Augmenting Math Word Problems via Iterative Question Composing

Haoxiong Liu[1*†], **Yifan Zhang**[1*], **Yifan Luo**[1 2], **Andrew C Yao**[1 2]

[1]Institute for Interdisciplinary Information Sciences, Tsinghua University,
[2]Shanghai Qi Zhi Institute
{liuhx20,zhangyif21,luoyf24}@mails.tsinghua.edu.cn, andrewcyao@tsinghua.edu.cn

Abstract

Despite the advancements in large language models (LLMs) for mathematical reasoning, solving competition-level math problems remains a significant challenge, especially for open-source LLMs without external tools. We introduce the MMIQC dataset, comprising a mixture of processed web data and synthetic question-response pairs, aimed at enhancing the mathematical reasoning capabilities of base language models. Models fine-tuned on MMIQC consistently surpass their counterparts in performance on the MATH benchmark across various model sizes. Notably, Qwen-72B-MMIQC achieves a 45.0% accuracy, exceeding the previous open-source state-of-the-art by 8.2% and outperforming the initial version GPT-4 released in 2023. Extensive evaluation results on Hungarian high school finals suggest that such improvement can generalize to unseen data. Our ablation study on MMIQC reveals that a large part of the improvement can be attributed to our novel augmentation method, Iterative Question Composing (IQC), which involves iteratively composing new questions from seed problems using an LLM and applying rejection sampling through another LLM.

Code —
https://github.com/iiis-ai/IterativeQuestionComposing

Datasets —
https://huggingface.co/datasets/Vivacem/MMIQC

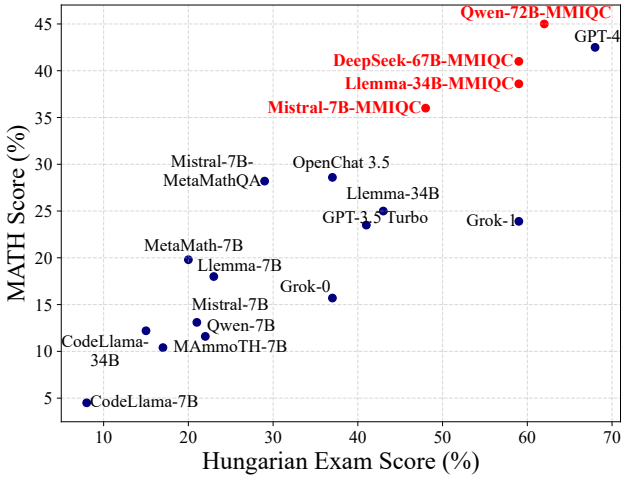

Figure 1: Performance evaluation of various LLMs on MATH (Hendrycks et al. 2021a) and the 2023 Hungarian National High School Mathematics Finals (Paster 2023a).

Introduction

Although large language models have been demonstrated to be powerful in various applications (Chen et al. 2021; Brown et al. 2020; Ouyang et al. 2022; Park et al. 2023; Huang et al. 2022b), solving math problems that require complex reasoning skills remains a challenging task. On MATH (Hendrycks et al. 2021b), a competition-level math problem benchmark containing algebra, calculus, geometry, combinatorics and number theory problems, open-source base LLMs such as the LLaMA family (Touvron et al. 2023a,b) fail to answer most of the problems correctly.

Previous work tries to enhance the mathematical reasoning abilities of base models by fine-tuning them on domain-specific data. Specifically, One line of work (Azerbayev et al. 2023; Lewkowycz et al. 2022) collects math corpora from the web and fine-tunes the models on them, which is also known as the procedure of continual pre-training (Cossu et al. 2022). Another line of work focuses on constructing synthetic data through rejection sampling (Yuan et al. 2023), distilling from GPT-4/GPT-3.5 (Yue et al. 2023) or question bootstrapping (Yu et al. 2023), and then use the generated question-response pairs to perform supervised fine-tuning in the way described in (Taori et al. 2023; Ouyang et al. 2022). However, there still exists a large performance gap between these fine-tuned models and the most advanced close-source models such as GPT-4 (OpenAI 2023) and Gemini-Ultra (Team et al. 2023). Given that simply adding more data does not always lead to better performance as shown in (Yu et al. 2023), how to bridge the gap remains an open challenge.

This work tackles the challenge by combining the two lines of work. On one hand, we reuse the high-quality corpora used in the pre-training stage during fine-tuning. Specifically, MMIQC contains around 1200k question-

*These authors contributed equally.
†Corresponding author.

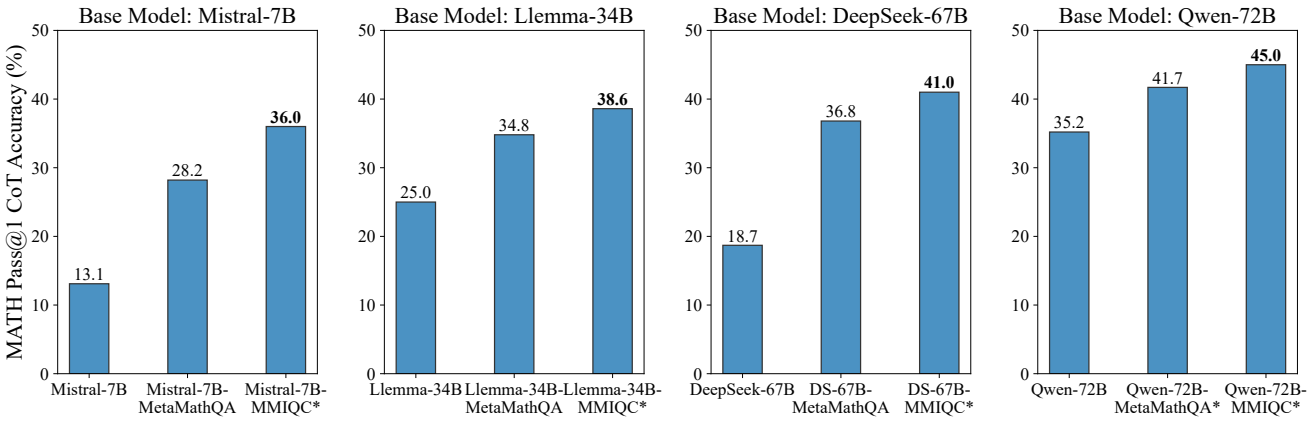

Figure 2: The performance of base models and their fine-tuned versions on MATH benchmark. The models remarked with an * are trained and evaluated by us. We can see that the models fine-tuned on MMIQC consistently outperform their counterparts by a clear margin.

response pairs we filtered and pre-processed from the web pages at math.stackexchange.com, which are included in the RedPajama dataset (Computer 2023). On the other hand, for the synthetic data part of MMIQC, we increase the diversity by using multiple kinds of augmentation methods listed below: 1) Prompting GPT-4 with an integrated version of the question bootstrapping prompts used in (Yu et al. 2023), and do rejection sampling with GPT-3.5-Turbo on both seed and augmented problems. 2) Using a modified prompt presented in (Liu et al. 2023) to ask GPT-4 to generate similar problems with answers given seed problems of the training set of MATH. Although the generated answers can be wrong, we perform rejection sampling on these problems as well. 3) Performing IQC (Iterative Question Composing) with 4 iterations in total. We iteratively ask GPT-4 to compose new questions from the given seed problems and do rejection sampling to filter those problems with answers aligned with GPT-3.5-turbo's answers. 4) Filtering a 204k subset of MetaMathQA (Yu et al. 2023) and adding it to the MMIQC dataset (More details on MMIQC will be introduced in Section).

We fine-tune several base models on MMIQC, resulting in models consistently achieving a large margin compared to their counterparts when evaluated on MATH, as shown in Figure 2. Specifically, the models Mistral-7B-MMIQC, Llemma-34B-MMIQC, DeepSeek-67B-MMIQC and Qwen-72B-MMIQC, which are obtained by fine-tuning Mistral-7B (Jiang et al. 2023), Llemma-34B (Azerbayev et al. 2023) and DeepSeek-67B (Bi et al. 2024) on MMIQC, achieve 36.0%, 38.6%, 41.0% and 45.0% accuracy on MATH, 5.8%, 3.8%, 4.2% and 3.3% higher than the counterpart models that are fine-tuned on MetaMathQA, respectively.

We also evaluate the models on the 2023 Hungarian national high school finals in mathematics (Paster 2023b). The results in Figure 1 suggest that the mathematical reasoning abilities the models acquire through being fine-tuned on MMIQC can generalize to unseen held-out problems.

We highlight our contributions as follows:

- We propose IQC (Iterative Question Composing), a data augmentation method that can iteratively generate diverse data starting from a seed dataset of math word problems.
- We release MMIQC, a mixture of processed web data and synthetic question-response pairs. In different model sizes, the models fine-tuned on MMIQC consistently outperform their counterparts by a clear margin on the MATH test set. Notably, Qwen-72B-MMIQC achieves a 45.0% accuracy, exceeding the previous open-source state-of-the-art[1] by 8.2% and outperforming the initial version GPT-4 released in 2023. Such improvement can generalize to unseen held-out data, e.g., Hungarian high school finals.
- Our results show that reusing the high-quality data in the pre-training corpora during the fine-tuning stage can improve the model performance, successfully combining the two lines of work of continual pre-training and supervised fine-tuning.
- Our results also show that using multiple augmentation methods to construct datasets for fine-tuning is an efficient way to boost the performance of LLMs.

Related Work

Base Large Language Models. Base large language models (LLMs) trained on massive corpora (e.g. 1.4T tokens of text for Llama (Touvron et al. 2023a)) from various sources with a simple auto-regressive next token prediction loss have

[1]As of the time of writing in January 2024, to the best of our knowledge, the open-source SOTA on MATH is the DeepSeek-67B-MetaMathQA model reported in (Wang et al. 2023a), which achieves 36.8% accuracy without external tool usage.

achieved great success in various natural language processing tasks (Radford et al. 2019; Brown et al. 2020; Touvron et al. 2023a,b; Jiang et al. 2023). Although these pre-trained models are not intended to serve for solving complex mathematical problems, (Wei et al. 2023) show that few-shot prompting can help the models answer a certain fraction of problems correctly. Nevertheless, to achieve better performance, fine-tuning the base LLMs on domain-specific data is required.

Fine-tuning Base LLMs on Mathematical Datasets. Current practice of fine-tuning base LLMs on mathematical datasets can be classified into two kinds: 1) continual pretraining (Lewkowycz et al. 2022; Azerbayev et al. 2023). This line of work typically collects billion-tokens level mathematical text data from the web, such as mathematical sub-sites of Stack Exchange and ArXiv, and fine-tune the model in the same way as that in the pre-training stage. 2) SFT (Supervised Fine-Tuning) (Yuan et al. 2023; Yu et al. 2023; Yue et al. 2023; Gou et al. 2023). Works in this line collect question-response pairs via various methods and train the models on their dataset in an Alpaca style. Due to the scarcity of publicly available high-quality question-response pairs datasets and the costly nature of manually composing math word problems, how to augment new data from the existing datasets becomes the focus of these works.

Our work is located in the middle between these two: MMIQC is a mixture of filtered pre-training corpus and question-response pairs generated using various augmentation methods.

Reasoning Frameworks for Solving Mathematical Problems. Much effort has been devoted to achieving a higher accuracy on math word problem benchmarks by designing different procedures of using the given LLMs to obtain the answers, which we refer to as *reasoning frameworks*. Among them, *Prompting-based* methods (Radford et al. 2019; Wei et al. 2023; Fu et al. 2022) play a significant role in activating the potential reasoning abilities for base LLMs through carefully designing the prompts shown to the models. Self-consistency (Wang et al. 2023b) samples multiple rationale paths for a model and then decides the answer by majority voting. In contrast of self-consistency, (Cobbe et al. 2021; Uesato et al. 2022; Lightman et al. 2023) use Outcome Reward Models (ORM) and Process Reward Models (PRM) trained on human annotations as verifiers to help select the answer with the highest score from the sampled reasoning paths of LLMs. Getting rid of the need of manual annotation, (Wang et al. 2023a) score a given reasoning step by estimating the potential of that step to lead to a correct answer automatically.

Some frameworks also include the use of plug-in tools and external APIs. Program-aided prompting (Gao et al. 2022; Yue et al. 2023) provides in-context samples containing Python codes for LLMs and uses code interpreters to execute the output to facilitate reasoning. Further, (Gou et al. 2023) interleave natural language rationales with Sympy[2] code and fine-tune the model on trajectories sampled from GPT-4 to follow their framework in two steps, namely imitation learning and output space shaping.

We note that our results in Figure 2 do not include multiple times of sampling, use of verifiers or code interpreters, thus cannot be directly compared with the results reported in these works.

[2]https://www.sympy.org/

Algorithm 1: Iterative Question Composing

Require: Question composing model π_q, rejection sampling model π_r, answer extractor defining $\simeq$, text templater $x(\cdot,\cdot)$ with inverse $x^{-1}(\cdot)$, initial seed dataset $S_0 = \{(q_i, a_i)\}_{i=1}^{n}$, total iterations K, question composing prompts $p_1, p_2, \ldots, p_K$, rejection sampling prompt p_r, maximum rejection samples per problem m

for $k = 1$ **to** K **do**
  Initialize $S_k \leftarrow \{\}$, $R_k \leftarrow \{\}$
  for all $(q, a) \in S_{k-1}$ **do**
    Sample $x' \sim \pi_q\left(\cdot | p_k \oplus x(q, a)\right)$
    Decompose $(q', a') \leftarrow x^{-1}(x')$
    Append $S_k \leftarrow S_k \cup \{(q', a')\}$
    for $j = 1$ **to** m **do**
      Sample $a^{(j)} \sim \pi_r(\cdot | p_r \oplus q')$
      if $a^{(j)} \simeq a'$ **then**
        Append $R_k \leftarrow R_k \cup \{(q', a^{(j)})\}$
      end if
    end for
  end for
  Combine $D_k \leftarrow S_k \cup R_k$
end for
Output Collections $D_1, D_2, \ldots, D_K$

Iterative Question Composing

Traditional data augmentation methods primarily concentrate on modifying either the questions or answers while retaining their original meanings, or generating similar problems, as discussed in (Yu et al. 2023) and (Liu et al. 2023). These methods, however, are limited in their diversity as they aim to create nearly identical problems. Our approach, termed **IQC (Iterative Question Composing)**, deviates from this by iteratively constructing more complex problems. It augments the initial problems, adding additional reasoning steps without altering their intrinsic logical structure. This ensures that the newly formed problems are organically linked to the original problem and elaborately tries to not include extraneous elements induced by a large transition of the reasoning process.

Notations. In our description, we refer to the combination of an LLM, its tokenizer, encoding/decoding methods, and a fixed generation configuration (inclusive of generation strategy, sampling temperature, and stopping criteria) simply as 'an LLM'. For an LLM π, we denote the output distribution given prompt $p \in \mathcal{A}^*$ as $\pi(\cdot|p)$. The concatenation of two text paragraphs p_1 and p_2 is represented as $p_1 \oplus p_2$.

The IQC process begins with specifying an LLM π_q for question composing and another model π_r for rejection sampling. An answer extractor is needed to derive answers from responses. Two responses r_1 and r_2 are considered equivalent, denoted $r_1 \simeq r_2$, if the same answer can be ex-

tracted from both. The process initiates with a seed dataset $S_0 = \{(q_i, a_i)\}_{i=1}^n$.

In iteration #1, we prompt π_q with $p_1 \oplus x(q, a)$ for each $(q, a) \in S_0$, where $x(\cdot, \cdot)$ is a text template transforming a question-response pair into text, and p_1 solicits a new question-answer composition. This yields a new dataset

$$S_1 = \{(q_i', a_i')\}_{i=1}^n,$$

where $(q_i', a_i') = x^{-1}(x_i')$ and $x_i' \sim \pi_q(\cdot|p_1 \oplus x_i)$ is the output for the ith sample. We further enhance S_1 by rejection sampling from π_r, resulting in

$$R_1 := \{(q_i', a_i^{(j)})|a_i^{(j)} \simeq a_i', i \in [n], j \in [m]\},$$

where $a_i^{(j)}$ are the sampled responses from $\pi_r(\cdot|p_r \oplus q_i')$. The dataset D_1 is then formed by uniting S_1 and R_1:

$$D_1 := S_1 \cup R_1.$$

For each subsequent iteration #k, the aforementioned procedure is repeated using S_{k-1} as the seed dataset, with varying question composing prompts p_k. The complete IQC process is delineated in Algorithm 1.

Seed Question:
Evaluate

$$(5a^2 - 13a + 4)(2a - 3)$$

for $a = 1\frac{1}{2}$.

Iter # 1 Question:
If $b = 2a - 3$ and $a = 1\frac{1}{2}$, what is the value of $(5a^2 - 13a + 4)b$?

Iter # 2 Question:
Given $b = 2a - 3$, $a = 1\frac{1}{2}$, and $c = 3b + 5$, find the value of $c(5a^2 - 13a + 4)$.

Iter # 3 Question:
Given $b = 2a - 3$, $a = 1\frac{1}{2}$, $c = 3b + 5$, and $d = c^2 - 4c$, find the value of $d + c(5a^2 - 13a + 4)$.

Iter # 4 Question:
Given $b = 2a - 3$, $a = 1\frac{1}{2}$, $c = 3b + 5$, $d = c^2 - 4c$, and $e = d^3 + 2cd - 7$, find the value of $e + c(5a^2 - 13a + 4) + d$.

Figure 3: An example of the questions composed via IQC by GPT-4 given 1 seed problem in MATH training set.

The MMIQC Dataset

In this section, we introduce how each part of MMIQC is constructed in detail.

Subset of MetaMathQA. The original MetaMathQA dataset is constructed by sampling GPT-3.5 for $k = 20$ times under a $T = 0.7$ temperature for each problem in the training set of MATH (Hendrycks et al. 2021a) and

You will be provided with 1 math problem and its solution and answer *(which are not guaranteed to be right)*. Please generate 1 new problem that (implicitly) contains the original problem as a subproblem or substep.

Your response should only contain one line text with 3 fields "problem", "solution" and "answer" in the same format as the given problem. The solution to the generated problem should be as brief as possible and **should not quote the conclusion of the original problem**. Ensure there is only one latex box in the solution and the answer is completely the same with the content in the box.

Please use two backslashes to represent one in the strings in order that it can be properly read in python. For example, you should write "\cdot" as "\\cdot".

Figure 4: The prompt we use to perform question composing in IQC. The *italics* part is not used in iteration #1.

GSM8K (Cobbe et al. 2021) dataset, or its bootstrapped versions. We restrict the number of samples for each completely same question to be 3 and 1 for MATH and GSM8K, respectively, to obtain a subset of MetaMathQA. This subset contains 112.2K GSM8K question-response pairs and 91.5K MATH pairs.

Answer Augmentation and Question Bootstrapping. We integrate the question bootstrapping methods used in (Yu et al. 2023) into a single prompt shown in Figure 5. Our motivation is that given GPT-4 is highly capable of natural language understanding, a few-shot prompting style used in (Yu et al. 2023) might suppress the diversity of the augmented questions. The seed dataset is constructed by the samples in the training set of MATH that do not contain Asymptote language in their question statements. We perform rejection sampling from GPT-3.5 on both the seed dataset and generated questions using the prompt shown in Figure 6, obtaining 66.5K question-response pairs. We use a temperature $T = 1.0$ for both question bootstrapping and rejection sampling.

Augmented Similar Problems. With the same seed dataset, we ask GPT-4 to generate 3 problems (with a solution, for rejection sampling) for 1 seed problem each time, using the prompt in Figure 7. This is different from the practice in (Liu et al. 2023), where they ask GPT-3.5 to generate 10 similar questions given 1 seed problem since we find that GPT tends to generate several almost the same problems regardless of the given seed problem when asked to generate up to 10 new problems. We use the stronger GPT-4 instead of GPT-3.5 considering rejection sampling needs the answer to the problem better to be correct. To control the cost, our prompt emphasizes that the solution should be as brief as possible. The total number of the augmented similar

DATA	# SAMPLES	#REPETITIONS	RATIO
METAMATHQA	203.7K	3	26.6%
ANSAUG & QB	66.5K	3	8.7%
AUGSIMILAR	38.2K	3	5.0%
IQC	**55.1K**	3	**7.2%**
MATHSTEX	1203.6K	1	52.5%

Table 1: The composition of MMIQC.

problems and the question-response pairs rejection sampled from them is 38.2K. The rejection sampling prompt is the same one in Figure 6 as well. We use a temperature $T = 1.0$ for both procedures.

Iterative Question Composing. We perform Iterative Question Composing for 4 iterations as described in Section . Specifically, we use GPT-4 for question composing model π_q with a $T = 0.7$ temperature and GPT-3.5 for rejection sampling model π_r with a $T = 1.0$ temperature. The question composing prompts and rejection sampling prompt are shown in Figure 4 and Figure 6, respectively. The text templater $x(\cdot, \cdot)$ we use is a program that transforms each question-response pair into JSON text format, with fields 'problem' and 'solution'. The seed dataset is also the samples in the training set of MATH that do not contain Asymptote code in their question statements. The resulting dataset has 55.1K samples in total.[3] We provide an example of the generated questions in different iterations corresponding to the same seed problem in Figure 3. We note that although some of the questions are not rigorously a sub-problem or sub-step of the corresponding problem in the previous iteration as required in our prompt, they are still valid questions that can increase the diversity of the dataset. We have checked the correctness of 100 randomly selected QA pairs generated by IQC and find that 85% of them are correct.

Mathematics Stack Exchange. We observe that in the OpenWebMath (Paster et al. 2023) dataset, the data from Mathematics Stack Exchange shows high quality and is most related to competition-level math. Motivated by this, we extract the data collected from Mathematics Stack Exchange in RedPajama (Computer 2023) and pre-process it into question-response pairs. For each Mathematics Stack Exchange page, we only retain the answer ranked first by RedPajama. Then we filter out the answer that does not contain a formula environment symbol '$'. This results in a dataset with 1203.6K question-response pairs.

Table 1 shows the make-up of MMIQC. When fine-tuning the models MMIQC contains 3 repetitions of the subsets mentioned above, except for the Mathematics Stack Exchange part. We shuffle the order of samples after combining the subsets.

[3]A part of the samples are generated by performing IQC for 2 iterations using a legacy version of prompts.

LR	1E-6	5E-6	**1E-5**	2E-5	5E-5	1E-4
MATH(%)	32.3	35.1	**36.0**	35.4	31.5	27.1

Table 2: Ablation study on the optimal learning rate. We fine-tune Mistral-7B on MMIQC with different maximal learning rate values and evaluate the fine-tuned models on MATH to decide the best candidate.

Experiments

Fine-tuning Setup

Our fine-tuning strategy mainly follows the practice of (Taori et al. 2023), except that we use a different prompt template to transform the question-response pairs. For a sample from Mathematics Stack Exchange, the corresponding prompt fed into the model during training is a simple concatenation of the question and response with two newline symbols. For a sample from other subsets, we additionally add a prefix 'Please solve the following problem and put your answer at the end with "The answer is: ".' to the question-response concatenation.

We use the HuggingFace transformers library (Wolf et al. 2019) for our fine-tuning experiments.

We fine-tune all models on MMIQC for 1 epoch, using a 3% warm-up ratio linear learning rate schedule. For the choice of maximum learning rate, we do a simple hyperparameter selection experiment shown in Table 2 and determine it to be 1e-5. We use the BFloat16 numerical format during training. Employing the DeepSpeed Zero-3 Stage (Rajbhandari et al. 2020), we fine-tune 7B models on one node of 8xA800 GPUs with micro batch-size at 8, and gradient accumulation at 4, 34B models on 2 nodes with micro batch-size at 4 and gradient accumulation at 4 and ∼70B models on 4 nodes with micro batch-size at 4 and gradient accumulation at 2, maintaining an effective batch size of 256. It takes around 14 hours, 61 hours and 90 hours to fine-tune 7B, 34B and ∼70B models under the setups stated above, respectively.

Model Evaluation

For a fair comparison, we first evaluate the fine-tuned models on MATH (Hendrycks et al. 2021a), a competition-level math word problems benchmark with 5000 test problems in a **zero-shot** setting. We prompt all our fine-tuned models with the test question with the prefix 'Please solve the following problem and put your answer at the end with "The answer is: ".', and extract the answer from the output using a modified version of the answer extractor provided in (Lewkowycz et al. 2022). We use a series of rules to infer whether the extracted answer is the same as the ground-truth answer, including a comparison using SymPy (Meurer et al. 2017). The complete results of our evaluation on MATH and a comparison with existing models are shown in Table 3.

For the evaluation on 2023 Hungarian national high school finals in mathematics, we use the few-shot prompt used in (Paster 2023b). We manually assess the grades for

You will be provided with 1 math problem in newline-delimited json format. Please augment 5 diverse problems from the given problem.

The way you augment a problem can be:
- Rephrase the problem.
- Change the scenario without modifying specific quantities.
- Set 1 number in the problem to an unknown variable, put the answer in the problem and ask what is the value of the variable. Ensure the generated problem is reasonable. Otherwise, skip this method.
- Other approaches that can ensure the correctness of the answer you provide to the augmented problem.

Your response should only contain text in newline-delimited json format, keeping the same with the given problem. Please use two backslashes to represent one in the strings.

Figure 5: The prompt we use to perform question bootstrapping for asking GPT-4.

You will be presented a mathematical problem. You should solve the problem step-by-step carefully. Present the final answer in latex boxed format, e.g., $\boxed{63\pi}$.

Figure 6: The prompt we use to do rejection sampling from GPTs.

You will be provided with 1 math problem in newline-delimited json format. Please generate 3 diverse new problems similar to the given problem.

Your response should only contain text in newline-delimited json format, keeping the same with the given problem. The solutions to the generated problems should be as brief as possible. Ensure there is only one box in the solution and the answer is completely the same with the content in the box. Please use two backslashes to represent one in the strings.

Figure 7: The prompt we use to generate questions similar to the seed problems for asking GPT-4.

every model according to the examiner instructions. The results shown in Figure 1 are the grades under a full mark of 117.

Ablation Study on Subsets of MMIQC

In order to understand the ratio of contribution to the improvement revealed in Table 3 of different subsets of MMIQC, we fine-tune Mistral-7B with a series of training sets constructed by gradually adding the subsets. When MathStackExchange is not added, we fine-tune for 3 epochs. When MathStackExchange is added to the training dataset, we mix 3 repetitions of other data with 1 repetition of the MathStackExchange, and fine-tune for only 1 epoch. It can be seen from Table 4 that

- Although our filtered subset of MetaMathQA is only half the size of the original dataset (which has 395K samples, more than the total number of samples of our synthetic data), the performance drop is only 1.8%. This shows that the $k = 20$ strategy in (Yu et al. 2023) results in some redundancy.
- Our Answer Augmentation & Question Boosting data help the fine-tuned model beat Mistral-7B-MetaMathQA, verifying our hypothesis that directly asking GPT to perform question bootstrapping is more efficient than providing few-shot examples to them.
- Our IQC method leads to a significant 3.1% improvement from a high accuracy of 31.5% with only 55.1K samples, showing its efficiency. Moreover, the later iterations of IQC also account for a certain ratio of improvement, proving that IQC is a method that can continuously generate new data that can help increase the diversity when added to the data generated in previous iterations.

Contamination Test

We check the n-gram matches for MMIQC to ensure that the improvement is not a result of direct memorization. We use the script provided by (Azerbayev et al. 2023) to check the n-gram matches between the synthetic part of the MMIQC and MATH test set. It turns out that for a 30-gram match check, there are 44 hits of match between the 'solution' field of MATH test set and the 'output' field of MMIQC, far fewer than the 168 hits between that of MATH test set and MATH training set. Moreover, we manually check these 44 hits and find that 43 among them belong to the case where intermediate steps of the solutions to similar but different questions collide, with the only exception being the question 'A regular polygon has interior angles of 144 degrees. How many sides does the polygon have?'. This almost rules out the possibility that fine-tuned models get memorization of solutions to the problems in the test set, indicating a very low risk of data contamination for MMIQC.

Model	FT-Dataset	Tool Usage?	Eval Method	MATH(%)
PROPRIETARY MODELS				
Minerva-540B (Uesato et al. 2022)	ArXiv+Web	No	maj1@64	50.3
GPT-4 (2023-0314) (Bubeck et al. 2023)	-	No	pass@1	42.5
Gemini-Ultra (Team et al. 2023)	-	No	pass@1	53.2
∼7B MODELS				
Llama-2-7B (Touvron et al. 2023b)	-	No	pass@1	2.5
Qwen-7B (Bai et al. 2023)	-	No	pass@1	11.6
Llemma-7B (Azerbayev et al. 2023)	Proof-Pile-2	No	pass@1	18.0
MetaMath-7B (Yu et al. 2023)	MetaMathQA	No	pass@1	19.8
Mistral-7B-MetaMathQA (Yu et al. 2023)	MetaMathQA	No	pass@1	<u>28.2</u>
Mistral-7B-MMIQC*	**MMIQC**	No	pass@1	**36.0**
MAmmoTH-Coder-7B (Yue et al. 2023)	MathInstruct	Code	pass@1	35.2
ToRA-Code-7B (Gou et al. 2023)	ToRA-Corpus	Code	pass@1	44.6
∼34B MODELS				
CodeLlama-34B	-	Code	pass@1	25.0
Llemma-34B-MetaMathQA	MetaMathQA	No	pass@1	<u>34.8</u>
Llemma-34B-MMIQC*	**MMIQC**	No	pass@1	**38.6**
Llemma-34B-MetaMathQA	MetaMathQA	Math-Shepherd	maj+verify1@256	47.3
ToRA-Code-34B (Gou et al. 2023)	ToRA-Corpus	Code	pass@1	50.8
∼70B MODELS				
Llama-2-70B (Touvron et al. 2023b)	-	No	pass@1	13.5
DeepSeek-67B (Bi et al. 2024)	-	No	Pass@1	18.7
Deepseek-67B-MetaMathQA	MetaMathQA	No	pass@1	<u>36.8</u>
Deepseek-67B-MMIQC*	**MMIQC**	No	pass@1	**41.0**
Deepseek-67B-MetaMathQA	MetaMathQA	No	maj1@256	45.4
Deepseek-67B-MetaMathQA	MetaMathQA	Math-Shepherd	maj+verify1@256	48.1
Qwen-72B (Bai et al. 2023)	-	No	pass@1	35.2
Qwen-72B-MetaMathQA*	MetaMathQA	No	pass@1	<u>41.7</u>
Qwen-72B-MMIQC*	**MMIQC**	No	pass@1	**45.0**

Table 3: A comparative analysis of the accuracies achieved by various models on the MATH benchmark. The models marked with an asterisk(∗) are fine-tuned and evaluated by us. Other results, unless otherwise cited, are derived from (Wang et al. 2023a). This comparison highlights the significant improvements our fine-tuned models demonstrate over existing solutions in mathematical problem-solving accuracy.

Data	# Samples	MATH(%)
MetaMathQA	395K	<u>28.2</u>
MetaMathQA (subset)	203.7K	26.4 (-1.8)
+ AnsAug & GQ	+66.5K	30.1 (+1.9)
+ AugSimilar	+38.2K	31.5 (+3.3)
+ **IQC** iter #1	+21.8K	33.0 (+4.8)
+ **IQC** iter #2	+16.0K	33.7 (+5.5)
+ **IQC** iter #3 & #4	+17.3K	34.4 (+6.2)
+ MathStackExchange	+1203.6K	**36.0** (**+7.8**)

Table 4: How different subsets of MMIQC affect the accuracy of the finetuned model on MATH.

Conclusion

In this work, we introduce a novel data augmentation method for math word problem datasets called IQC (Iterative Question Composing) and use it in the construction of our MMIQC dataset. Our evaluation results show that the models fine-tuned on MMIQC achieve new SOTAs on the MATH benchmark. The improvements of our models benefit from the diverse data sources of MMIQC and the effectiveness of IQC.

For future directions, we are interested in how to equip open-source models with the ability to compose questions, in order to perform IQC in a self-evolution style, similar to that in (Huang et al. 2022a). Besides, how to integrate the verification systems (Wang et al. 2023a; Liu et al. 2023) that are originally used to improve the accuracy during inference time into the procedure of IQC, is also an attractive topic.

Acknowledgements

We thank Yang Yuan, Kaiyue Wen, Xingyu Dang, and Jingqin Yang for their helpful discussions.

References

Azerbayev, Z.; Schoelkopf, H.; Paster, K.; Santos, M. D.; McAleer, S.; Jiang, A. Q.; Deng, J.; Biderman, S.; and

Welleck, S. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Bi, X.; Chen, D.; Chen, G.; Chen, S.; Dai, D.; Deng, C.; Ding, H.; Dong, K.; Du, Q.; Fu, Z.; et al. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. *arXiv preprint arXiv:2401.02954*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T. J.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. *ArXiv*, abs/2005.14165.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; McGrew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Computer, T. 2023. RedPajama: An Open Source Recipe to Reproduce LLaMA training dataset.

Cossu, A.; Tuytelaars, T.; Carta, A.; Passaro, L.; Lomonaco, V.; and Bacciu, D. 2022. Continual pre-training mitigates forgetting in language and vision. *arXiv preprint arXiv:2205.09357*.

Fu, Y.; Peng, H.; Sabharwal, A.; Clark, P.; and Khot, T. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; and Neubig, G. 2022. PAL: Program-aided Language Models. *arXiv preprint arXiv:2211.10435*.

Gou, Z.; Shao, Z.; Gong, Y.; Yang, Y.; Huang, M.; Duan, N.; Chen, W.; et al. 2023. ToRA: A Tool-Integrated Reasoning Agent for Mathematical Problem Solving. *arXiv preprint arXiv:2309.17452*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021a. Measuring Massive Multitask Language Understanding. *arXiv preprint arXiv:2009.03300*.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021b. Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS*.

Huang, J.; Gu, S. S.; Hou, L.; Wu, Y.; Wang, X.; Yu, H.; and Han, J. 2022a. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022b. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *ArXiv*, abs/2201.07207.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Lewkowycz, A.; Andreassen, A. J.; Dohan, D.; Dyer, E.; Michalewski, H.; Ramasesh, V. V.; Slone, A.; Anil, C.; Schlag, I.; Gutman-Solo, T.; Wu, Y.; Neyshabur, B.; Gur-Ari, G.; and Misra, V. 2022. Solving Quantitative Reasoning Problems with Language Models. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*.

Liu, B.; Bubeck, S.; Eldan, R.; Kulkarni, J.; Li, Y.; Nguyen, A.; Ward, R.; and Zhang, Y. 2023. TinyGSM: achieving¿ 80% on GSM8k with small language models. *arXiv preprint arXiv:2312.09241*.

Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roučka, v.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; and Scopatz, A. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3: e103.

OpenAI. 2023. GPT-4 Technical Report. *ArXiv*, abs/2303.08774.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Park, J. S.; O'Brien, J.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 1–22.

Paster, K. 2023a. Testing Language Models on a Held-Out High School National Finals Exam. https://huggingface.co/datasets/keirp/hungarian_national_hs_finals_exam.

Paster, K. 2023b. Testing Language Models on a Held-Out High School National Finals Exam. https://huggingface.co/datasets/keirp/hungarian_national_hs_finals_exam.

Paster, K.; Dos Santos, . M.; Azerbayev, Z.; and Ba, . J. 2023. OpenWebMath: An Open Dataset of High-Quality Mathematical Web Text. *arXiv preprint, forthcoming*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*.

Rajbhandari, S.; Rasley, J.; Ruwase, O.; and He, Y. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–16. IEEE.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023a. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Touvron, H.; Martin, L.; Stone, K. R.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D. M.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A. S.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I. M.; Korenev, A. V.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023b. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv*, abs/2307.09288.

Uesato, J.; Kushman, N.; Kumar, R.; Song, F.; Siegel, N.; Wang, L.; Creswell, A.; Irving, G.; and Higgins, I. 2022. Solving math word problems with process- and outcome-based feedback. arXiv:2211.14275.

Wang, P.; Li, L.; Shao, Z.; Xu, R.; Dai, D.; Li, Y.; Chen, D.; Wu, Y.; and Sui, Z. 2023a. Math-Shepherd: A Label-Free Step-by-Step Verifier for LLMs in Mathematical Reasoning. *arXiv preprint arXiv:2312.08935*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023b. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2023. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. *arXiv preprint arXiv:2309.12284*.

Yuan, Z.; Yuan, H.; Li, C.; Dong, G.; Tan, C.; and Zhou, C. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Yue, X.; Qu, X.; Zhang, G.; Fu, Y.; Huang, W.; Sun, H.; Su, Y.; and Chen, W. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.