62). LLaMA 13B ($\texttt{Asker}_{\text{13B}}$) improves it by over 4 points (78.62 $\rightarrow$ 82.18). The trend is similar when refinements are compared with the self-refinement capabilities (`Self`) of LLaMA 70B, where a 7B model ($\texttt{Asker}_{\text{7B}}$) outperforms the pre-trained self-refinement capabilities of LLaMA 70B by about 2 points (61.33 vs. 59.83) and over 1 point for the chat model (58.83 vs. 60.12). The 13B model ($\texttt{Asker}_{\text{13B}}$), on the other hand, improves it by over 3 points for the pre-trained LLaMA 70B model (59.83 $\rightarrow$ 62.74) and the chat version by more than 4 points (58.83 $\rightarrow$ 63.00). Finally, using the 70B model as `Asker` ($\texttt{Asker}_{\text{70B}}$) further improves the results by 4 points for the pre-trained version (59.83 $\rightarrow$ 63.60) and over 5 points for the chat version (58.83 $\rightarrow$ 63.80). The results follow a similar trend for the GPT-4 models, where both the 7B ($\texttt{Asker}_{\text{7B}}$) and 13B ($\texttt{Asker}_{\text{13B}}$) models improve the results over the initial generation by about 2 points (91.88 $\rightarrow$ 93.72), which is higher than other baselines from Madaan et al. (2023) and Shridhar et al. (2023a). Finally, note that our proposed strategy `ART` improves the overall performance of ChatGPT to 82.18 after refining with a single pass (`maj1@1`), which is similar to the self-consistency score of 3 samples (`maj1@3`) Huang et al. (2023).

The results on StrategyQA follow a similar trend, where a 7B model $\texttt{Asker}_{\text{7B}}$ improves the refinement score by 1 point for LLaMA 70B (75.15 $\rightarrow$ 76.22) and over 3 points for ChatGPT (70.52 $\rightarrow$ 73.84),

Model	Initial Prediction		Refinement			Trust	
Type	**Method**	**Accuracy**	**Subquestions**	**Model**	**Accuracy**	**Model**	**Accuracy**
LLaMA 70B							
Pre-trained	CoT	59.74	No	Self	59.07	Self	59.83
Pre-trained	CoT	59.74	Yes	Self	59.83	Self	60.43
Pre-trained	Decomp	54.66	No	Self	55.11	Self	55.34
Pre-trained	Decomp	54.66	Yes	Self	50.26	Self	54.51
Pre-trained	CoT	59.74	Yes	$\texttt{Asker}_{7B}$	61.33	Truster	61.94
Pre-trained	CoT	59.74	Yes	$\texttt{Asker}_{13B}$	62.74	Truster	63.85
Pre-trained	CoT	59.74	Yes	$\texttt{Asker}_{70B}$	**63.60**	Truster	**64.24**
Chat	CoT	58.90	No	Self	59.10	Self	58.79
Chat	CoT	58.90	Yes	Self	58.83	Self	59.55
Chat	CoT	58.90	Yes	$\texttt{Asker}_{7B}$	60.12	Truster	61.18
Chat	CoT	58.90	Yes	$\texttt{Asker}_{13B}$	63.00	Truster	63.30
Chat	CoT	58.90	Yes	$\texttt{Asker}_{70B}$	**63.80**	Truster	**64.40**
ChatGPT							
Turbo	CoT^{S}	71.64	No^{S}	Self^{S}	73.00	Self^{S}	72.93
Turbo	CoT^{S}	71.64	Yes^{S}	Self^{S}	73.99	Self^{S}	73.99
Turbo	CoT^{SR}	74.58	No^{SR}	Self^{SR}	75.00	Most Recent^{SR}	75.00
Turbo	$\text{CoT}^{!C}$	75.90	$\text{No}^{!C}$	$\text{Self}^{!C}$	75.10	$\text{Most Recent}^{!C}$	75.10
Turbo	CoT	77.71	No	Self	78.16	Self	78.28
Turbo	CoT	77.71	Yes	Self	78.46	Self	78.89
Turbo	Decomp	78.62	No	Self	78.99	Self	78.99
Turbo	Decomp	78.62	Yes	Self	78.24	Self	79.22
Turbo	CoT	77.71	No	$\texttt{Asker}_{7B}$	80.89	Truster	81.14
Turbo	CoT	77.71	Yes	$\texttt{Asker}_{13B}$	**82.18**	Truster	**82.64**
Instruct	CoT	71.26	No	Self	70.28	Self	71.50
Instruct	CoT	71.26	Yes	Self	72.32	Self	72.85
Instruct	CoT	71.26	Yes	$\texttt{Asker}_{7B}$	76.19	Truster	76.34
Instruct	CoT	71.26	Yes	$\texttt{Asker}_{13B}$	**78.46**	Truster	**79.86**
GPT-4							
-	CoT^{S}	91.45	Yes^{S}	Self^{S}	90.80	Self^{S}	93.10
-	CoT^{SR}	92.90	No^{SR}	Self^{SR}	93.10	Most Recent^{SR}	93.10
-	CoT	91.88	Yes	$\texttt{Asker}_{7B}$	93.25	Truster	93.45
-	CoT	91.88	Yes	$\texttt{Asker}_{13B}$	**93.72**	Truster	**94.08**

Table 2: Accuracy (maj1@1) comparison between different methods and refinement strategies on the GSM8K dataset. Initial Prediction refers to the initial generation from the LLM with its Method referring to one of the reasoning strategies (Chain of Thought (CoT) or Subquestion Decomposition (Decomp) in our case). Refinement refers to the combination of the *Ask* and the *Refine* stages in ART with or without the use of subquestions during refinement (subquestions). Finally, Trust refers to the *Trust* stage in ART, where *Self* refers to *self-refinement*, Truster is our trained model and *Most Recent* refers to choosing refinement as the final result. Yellow represents the baseline methods from previous work ($(.)^{S}$ represents results from Shridhar et al. (2023a), $(.)^{SR}$ from Madaan et al. (2023), and $(.)^{!C}$ from Huang et al. (2023)), Red represents our implementations of the baselines, and Green represents our proposed methods. Underline represents the best results from previous strategies, and **bold** represents the overall best result.

as shown in Table 3. Note that following Shridhar et al. (2023a), we also provide some factual information along with the questions during refinement so that the model can correct its factual inaccuracy. The gains are larger for the $\texttt{Asker}_{13B}$ model, where the performance improves by 3 points for LLaMA 70B (75.15 $\rightarrow$ 78.38) and 5 points for ChatGPT (70.52 $\rightarrow$ 75.76), demonstrating the clear importance of asking questions for refinement decision making.

(Don't) Always Trust Refinement Table 2 also demonstrates the usefulness of a trust module that decides whether the refinement improves or degrades the initial prediction. We train a Truster model that learns to rank the initial prediction and the refined output and decides which one to choose

Initial Pred	Refinement		Trust	
Acc	**Model**	**Acc**	**Model**	**Acc**
LLaMA 70B Pre-trained				
74.45	Self	75.15	Self	75.74
74.45	$\texttt{Asker}_{7B}$	76.22	Truster	76.12
74.45	$\texttt{Asker}_{13B}$	**78.38**	Truster	**78.44**
ChatGPT Turbo				
73.58	Self	70.52	Self	74.89
73.58	$\texttt{Asker}_{7B}$	73.84	Truster	74.04
73.58	$\texttt{Asker}_{13B}$	**75.76**	Truster	**75.86**

Table 3: Accuracy comparison on the StrategyQA dataset for refinement and trust with different models. Red represents our implementations of the baselines, and Green represents our proposed methods.

for a given input. Our trained Truster model (LLaMA 13B) achieves an accuracy of the pre-trained LLaMA 70B of as high as 64.24, which is 4 points higher than the baseline (60.43). The trend is similar for the chat version, where the improvement is almost 5 points over the baseline method of using the same LLM for decision-making (59.55 $\rightarrow$ 64.40). The results follow a similar trend for ChatGPT where the improvement over baselines (the same LLM) is about 4 points for the Turbo model over the baselines (78.89 $\rightarrow$ 82.64) and about 7 points from the best previous method of Self-Refine Madaan et al. (2023) (75.10 of Self-Refine $\rightarrow$ 82.64). The gains for GPT-4 are very small, possibly due to the high performance of the GPT-4 model, but Truster improves the performance to 94.08 from the previous best refinement score of 93.10.

For StrategyQA, the trust module does not prove to be very helpful with a performance very similar to the refinement scores. This shows that it is difficult to train a Truster on fact-based datasets, as it is hard to rank two pieces of factual information without knowing the true facts.

Cost of fine-tuning LLMs vs. ART-based refinement We fine-tune LLaMA 70B to compare it with ART based refinment approach. Fine-tuning LLaMA 70B achieves 63.2% accuracy on GSM8K Yuan et al. (2023). This is similar to what a trained 13B $\texttt{Asker}_{13B}$ and Truster can achieve with a pre-trained LLaMA 70B model, while incurring much lower training costs and computational requirements. Table 4 shows that training a 13B model as Truster is 10X cheaper than fine-tuning a 70B model, and even considering both the trained models (Asker and Truster), ART is still 5X cheaper. In addition, while fine-tuning usually makes the model narrowly specialized to the trained dataset with reduced general in-context learning performance Wang et al. (2022b), this doesn't happen with our pre-trained model as a separate model decides when to refine in our ART framework.

Objective	Model Size	Flops	GPU Hours
Asker	7B	1.5×10^{17}	1
Truster	13B	3×10^{17}	4
FineTuning	70B	1.5×10^{18}	75

Table 4: Comparison of different compute requirements for training different sized LLaMA models on GSM8K to train a decision maker (Asker and Truster) vs. finetuning a model (FineTuning).

4 Ablation Studies

Importance of Asking Questions for Refinement For this ablation, we trained Asker to make only a binary decision of "Yes" or "No" to refine, without asking the relevant questions, and found that all versions of the LLaMA models always trusted the predictions and never decided to refine them. LLMs are often bad at judging their own predictions and often prefer their own predictions Kadavath et al. (2022), and our experiments observed a similar phenomenon. However, asking questions leads to a better refinement decision. A qualitative example is presented in Figure 4.

Importance of `Truster` for selection We compared the performance of the selection module of the LLM (Self) vs. our trained `Truster` for the GSM8K dataset and observed that the trained `Truster` can better assess the errors made in the predictions and asks the model to revert to the previous generation more (about 50% more compared to self-selection); leading to superior performance.

When to refine? Assessing when to refine is an important component of the refinement pipeline, as always refining leads to worse results Huang et al. (2023). Figure 3 supports the previous findings and shows that always refining (100% refinement) can hurt overall performance and is worse than the initial prediction (0% refinement). The sweet spot is somewhere in the middle (about 30-35% refinement seems to work for both ChatGPT and LLaMA 70B models on the GSM8K dataset).

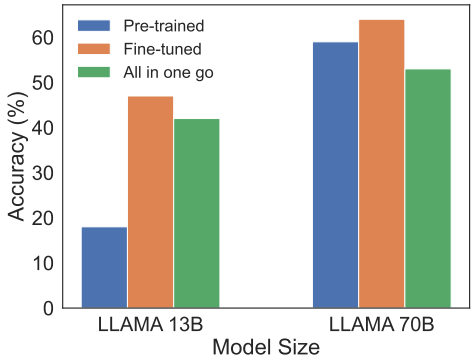

Figure 2: Comparison of the all-in-one approach to fine-tuning the LLMs on the GSM8K dataset.

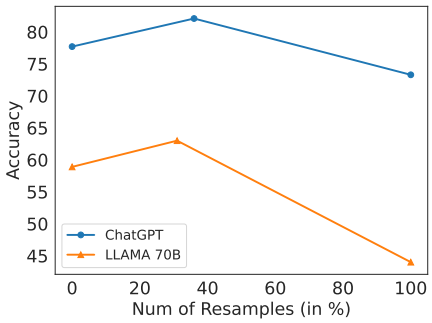

Figure 3: Number of resamples for refinement for ChatGPT and LLaMA 70B models on GSM8K. 0 means no resampling and 100 means resampling all the time.

Can `Asker` be trained on its own output? Instead of training an `Asker` to ask questions on the output of the pre-trained LLM, can we train it on its predictions? We find that `Asker` trained on its data can make a better refinement decision than LLM's self-refinement abilities. However, as expected, training on pre-trained model data proved to be more beneficial in deciding when to refine, due to a similar distribution of test and training samples, as shown in Table 5. However, for ChatGPT models, `Asker` trained on its data performs similarly to that trained on LLaMA 70B models, as both data distributions are different from the ChatGPT test distribution (82.10 vs. 82.18 for the 13B model and 80.69 vs. 80.89 for the 7B model).

Initial Prediction	Refinement		
Accuracy	**Model**	**Data**	**Accuracy**
LLaMA 70B Pre-trained			
59.74	$\texttt{Asker}_{7B}$	7B	59.21
59.74	$\texttt{Asker}_{7B}$	70B	**61.33**
59.74	$\texttt{Asker}_{13B}$	13B	62.39
59.74	$\texttt{Asker}_{13B}$	70B	**62.74**
ChatGPT Turbo			
77.71	$\texttt{Asker}_{7B}$	7B	80.69
77.71	$\texttt{Asker}_{7B}$	70B	**80.89**
77.71	$\texttt{Asker}_{13B}$	13B	82.10
77.71	$\texttt{Asker}_{13B}$	70B	**82.18**

Table 5: Comparison of refinement results when `Asker` models are trained on their own outputs vs. LLaMA 70B model outputs on the GSM8K dataset. LLaMA 70B and ChatGPT models are used as decision makers.

Can `ART` methodology be extended to SoTA models? We fine-tune MetaMath 7B and 13B Yu et al. (2023) as `Asker` using the same training strategy as `Asker` models and compare the results

Question A customer's loyalty card at a store gives them rewards of $1 off their next purchase for every $20 they spend. On their last shopping trip, they spent $80. This shopping trip, they spent $43, used their rewards, and applied a coupon that took twice the amount of rewards off the price. How many dollars did the customer pay on this shopping trip?

Initial Prediction The customer spent $80 on their last shopping trip, so they earned 80/20 = «80/20=4»4 rewards. They used 2*4 = «2*4=8»8 rewards on this shopping trip. So the price of their purchase was $43 - $8 = $«43-8=35»35. The customer paid $35 on this shopping trip. The answer is 35. ✗

Self-Refinement Yes, the reasoning is correct. The customer spent $80 on their last shopping trip, earning them 80/20 = 4 rewards. They used 8 rewards on this shopping trip. Therefore, the price of their purchase was $43 - $8 = $35. ✗

`Asker` How much did the customer get in rewards? What was the coupons cost? How much did the customer pay on this shopping trip? All questions are not answered.

Refinement How much did the customer get in rewards? The customer had 80 / 20 = $«80/20=4»4 in rewards. What was the coupons cost? Their coupon was worth 4 * 2 = $«4*2=8»8. How much did the customer pay on this shopping trip? Thus they paid 43 - 4 - 8 = $«43-4-8=31»31 on this shopping trip. ✓

Figure 4: Qualitative example demonstrating the usefulness of asking questions to decide *when to refine* and then using those questions to refine the initial prediction by fixing the errors.

in Table 6. MetaMath, due to its training on additional data and superior mathematical reasoning abilities, can evaluate the prediction better and leads to a better overall performance after refinement (62.31 vs. 61.33 for 7B and 64.06 vs. 62.74 for 13B variant). MetaMath models also perform better with fewer samples (about 27% fewer samples for 7B and 25% fewer for 13B), suggesting that superior mathematical reasoning can help to better evaluate predictions, leading to fewer uncertain samples for refinement. Since MetaMath was trained on over 250K samples with rejection sampling, we couldn't run all experiments on this large dataset, and we stick to LLaMA models for all of our experiments.

Initial pred		Refinement	
Accuracy	`Asker`	**Acc (↑)**	**% samp (↓)**
59.74	LLaMA 7B	61.33	48
59.74	MetaMath 7B	**62.31**	35
59.74	LLaMA 13B	62.74	36
59.74	MetaMath 13B	**64.06**	27

Table 6: Comparison of LLaMA 7B and 13B refinement accuracy (Acc) with MetaMath 7B and 13B models Yuan et al. (2023) and their sampling percentage (% samp) for refinement.

Entire `ART` pipeline in one go To test whether the entire `ART` pipeline of asking relevant questions, then deciding whether the questions are answered or not, and then refining can be learned in one go instead of individual models for each step, we train a LLaMA 13B and 70B model over the entire sequence (all-in-one-go). Figure 2 shows that all-in-one-go (green) performs worse than fine-tuning (orange) for the LLM, demonstrating that learning the sequence of modules together is more challenging for LLMs than modularly learning the various components in the pipeline.

5 Key Findings

From the experiments, we observe the following:

- **`ART` allows smaller models to make refinement decisions superior to LLM self-refinement**: Smaller models trained to make a refinement decision can outperform a much larger model in *self-refinement* style (Table 2).
- **Ask questions before refinement** Asking questions is an effective way to verify the quality of the generations and allows the models to make better refinement decisions.

- **Smaller models' refinement decisions are a cost-effective alternative to fine-tuning LLMs** The refinement decision of smaller models combined with a pre-trained LLM performs similarly to a larger model when fine-tuned. This saves a lot of computation required to fine-tune a larger model (Table 4) and preserves downstream performance on other tasks.
- **Expert models can make better judgments about refinement** Larger models ($\texttt{Asker}_{13B}$ performance is better than $\texttt{Asker}_{7B}$ in all cases) show that better models can make more informed decisions about when to refine. Table 6 shows that MetaMath trained models outperform LLaMA models of similar size.
- **Trained `Truster` can rank decisions better** A trained smaller `Truster` model can rank the results better than the self-selection version of LLMs.

6 Related Work

Strategies that use intermediate computation to solve reasoning tasks such as chain of thought Wei et al. (2022); Lewkowycz et al. (2022); Zhang et al. (2022); Kojima et al. (2022); Wang et al. (2022a); LYU et al. (2023) and subquestion decomposition Min et al. (2019); Shridhar et al. (2022); Zhou et al. (2023); Radhakrishnan et al. (2023) have proven to be very effective. However, a lot of times, these LLMs don't get the correct outputs in the first place, often requiring a *refinement*. Most LLM refinement techniques use one of these two strategies Madaan et al. (2023); Welleck et al.; Huang et al. (2023); Paul et al. (2023); Yoran et al. (2023) or occasionally a combination of the two Shridhar et al. (2023a). Shridhar et al. (2023a) unified past *reasoning with refinement* methods under a common umbrella of *sampling* (given a query, LLM generates the initial response), *re-sampling* (LLM refines the initial response), and *selection* (choose either the refinement or rollback to initial response). However, a single LLM was used to perform the initial generation, refinement, and later selection by using different prompts. This makes the entire process dependent on the capability of a single model. We, on the other hand, propose to train a separate, much smaller capacity model to make refinement decisions and later decide whether or not to trust the refinement over the initial generation and are not limited to prompting-based solutions.

Asking questions to verify the factuality of the LLM output has been studied by Dhuliawala et al. (2023). However, this work only deals with hallucinations in the form of directly stated factual inaccuracies. It is important to note that hallucinations can take other forms, including incorrect reasoning steps for reasoning problems considered in our paper. To address this, we train an expert model to verify each reasoning step by asking relevant questions.

Training a model to rank LLM outputs has been studied in the past in various contexts Burges et al. (2005), including but not limited to open-ended text generation Krishna et al. (2022), mathematical reasoning Cobbe et al. (2021), machine translation Tunstall et al. (2023), and so on. In our work, we do not study the standard setting of training to rank the quality of generations, but rather to decide if the refinement is inappropriate and needs to be rolled back. Choosing one decision over the other has some additional similarities to rejection sampling fine-tuning Yuan et al. (2023), where a model is trained to generate and collect the correct reasoning chains as augmented fine-tuning datasets. On the other hand, we collect both correct and incorrect reasoning chains for ranking the outputs.

Finally, our work is similar to distilling reasoning skills into smaller models Shridhar et al. (2023b); Magister et al. (2023); Hsieh et al. (2023). However, instead of teaching smaller models to reason, we train smaller models to ask questions to verify the reasoning and decide whether the reasoning is correct, which differs from asking questions as planning to reason Shridhar et al. (2022).

7 Conclusion

In this work, we propose a refinement strategy called `ART: Ask, Refine, and Trust`, which allows smaller models to make refinement decisions for LLMs and determine whether these refinements are reliable. We empirically demonstrate the effectiveness of our approach on two reasoning tasks, mathematical word problems and question answering. Our results show that smaller models, even up to 10X smaller, can outperform larger models in making refinement decisions.

REFERENCES

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 89–96, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *ArXiv*, abs/2309.11495, 2023. URL https://arxiv.org/abs/2309.11495.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*, 2021. URL https://aclanthology.org/2021.tacl-1.21/.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander J. Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *ArXiv*, abs/2305.02301, 2023. URL https://arxiv.org/abs/2305.02301.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *ArXiv*, abs/2310.01798, 2023. URL https://arxiv.org/abs/2310.01798.

Saurav Kadavath, Tom Conerly, Amanda Askell, T. J. Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zachary Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, John Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom B. Brown, Jack Clark, Nicholas Joseph, Benjamin Mann, Sam McCandlish, Christopher Olah, and Jared Kaplan. Language models (mostly) know what they know. *ArXiv*, abs/2207.05221, 2022. URL https://arxiv.org/abs/2207.05221.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. Rankgen: Improving text generation with large ranking models. *ArXiv*, abs/2205.09726, 2022. URL https://arxiv.org/abs/2205.09726.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning

problems with language models. *ArXiv*, abs/2206.14858, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

QING LYU, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. *ArXiv*, abs/2301.13379, 2023. URL https://api.semanticscholar.org/CorpusID:256416127.

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

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1773–1781, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.151. URL https://aclanthology.org/2023.acl-short.151.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6097–6109, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1613. URL https://aclanthology.org/P19-1613.

OpenAI. Gpt-4 technical report, 2023.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations. *ArXiv*, abs/2304.01904, 2023. URL https://arxiv.org/abs/2304.01904.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson E. Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, John Kernion, Kamil.e Lukovsiut.e, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Sam McCandlish, Sheer El Showk, Tamera Lanham, Tim Maxwell, Venkat Chandrasekaran, Zac Hatfield-Dodds, Jared Kaplan, Janina Brauner, Sam Bowman, and Ethan Perez. Question decomposition improves the faithfulness of model-generated reasoning. *ArXiv*, abs/2307.11768, 2023. URL https://arxiv.org/abs/2307.11768.

Kumar Shridhar, Jakub Macina, Mennatallah El-Assady, Tanmay Sinha, Manu Kapur, and Mrinmaya Sachan. Automatic generation of socratic subquestions for teaching math word problems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, December 2022. URL https://aclanthology.org/2022.emnlp-main.277/.

Kumar Shridhar, Harsh Jhamtani, Hao Fang, Benjamin Van Durme, Jason Eisner, and Patrick Xia. Screws: A modular framework for reasoning with revisions. *ArXiv*, abs/2309.13075, 2023a. URL https://arxiv.org/abs/2309.13075.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 7059–7073, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.441. URL https://aclanthology.org/2023.findings-acl.441.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra,

Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023. URL https://arxiv.org/abs/2307.09288.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Alexander M. Rush, and Thomas Wolf. The alignment handbook. https://github.com/huggingface/alignment-handbook, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. Self-Consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022a. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix X. Yu, Cho-Jui Hsieh, Inderjit S. Dhillon, and Surinder Kumar. Two-stage llm fine-tuning with less specialization and more generalization. 2022b. URL https://arxiv.org/abs/2211.00635.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. *In The Eleventh International Conference on Learning Representations, 2023*. URL https://openreview.net/forum?id=hH36JeQZDaO.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. Answering questions by meta-reasoning over multiple chains of thought. *ArXiv*, abs/2304.13007, 2023. URL https://arxiv.org/abs/2304.13007.

Long Long Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zheng Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *ArXiv*, abs/2309.12284, 2023. URL https://arxiv.org/abs/2309.12284.

Zheng Yuan, Hongyi Yuan, Cheng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *ArXiv*, abs/2308.01825, 2023. URL https://arxiv.org/abs/2308.01825.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alexander J. Smola. Automatic chain of thought prompting in large language models. *ArXiv*, abs/2210.03493, 2022. URL https://openreview.net/forum?id=5NTt8GFjUHkr.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. *In The Eleventh International Conference on Learning Representations, 2023*, 2023. URL https://openreview.net/pdf?id=WZH7099tgfM.

A Appendix

Question A customer's loyalty card at a store gives them rewards of $1 off their next purchase for every $20 they spend. Their last shopping trip, they spent $80. This shopping trip, they spent $43, used their rewards, and applied a coupon that took twice the amount of rewards off the price. How many dollars did the customer pay on this shopping trip?

Initial Prediction The customer spent $80 on their last shopping trip, so they earned 80/20 = «80/20=4»4 rewards. They used 2*4 = «2*4=8»8 rewards on this shopping trip. So the price of their purchase was $43 - $8 = $«43-8=35»35. The customer paid $35 on this shopping trip. The answer is 35. ✗

Ask How much did the customer get in rewards? What was the coupons cost? How much did the customer pay on this shopping trip? All questions are not answered.

Refine *How much did the customer get in rewards?* The customer had 80 / 20 = $«80/20=4»4 in rewards. *What was the coupons cost?* Their coupon was worth 4 * 2 = $«4*2=8»8. *How much did the customer pay on this shopping trip?* Thus they paid 43 - 4 - 8 = $«43-4-8=31»31 on this shopping trip. ✓

Trust The customer had 80 / 20 = $«80/20=4»4 in rewards. Their coupon was worth 4 * 2 = $«4*2=8»8. Thus they paid 43 - 4 - 8 = $«43-4-8=31»31 on this shopping trip. ✓

Question The cell-phone recharges at the rate of 1 percentage-point of charge per 3 minutes. Now, the phone is at 60% charged. How long will it take to fully charge, in hours?

Initial Prediction The phone needs 100-60 = «100-60=40»40% more charge to be fully charged. Since the phone recharges at the rate of 1 percentage-point of charge per 3 minutes, it will take 40*3 = «40*3=120»120 minutes to fully charge. Therefore, it will take 120/60 = «120/60=2»2 hours to fully charge. The answer is 2. ✓

Ask How many percentage points is left to be charged? How many minutes will it take to fully charge? How long will it take to fully charge, in hours? All questions are answered.

Figure 5: End to end flow of our proposed objective of `ART: Ask, Refine, and Trust` if `Asker` decides to refine (top) vs no refinement needed (bottom).

Question Claire makes a 3 egg omelet every morning for breakfast. How many dozens of eggs will she eat in 4 weeks?

Subquestions How many eggs does Claire eat in a week? How many eggs will Claire eat in 4 weeks? How many dozens of eggs will Claire eat in 4 weeks?

Answer She eats 3 eggs every day and there are 7 days in a week so she eats 3*7 = «3*7=21»21 eggs a week. After 4 weeks she will have eaten 4*21 = «4*21=84»84 eggs. There are 12 eggs in 1 dozen and she'll eat 84 eggs so that's 84/12 = «84/12=7»7 dozen eggs.

Question Can a greyhound walk on two legs?

Subquestions What type of animal is a greyhound? Does #1 walk on two legs?

Facts Greyhounds are dogs. Dogs walk on four legs.

Answer False

Figure 6: Example of a GSM8K data sample (top) and StrategyQA data sample (bottom).