LLM2LLM: Boosting LLMs with Novel Iterative Data Enhancement

Nicholas Lee*[1] Thanakul Wattanawong*[1] Sehoon Kim[1]
Karttikeya Mangalam[1] Sheng Shen[1] Gopala Anumanchipalli[1]
Michael W. Mahoney[1,2,3] Kurt Keutzer[1] Amir Gholami[1,2]
[1]UC Berkeley [2]ICSI [3]LBNL
{nicholas.lee, j.wat, sehoonkim, mangalam, s.sheng, gopala, mahoneymw, keutzer, amirgh}@berkeley.edu

Abstract

Pretrained large language models (LLMs) are currently state-of-the-art for solving the vast majority of natural language processing tasks. While many real-world applications still require fine-tuning to reach satisfactory levels of performance, many of them are in the low-data regime, making fine-tuning challenging. To address this, we propose LLM2LLM, a targeted and iterative data augmentation strategy that uses a teacher LLM to enhance a small seed dataset by augmenting additional data that can be used for fine-tuning on a specific task. LLM2LLM (1) fine-tunes a baseline student LLM on the initial seed data, (2) evaluates and extracts data points that the model gets wrong, and (3) uses a teacher LLM to generate synthetic data based on these incorrect data points, which are then added back into the training data. This approach amplifies the signal from incorrectly predicted data points by the LLM during training and reintegrates them into the dataset to focus on more challenging examples for the LLM. Our results show that LLM2LLM significantly enhances the performance of LLMs in the low-data regime, outperforming both traditional fine-tuning and other data augmentation baselines. LLM2LLM reduces the dependence on labor-intensive data curation and paves the way for more scalable and performant LLM solutions, allowing us to tackle data-constrained domains and tasks. We achieve improvements up to 24.2% on the GSM8K dataset, 32.6% on CaseHOLD, 32.0% on SNIPS, 52.6% on TREC and 39.8% on SST-2 over regular fine-tuning in the low-data regime using a Llama-2-7B student model. Our code is available at https://github.com/SqueezeAILab/LLM2LLM.

1 Introduction

Pretrained large language models (LLMs) have achieved impressive performance on various benchmarks and datasets that have previously required specialized neural network architectures. For many of these general benchmarks (Hendrycks et al., 2020; Zhong et al., 2023), LLMs are prompted with custom instructions or in-context examples.

However, in various real-world applications, these prompting strategies are not a one-size-fits-all solution. For instance, LLMs have a limit on the amount of input context they can process, thus limiting the number of in-context examples or instructions we can input to make the LLM follow a certain behavior. For simple tasks that are closely aligned with the data that the LLM was pretrained on, extensive prompting may not be necessary. However, applying LLMs to specialized domains (e.g., a specific medical field (Nori et al., 2023) or private data with niche protocols) can be more challenging, often requiring prohibitively long prompts to achieve adequate performance. Even if the prompt length does not exceed the limit, processing long prompts increases the latency and cost of each inference. Additionally, LLMs also tend to forget or ignore information in long contexts (Liu et al., 2023b), leading to potential accuracy drops even when the model can handle long input prompts. While Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has been developed to address some of these challenges, it may sometimes retrieve irrelevant passages or documents, which can potentially degrade the generation performance. Furthermore, RAG does not necessarily solve the latency and cost issue as processing a long input prompt may still be required.

A promising method for addressing this is fine-tuning. With the emergence of Parameter Efficient Fine-tuning (PEFT) (Hu et al., 2021; Mangrulkar et al., 2022), the computational resources required to fine-tune a task-specific LLM have decreased significantly. However, herein lies a new problem: successful fine-tuning requires enough training data. This can be challenging for some applications, where we only have access to a small amount

*Equal contribution

Findings of the Association for Computational Linguistics: ACL 2024, pages 6498–6526
August 11-16, 2024 ©2024 Association for Computational Linguistics

of task-specific data. Often, collecting, cleaning, and labeling additional data can be costly and time-consuming. So the key question is: *how should we increase the user's training data to be enough for fine-tuning?*

Data augmentation is a known method that could help effectively expand the training dataset. For natural language processing (NLP) tasks, one can use approaches such as synonym replacement, character replacement (e.g., by intentionally introducing spelling errors), random swapping, and back translation, just to name a few (Wei and Zou, 2019; Belinkov and Bisk, 2017; Coulombe, 2018; Zhang et al., 2018). However, these approaches fail to effectively expand the training data for fine-tuning LLMs in the case of new and specialized tasks, as we will show later in Section 4.3.

To address this, several recent papers have explored using an LLM to expand the fine-tuning dataset (Dai et al., 2023; Kumar et al., 2020; Zhou et al., 2023; Chen et al., 2023; Cao et al., 2023; Wei et al., 2023; Zhu et al., 2023). This approach has proven to be more effective than traditional data augmentation methods. However, these approaches often apply LLM-based data augmentation on all of the available training dataset, without considering the LLM's prediction accuracy on individual training data points. We have observed that for various reasoning tasks such as arithmetic and reading comprehension, the LLM correctly solves simpler examples in the fine-tuning dataset, but may struggle with harder examples. It will be sub-optimal to keep augmenting data points for which the LLM is already achieving high accuracy on.

To address these challenges, we introduce LLM2LLM, a new targeted and iterative data augmentation framework that uses a teacher LLM to expand the training dataset, with a targeted and iterative approach. In more detail, we make the following contributions:

- We propose LLM2LLM, a targeted and iterative LLM-based data augmentation technique that efficiently and effectively augments small task-specific datasets. LLM2LLM achieves this by (1) fine-tuning a student LLM on the initial dataset, (2) evaluating on the training data and extracting data points which the model got incorrect after training, and (3) using a Self-Instruct (Wang et al., 2023) style data augmentation to augment these data points, which are then added back into the training data (Section 3.1).
- We benchmark LLM2LLM on randomly sampled subsets of GSM8K (Cobbe et al., 2021), CaseHOLD (Zheng et al., 2021), SNIPS (Coucke et al., 2018), TREC (Li and Roth, 2002) and SST-2 (Socher et al., 2013) in order to evaluate the effectiveness of our approach in the low-data regime (Section 4.2). Here, we get up to a 24.2% improvement on GSM8K, 32.6% on CaseHOLD, 32.0% on SNIPS, 52.6% on TREC, and 39.8% on SST-2 (Table 1).
- We conduct a series of ablations studies comparing LLM2LLM to several existing baselines as well as to variants of LLM2LLM to evaluate the effectiveness of our design decisions (Section 4.5). We observe that both the iterative and targeted nature of LLM2LLM are critical to improving model performance.

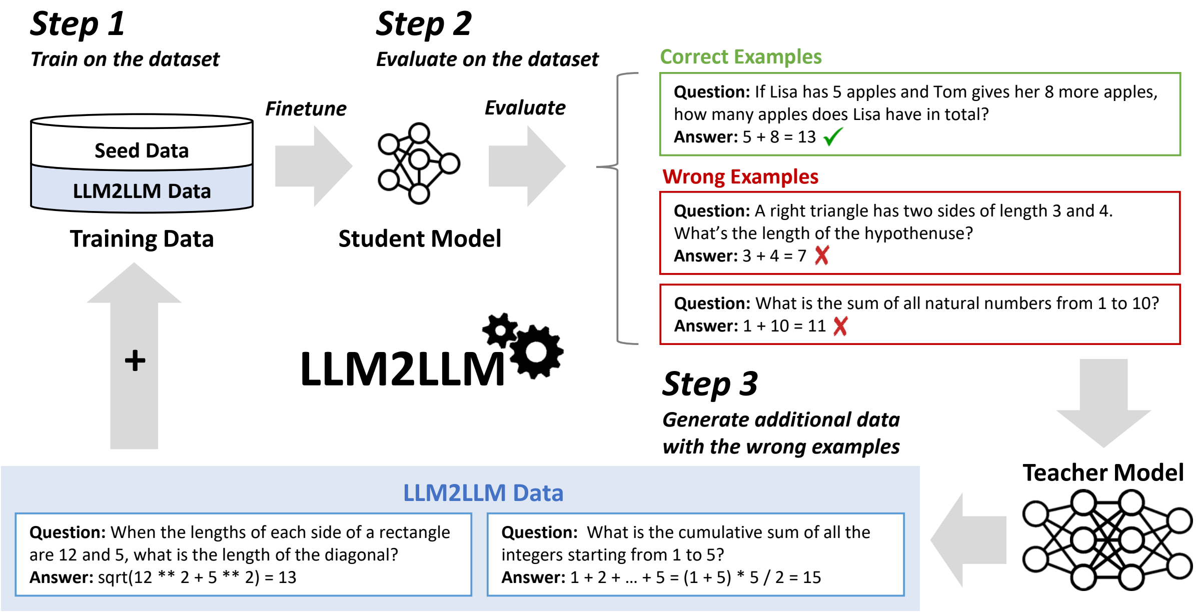
2 Background and Related Work

2.1 Instruction Following LLMs

The earliest works (Wei et al., 2021; Longpre et al., 2023; Chung et al., 2022; Aribandi et al., 2021; Sanh et al., 2021; Muennighoff et al., 2023; Wang et al., 2022b; Mishra et al., 2022; Wang et al., 2022a; Xu et al., 2022) in instruction fine-tuning involved gathering and processing different existing NLP datasets in order to improve the performance of LLMs on a wide range of tasks. Self-Instruct (Wang et al., 2023) removed the reliance on existing datasets by introducing a framework for bootstrapping instruction datasets with the outputs of the model itself. Follow-up work (Ouyang et al., 2022; Taori et al., 2023; Geng et al., 2023; Chiang et al., 2023; Xu et al., 2023; Mukherjee et al., 2023; Mitra et al., 2023; Kang et al., 2023; Nori et al., 2023) took advantage of stronger models (Achiam et al., 2023; Touvron et al., 2023a,b) in order to fine-tune stronger general-purpose instruction-following models.

2.2 Self-Improving LLMs

Various early works (Zelikman et al., 2023; Haluptzok et al., 2023; Zelikman et al., 2022; Madaan et al., 2023; Gulcehre et al., 2023; Singh et al., 2023) explore using self-improvement for fine-tuning LLMs. These works generally filtered the outputs of the model before fine-tuning it on its own outputs. LLM2LLM differs from these methods, as we do not directly fine-tune on the outputs of our own model, and we employ a teacher model to provide feedback in the form of synthetic data.



Question: If Lisa has 5 apples and Tom gives her 8 more apples, how many apples does Lisa have in total?
Answer: 5 + 8 = 13 ✓

Question: A right triangle has two sides of length 3 and 4. What's the length of the hypothenuse?
Answer: 3 + 4 = 7 ✗

Question: What is the sum of all natural numbers from 1 to 10?
Answer: 1 + 10 = 11 ✗

Question: When the lengths of each side of a rectangle are 12 and 5, what is the length of the diagonal?
Answer: sqrt(12 ** 2 + 5 ** 2) = 13

Question: What is the cumulative sum of all the integers starting from 1 to 5?
Answer: 1 + 2 + ... + 5 = (1 + 5) * 5 / 2 = 15

Figure 1: LLM2LLM: Boosting LLMs with Novel Iterative Data Enhancement. One iteration of LLM2LLM begins with training and evaluating the model on the training data. Incorrect answers from the training data are used as inputs to generate extra samples with similar styles to the teacher model. Then, a new student model is trained using a combination of the old training data and newly generated samples. After the model is fine-tuned, we evaluate and find questions that the model got incorrect. The teacher model is used to generate additional data points based on the wrong examples, which test for similar concepts and ideas. These synthetic data points are folded back into training dataset. This process then repeats, training the student model on increasingly targeted data points.

Concurrent with our work, several papers have been recently published that use an iterative approach to improving LLMs (Chen et al., 2024; Anil et al., 2023; Burns et al., 2023; Li et al., 2023b; Yuan et al., 2024). These works combine ideas from Reinforcement Learning (RL) and Self-Play (Samuel, 2000; Tesauro et al., 1995) in order to iteratively build stronger LLMs by fine-tuning on the outputs of the model itself. LLM2LLM is distinguished by how it focuses on the low-data regime for task-specific fine-tuning of LLMs whereas others are attempting to create stronger general-purpose LLMs. In addition, our technique exclusively uses the data points that the model got incorrect during training, and it uses the teacher model to augment these data points. Instead of providing feedback in the form of a critique or rationale, the teacher model's feedback is only in the form of synthetic data points, which simplifies the training pipeline.

2.3 Data Augmentation

Data augmentation has been long studied in NLP. Early work augmented at the character (Belinkov and Bisk, 2017; Coulombe, 2018) and word (Wei and Zou, 2019) level. Notably, Easy Data Augmentation (EDA) (Wei and Zou, 2019) was a popular early method that used word level augmentations: synonym replacement, random insertion, swap, and deletion to augment data for text classification. We refer the reader to (Feng et al., 2021) for a more complete summary of data augmentation in NLP.

A popular new approach is to use LLMs themselves to synthesize new training data (Deng et al., 2023a; Prasad et al., 2023; Fu et al., 2023b; Dai et al., 2023; Ubani et al., 2023; Fang et al., 2023; Liu et al., 2023a; Yu et al., 2023; Kumar et al., 2020; Yoo et al., 2021; Wang et al., 2021; Ding et al., 2023; Li et al., 2023a; Liang et al., 2023). A noteworthy example is AugGPT (Dai et al., 2023), which used ChatGPT to rephrase text to augment text classification tasks.

Many of these techniques generate very large amounts of synthetic data. Recent work (Chen et al., 2023; Cao et al., 2023; Wei et al., 2023; Zhou et al., 2023) found that one could replicate the results of fine-tuning on these large datasets with significantly smaller subsets.

3 Methodology

We assume that we are given an LLM model $\mathcal{M}$ (e.g., GPT-3.5 or Llama-2-7B) that is pre-trained on some source dataset (e.g., Common Crawl). The goal is to adapt $\mathcal{M}$ (hereon called the student model) to a new target domain by using a small

Algorithm 1 LLM2LLM: Boosting LLMs with Novel Iterative Data Enhancement. Given a seed dataset D^0, we finetune the model $\mathcal{M}^i_{\text{student}}$, evaluate, and extract training set data points that the model gets wrong. These are used to generate new training data points using the teacher model $\mathcal{M}_{\text{teacher}}$ for the next step.

```
 1: procedure LLM2LLM($\mathcal{M}^0_{\text{student}}$, $\mathcal{M}_{\text{teacher}}$, $D^0$)
 2:     $i \leftarrow 0$
 3:     while $i < n$ do
 4:         $\mathcal{M}^i_{\text{student}} \leftarrow$ Finetune($\mathcal{M}^0_{\text{student}}$, $D^i$)
 5:         $E^i \leftarrow$ Evaluate($\mathcal{M}^i_{\text{student}}$, $D^0$)      ▷ Evaluate on seed data
 6:         $W^i \leftarrow$ Filter($E^i$,$D^0$)                      ▷ Keep wrong answers
 7:         $A^i \leftarrow$ Generate($\mathcal{M}_{\text{teacher}}$, $W^i$)    ▷ Augment using teacher
 8:         $D^{i+1} \leftarrow D^i + A^i$                          ▷ Append to data
 9:         $i \leftarrow i + 1$
10:     end while
11:     Evaluate $M^*_{\text{student}}$
12: end procedure
```

seed dataset D, where D potentially has unseen characteristics, compared to the pre-trained dataset (e.g., a medical dataset with specific terminology, or a private database with specific characteristics). In this case, the model's zero-shot or fine-tuned performance is likely to be unsatisfactory. While strategies to address this challenge have been explored, e.g., through enhanced few-shot learning methods as discussed in Section 2, here we strictly focus on enriching the provided target dataset D with an LLM. This method is orthogonal to the aforementioned techniques, offering a complementary solution that can be applied alongside them.

To enrich D, AugGPT (Dai et al., 2023) has introduced a promising approach that generates additional augmented data by applying a prompted LLM to all available data points in the target training dataset. However, this method falls short by indiscriminately augmenting data without considering the student model's varying performance across different data points. For instance, the model may easily solve the majority of the dataset, but it may struggle with a small subset of more challenging examples. In this case, rather than indiscriminately expanding the dataset by replicating simpler cases, a better augmentation strategy would be to generate more data points that align conceptually with these challenging examples. This is because the former approach could lead to longer training time without noticeable performance improvement.

Here, we propose a more general formulation of an LLM-based data augmentation pipeline that addresses the aforementioned limitation. To do so, we consider the following iterative process:

$$D^{n+1} = f(\mathcal{M}_{\text{teacher}}, \mathcal{M}_{\text{student}}, D^n, \cdots, D^0). \qquad (1)$$

In Equation (1), $\mathcal{M}_{\text{teacher}}$ is the teacher model, $\mathcal{M}_{\text{student}}$ is the student model (potentially being fine-tuned in many iterations), n refers to the n^{th} step of data augmentation, D^{n+1} is the new training dataset at the next iteration, and f is the data-generation algorithm. At each step, the teacher model has access to how the student model performs at the n^{th} step (e.g., correct/incorrect labels, or possibly prediction distributions for white-box models), and based on that it can edit training data points for the next iteration.

Note that LLM2LLM is different from knowledge distillation (Hinton et al., 2015). Knowledge distillation is generally applicable to cases where the teacher model has high accuracy on the target data. In contrast, in this case, it is possible that the teacher model also performs sub-optimally on the target data (e.g., in the private database case, where the teacher lacks domain-specific knowledge). However, if the teacher model has enough reasoning capability to produce conceptually similar but semantically different examples when it is given both the prompt and answer, then our framework can improve performance.

In LLM2LLM, we consider a specific instantiation of Equation (1), as discussed next.

3.1 LLM2LLM

The end-to-end algorithm of LLM2LLM is presented in Algorithm 1. Inspired by Self-Instruct (Wang et al., 2023), we use the teacher model $\mathcal{M}_{\text{teacher}}$ to generate synthetic data from the data points that the model got incorrect during training in order to target these deficiencies in the student model. In more detail, we first train the baseline student model $\mathcal{M}_{\text{student}}$ on the provided target data D^0, and we evaluate its performance (lines 4-5 of Algorithm 1). We then filter the results and keep the incorrect training examples that the student model struggled to answer correctly (E^i in line 6). Then the teacher model is prompted to create additional training data points that are conceptually aligned but semantically different (line 7, see Section B.4 for specifics on the prompt). The teacher model does not necessarily need to be bigger, although that could potentially improve performance. The primary requirement for the teacher model is to have reasoning capability to be able to follow the

Dataset	% Data	# Seed Examples	# Augmented	Test Accuracy (%)	
				Baseline	LLM2LLM
GSM8K	0	0	0	0.00[1]	**N/A**
	1	74	391	0.99	**19.56**
	2	149	802	1.52	**25.70**
	5	373	1641	9.63	**27.07**
	10	747	2573	21.27	**30.93**
	20	1494	4028	25.70	**35.03**
	50	3737	8252	33.89	**38.67**
	100	7473	14925	36.01	**41.24**
CaseHOLD	0	0	0	12.28	**N/A**
	0.5	225	490	33.94	**66.50**
	1	450	751	46.25	**70.97**
	2	900	580	69.44	**74.97**
	5	2250	423	74.14	**76.83**
	10	4500	505	77.03	**78.21**
	20	9000	1100	78.00	**78.97**
	50	22500	2709	80.39	**82.92**
	100	45000	5805	87.94	**88.14**
SNIPS	0	0	0	11.86	**N/A**
	0.5	70	38	60.14	**92.14**
	0.8	105	109	69.71	**93.71**
	1.0	140	91	85.43	**93.86**
TREC	0	0	0	11.20	**N/A**
	1.1	60	105	26.20	**78.80**
	1.6	90	22	80.80	**90.20**
	2.2	120	44	81.20	**91.20**
SST-2[2]	0	0	0	27.06	**N/A**
	0.02	20	44	52.87	**92.66**
	0.04	30	46	62.04	**93.00**
	0.06	40	14	82.80	**94.04**

Table 1: LLM2LLM on datasets under evaluation. The *% Data* and *# Seed Examples* columns indicate the percentage and number of data points respectively that were sampled from the original training data as seed data. The *# Augmented* column shows the number of data points created by LLM2LLM. The last column (*Test Accuracy %*) shows the baseline accuracy from fine-tuning with the original seed examples (Baseline), as well as when training with augmented data added to the dataset (LLM2LLM). Overall, test accuracy improves significantly with LLM2LLM, especially in low data regimes.

data augmentation instruction, and the ability to create data points similar to the incorrect examples. This process is schematically illustrated in Figure 1.

A subtle but important design decision in LLM2LLM is that we only use examples from the seed data when prompting the teacher model to generate additional data points. This is similar to Alpaca (Taori et al., 2023), but unlike Evol-Instruct (Xu et al., 2023). There are two main reasons for this. First, our approach prevents data degradation from multiple augmentation iterations. Early experiments revealed that while the teacher model could generate high-quality augmentations, some examples contained logical errors. Therefore, further augmentation applied to these examples could potentially propagate the error, degrading the quality of the dataset over time. This is highlighted in our ablation studies in Table 4, where using both seed and synthetic data for data augmentation leads to an accuracy drop.

Second, this approach limits the amount of new data being generated overall. Suppose that the original seed dataset is of size n, and at each iteration, the student model gets p_i proportion of the training dataset D^i wrong, where $0 < p_i < 1$. If we include the augmented data into the seed data for data generation, then the size of the dataset D^j at step j will be

$$|D^j| = n \prod_{i=0}^{j} (1 + p_i) \geq n(1 + p_{\text{min}})^j.$$

This has a lower bound that grows exponentially with each step. Limiting the input wrong answers

[1] The 0% here is because our prompting does not include the formatting that we used to extract the answer with. Adding some instructions to make sure the output format is correct results in a 0.30% test accuracy, which is in line with the rest of the results in Table 1.

[2] SST-2 has no test set, therefore we evaluate on the dev set instead, see Section A.3.

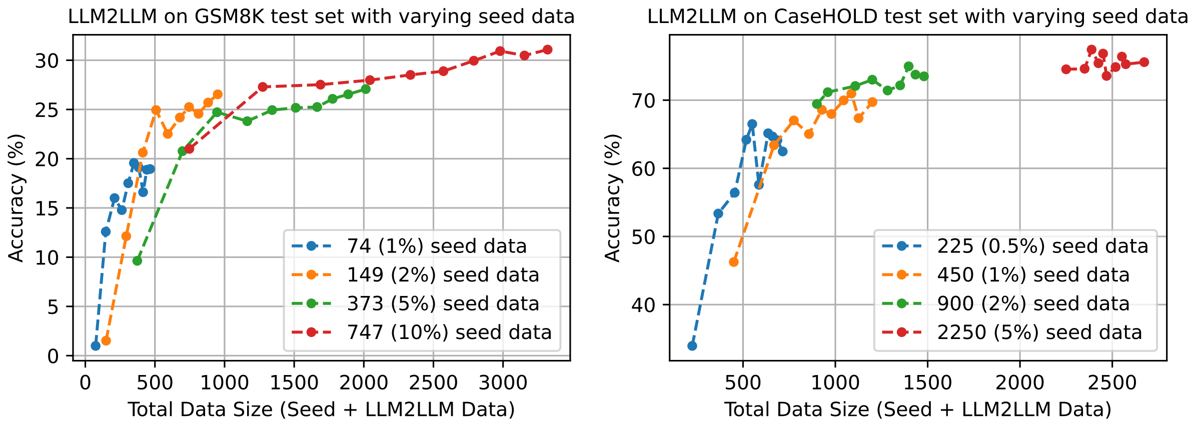


Figure 2: LLM2LLM on GSM8K (left) and CaseHOLD (right) with various seed data sizes. Each line shows the test accuracy of the finetuned Llama-2-7B model with each step of LLM2LLM with varying seed dataset size. The first (left-most) data point on each line represents finetuning only on the seed data. Each point afterward corresponds to the performance after one more iteration of LLM2LLM. The total data size (x-axis) represents the total amount of seed plus LLM2LLM data that was used to train the model at that step. By applying LLM2LLM with low amounts of seed data and iteratively improving the training dataset, we can attain significant performance improvements. In particular, we can see that running LLM2LLM can match or even exceed the performance of simply annotating more real data in some cases (detailed breakdown provided in Table 1).

W^i during dataset generation to only data from the original seed data allows us to bound the total number of training data points to

$$|D^j| = n + \sum_{i=0}^{j} np_i \leq n(1 + jp_{\text{max}}),$$

which has an upper-bound that grows linearly with the number of steps. The empirical evaluations shown in Section 4.5.2 (Table 4) corroborates this.

4 Results

4.1 Experimental Setup

To evaluate the performance of LLM2LLM, we applied our framework to fine-tune Llama-2-7B on various target datasets, including GSM8K (Cobbe et al., 2021), CaseHOLD (Nori et al., 2023), SNIPS (Coucke et al., 2018), TREC (Li and Roth, 2002) and SST-2 (Socher et al., 2013). We subsampled these datasets with different sampling rates from 0.02% to 50% to evaluate performance across different low-data regimes. Our teacher model for these results is GPT-3.5 (1106 release) unless otherwise specified. We considered several other teacher models, including GPT-3.5, GPT-4-Turbo, Llama-2-70B (Touvron et al., 2023b), and Airoboros-L2-70B (Durbin, 2023) in Section 4.4. We include a more detailed experimental setup in Section A. Additionally, we conducted additional experiments (Section B.6) to ensure that the augmented data from the teacher models differs from the test dataset used to evaluate final model accuracy. This addresses the issue of potential test data leakage that could have happened if the teacher model had been trained on similar data.

4.2 Main Results

Here, we discuss LLM2LLM's performance with varying amount of training data by presenting results for fine-tuning Llama-2-7B on GSM8K using GPT-3.5 as the teacher model. We then discuss how these trends extend to different datasets (Table 1).

The final model accuracy after applying 10 iterations of LLM2LLM is given in Table 1. For a low-data regime with 74 available examples (i.e., 1% of the GSM8K training dataset), vanilla fine-tuning achieves only 0.99% test accuracy. However, LLM2LLM boosts the accuracy to 19.56% by iteratively generating 391 additional examples based on data points where the model makes mistakes. With slightly more available data of 149 seed examples (i.e., 2% of the training dataset) we can achieve 25.70% accuracy. As shown in the baseline accuracy with 20% data in Table 2, we would need over 10× more training data points to match this accuracy if we only rely on vanilla fine-tuning. We also highlight that LLM2LLM can lead to noticeable gains with data-sufficient regimes (e.g., 100% data), albeit at a smaller improvement over the baseline compared to lower-data regimes.

We observe a similar trend for CaseHOLD, SNIPS, TREC, and SST-2, where LLM2LLM helps improve performance in the low-data regime.

Dataset	Technique	# Seed	Total Aug.	Acc. (%)
GSM8K	Fine-tuning	100	0	1.59
	EDA		500	15.16
	AugGPT		500	18.12
	More Data		471	19.86
	LLM2LLM		471	**23.73**
CaseHOLD	Fine-tuning	100	0	28.78
	EDA		200	62.19
	AugGPT		200	63.42
	More Data		198	37.11
	LLM2LLM		198	**64.50**
SNIPS	Fine-tuning	70	0	60.14
	EDA		70	91.43
	AugGPT		70	89.86
	More Data		70	89.00
	LLM2LLM		38	**92.14**
TREC	Fine-tuning	60	0	26.20
	EDA		120	72.40
	AugGPT		120	32.80
	More Data		138	**89.20**
	LLM2LLM		135	78.80
SST-2	Fine-tuning	20	0	52.87
	EDA		40	63.19
	AugGPT		40	88.07
	More Data		40	72.94
	LLM2LLM		44	**92.66**

Table 2: Results of LLM2LLM compared to other baseline methods. Column *Technique* refers to the augmentation method used as described in Section A.3. Column *# Seed* indicates the size of the initial seed dataset. Column *Total Aug.* represents the total amount of LLM2LLM data generated. For GSM8K and CaseHOLD, we randomly sample 100 data points while for SNIPS, TREC, and SST-2, we sample 10 samples per class. Column *Acc.* indicates the final test accuracy. Clearly, LLM2LLM outperforms all of synthetic baselines; even sometimes when adding in more real data from the dataset.

Interestingly, LLM2LLM generally generates proportionally more augmented data for GSM8K than other datasets. This is because the baseline accuracy is lower for GSM8K overall, suggesting that it is a more difficult dataset compared to the others. However, in all cases, we find that LLM2LLM helps recover a high-performing model.

In Figure 2, we also illustrate how the baseline accuracy improves on GSM8K and CaseHOLD with each iteration of applying LLM2LLM. We can observe a rapid increase in test accuracy in the first few iterations of LLM2LLM, especially in lower-data regimes.

4.3 Comparison with Other Augmentation Methods

In Table 2, we compare our method against other augmentation techniques, including EDA (Wei and Zou, 2019) and AugGPT (Dai et al., 2023). We also compare against adding more data from the unseen training set. The details of all augmentation methods we used in our comparison are provided in Section A.3.

On GSM8K, LLM2LLM outperforms naive fine-tuning by over 20%, EDA by over 8%, and AugGPT by over 5%. Similarly, on CaseHOLD, LLM2LLM outperforms the fine-tuning baseline by approximately 35%, EDA by 2.3%, and AugGPT by 1.1%. These improvements, particularly in comparison to AugGPT, can be attributed to LLM2LLM's capability to generate more targeted examples based on where the model struggles, as opposed to AugGPT which augments data indiscriminately. This allows for more effective and targeted use of the augmented data budget.

4.4 Choice of Teacher Model

Thus far, we have illustrated LLM2LLM's performance with GPT-3.5 as the teacher model, but other LLMs can serve this role as well. A stronger teacher model is expected to yield higher-quality augmentation and, consequently, higher accuracy. Table A.1 demonstrates the LLM2LLM's accuracy with GPT-4-Turbo, Llama-2-70B, and Airoboros-L2-70B as the teacher model on GSM8K. With 74 seed data examples, LLM2LLM only achieves 11.8% accuracy with Llama-2-70B, which can be contrasted with 15.0% with Airoboros and 19.8% with GPT-4-Turbo. This aligns with our expectation, as GPT-4-Turbo's mathematical reasoning is known to be better than the other models, being generally on par with that of GPT-4 (Fu et al., 2023a; Deng et al., 2023b). The qualitative analysis of augmented data using different models (Figure B.16) further supports this, showing that Llama and Airoboros models produce less varied data than GPT-3.5 or GPT-4-Turbo.

4.5 Ablation Studies

Here, we provide ablation studies to justify the design decisions we made in LLM2LLM.

4.5.1 Iterative Augmentation vs One-Shot Augmentation

We first evaluate the efficacy of iterative augmentation versus adding all augmented data at once. To evaluate this, we compare the final accuracy achieved by augmenting data over 10 iterations against adding the equivalent amount of data in

Dataset	Steps	Total Aug.	Acc. (%)
GSM8K	1 (one-shot)	490	16.30
	10 (iterative)	471	**23.73**
CaseHOLD	1 (one-shot)	276	59.94
	10 (iterative)	198	**64.50**

Table 3: Ablation on the iterative nature of LLM2LLM with 100 seed data points. *Steps* refers to the total number of augmentation steps in LLM2LLM. For the case of 1 iteration, we prompt the teacher model to generate more samples all at once, whereas in the 10 steps case the teacher model only generates 1 new data point per wrong example. The results clearly show that the latter iterative approach results in better performance.

Dataset	Only Aug. Seed Data	Total Aug.	Acc. (%)
GSM8K	✗	4302	18.32
	✓	471	**23.75**
CaseHOLD	✗	351	63.75
	✓	198	**64.50**

Table 4: Ablation study on whether to augment previously generated LLM2LLM data. *Only Aug. Seed Data* refers to augmenting only the seed data vs. also re-augmenting the augmented data. *Total Aug.* refers to the total number of augmentations generated over 10 steps of LLM2LLM.

a single iteration, for both the GSM8K and CaseHOLD datasets. As shown in Table 3, using a single augmentation step with a larger amount of augmented data significantly underperforms the alternative of executing 10 iterative steps of LLM2LLM with a smaller number of augmentations per iteration. In particular, on GSM8K, augmenting one data point per example over 10 steps yields a 7.4% higher accuracy than augmenting five data points per example in a single step. Similarly, on CaseHOLD, iterative augmentation of one data points per example over 10 steps results in a 4.6% improvement over a one-shot augmentation with four data points per example. This justifies the LLM2LLM's iterative augmentation approach that generates one data point per each incorrectly answered example.

4.5.2 Data Augmentation with Seed Data vs Augmented Data

In each iteration, LLM2LLM evaluates the student model's performance only on the original seed dataset and generates augmented data from incorrect seed examples. However, a possible alternative is performing evaluation and data augmentation using both seed and previously augmented data. The latter often leads to sub-optimal performance as well as excessive amounts of total augmented data points, as we demonstrate in Table 4. On GSM8K, generating augmented data from the previous iteration's augmented data yields 18.3% accuracy, while using the seed data for further augmentation improves the accuracy to 23.75%. We observe a similar trend for CaseHOLD. As discussed in Section 3.1, a potential reason for the performance drop, when using augmented data for further augmentation, has to do with a deviation from the original data distribution.

Dataset	From-scratch Fine-tuning	Total Aug.	Acc. (%)
GSM8K	✗	230	14.71
	✓	471	**23.75**
CaseHOLD	✗	154	60.50
	✓	198	**64.50**

Table 5: Ablation study on whether to fine-tune from scratch or to do continuous fine-tuning. *From-scratch Fine-tuning* refers to whether we fine-tune the base model from scratch vs. fine-tune the previous step's model. *Total Aug.* refers to the total number of augmentated examples generated over 10 steps of LLM2LLM.

4.5.3 From-scratch Fine-tuning vs Continuous Fine-tuning

Another key decision for LLM2LLM is whether to continue fine-tuning from the last iteration's checkpoint (i.e. continuous fine-tuning) or to restart fine-tuning from the pre-trained model at each iteration (i.e. from-scratch fine-tuning). Considering the non-convex nature of the optimization target and complex loss landscapes, this decision is not necessarily obvious. Nevertheless, as shown in Table 5, we observe that from-scratch fine-tuning consistently and significantly outperforms continuous fine-tuning, with up to 9% accuracy improvement. The inferior performance of continuous fine-tuning can be attributed to a potential overfitting to small seed data over multiple iterations of fine-tuning, especially in lower-data regimes where the seed data is small. This can be alleviated by restarting fine-tuning from scratch in each iteration with sufficient augmented data appended to the seed data to form the training dataset.

5 Conclusion

We have introduced LLM2LLM, an adaptive and iterative LLM-based data augmentation framework that uses LLMs to scale up smaller fine-tuning datasets in lieu of manually generating more data. This framework substantially reduces the amount of real data needed, and it allows us to efficiently scale the dataset with synthetic data that can match or even exceed the effect of hand-collecting more data. The method is effective because of the iterative and targeted nature of the process, which allows us to boost the signal from data points that the LLM gets wrong. As a result, we were able to achieve a 24.2% improvement on GSM8K, 32.6% on CaseHOLD, 32.0% on SNIPS, 52.6% on TREC, and 39.8% on the SST-2 dataset in the low-data regime using a Llama-2-7B student model. Future work can focus on tuning the hyperparameters of our framework as well as incorporating our approach with other LLM techniques such as prompt tuning and few-shot learning.

Limitations

Our results primarily reflect improvements that occur in a low training data regime, from tens of examples to a couple thousand. However, practitioners may deal with larger datasets from time to time, in which our method may be out of scope.

Furthermore, there could be other factors that help explain the disparity in performance between different teacher models. Also, we have analyzed the generated data for differences in quality, but there may be other ways to close the gap between open-source models and the GPT models as a teacher model. This warrants further investigation.

Our focus primarily reflects a specific use case where there is low training data available due to difficulty in data collection such as labor or resource constraints. Exploring the effects of using synthetic data to further eke out performance when there is abundant data is a promising research direction.

Ethics Statement

LLM2LLM relies on using LLMs to augment a training dataset in order to train another student LLM more efficiently. This can reduce the energy and monetary cost of experimentation and machine learning research, as it enables those with smaller datasets to achieve better performance on a domain-specific task. Of course, misuse of this method may lead to unethical data being generated, which can lead to societal harm. This is not a concern specific to this work, but to LLM research in general. Furthermore, there are still open questions about latent implicit biases and ethical issues surrounding the generated output of LLMs that the authors and practitioners of this method are aware of and continue to consider throughout the whole process.

Acknowledgements

We appreciate the valuable feedback from Andrew Aikawa. We acknowledge gracious support from Furiosa team. We also appreciate the support from Microsoft through their Accelerating Foundation Model Research, including great support from Sean Kuno. Furthermore, we appreciate support from Google Cloud, the Google TRC team, and specifically Jonathan Caton, and Prof. David Patterson. Prof. Keutzer's lab is sponsored by the Intel corporation, Intel One-API, Intel VLAB team, the Intel One-API center of excellence, Apple, Samsung, Panasonic, as well as funding through BDD and BAIR. We appreciate great feedback and support from Ellick Chan, Saurabh Tangri, Andres Rodriguez, and Kittur Ganesh. Sehoon Kim would like to acknowledge the support from the Korea Foundation for Advanced Studies (KFAS). Amir Gholami was supported through funding from Samsung SAIT. Michael W. Mahoney would also like to acknowledge a J. P. Morgan Chase Faculty Research Award as well as the DOE, NSF, and ONR. Our conclusions do not necessarily reflect the position or the policy of our sponsors, and no official endorsement should be inferred.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua

Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. 2021. Ext5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: When data mining meets large language model finetuning.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2023. Alpagasus: Training a better alpaca with fewer data.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan 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, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Claude Coulombe. 2018. Text data augmentation made simple by leveraging nlp cloud apis. *arXiv preprint arXiv:1812.04718*.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023a. Rephrase and respond: Let large language models ask better questions for themselves.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023b. Gpt-4 turbo v.s. gpt-4 comparison. https://github.com/da03/implicit_chain_of_thought/tree/main/gpt4_baselines.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Boyang Li, Shafiq Joty, and Lidong Bing. 2023. Is gpt-3 a good data annotator? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11173–11195.

Jon Durbin. 2023. Jondurbin/airoboros-l2-70b-3.1.2 · hugging face.

Luyang Fang, Gyeong-Geon Lee, and Xiaoming Zhai. 2023. Using gpt-4 to augment unbalanced data for automatic scoring.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023a. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023b. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

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

Patrick Haluptzok, Matthew Bowers, and Adam Tauman Kalai. 2023. Language models can teach themselves to program better.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy Chen, Zhengyuan Liu, and Diyi Yang. 2023a. Coannotating: Uncertainty-guided work allocation between human and large language models for data annotation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1487–1505.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023b. Self-alignment with instruction back-translation. *arXiv preprint arXiv:2308.06259*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Zhenwen Liang, Wenhao Yu, Tanmay Rajpurohit, Peter Clark, Xiangliang Zhang, and Ashwin Kalyan. 2023. Let gpt be a math tutor: Teaching math word problem solvers with customized exercise generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14384–14396.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023a. Tinygsm achieving 80% on gsm8k with small language models.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, et al. 2023. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir R. Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023. Crosslingual generalization through multitask finetuning. In *Annual Meeting of the Association for Computational Linguistics*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolo Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, et al. 2023. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. instructgpt.

Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2023. Rephrase, augment, reason: Visual grounding of questions for vision-language models.

Arthur L Samuel. 2000. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 44(1.2):206–226.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, et al. 2021. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Gerald Tesauro et al. 1995. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

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

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

Solomon Ubani, Suleyman Olcay Polat, and Rodney Nielsen. 2023. Zeroshotdataaug: Generating and augmenting training data with chatgpt. *arXiv preprint arXiv:2304.14334*.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? gpt-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 2022a. Supernaturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109.

Yufei Wang, Jiayi Zheng, Can Xu, Xiubo Geng, Tao Shen, Chongyang Tao, and Daxin Jiang. 2022b. Knowda: All-in-one knowledge mixture model for data augmentation in few-shot nlp. *arXiv preprint arXiv:2206.10265*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Lai Wei, Zihao Jiang, Weiran Huang, and Lichao Sun. 2023. Instructiongpt-4: A 200-instruction paradigm for fine-tuning minigpt-4.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Wang Yanggang, Haiyu Li, and Zhilin Yang. 2022. Zeroprompt: Scaling prompt-based pretraining to 1,000 tasks improves zero-shot generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4235–4252.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. 2021. Gpt3mix: Leveraging large-scale language models for text augmentation. *arXiv preprint arXiv:2104.08826*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. 2023. Self-taught optimizer (stop): Recursively self-improving code generation.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. When does pretraining help? assessing self-supervised learning for law and the casehold dataset. In *Proceedings of the 18th International Conference on Artificial Intelligence and Law*. Association for Computing Machinery.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models.

A Experimental Setup

A.1 Datasets

We evaluate LLM2LLM on five different datasets that are either multiple-choice or classification tasks and were widely adopted in prior works including (Ubani et al., 2023). Our datasets are as follows:

1. **GSM8K**: a grade school math work multiple-choice dataset that consists of 7.5K train problems and 1K test problems (Cobbe et al., 2021). Figure B.1 shows an example.
2. **CaseHOLD**: a multiple-choice law dataset that requires one to choose the relevant holding (i.e. the court's determination) of a cited case which backs up the proceeding argument (Zheng et al., 2021). Figure B.2 shows an example.
3. **SNIPS**: a 7-way classification dataset to determine the correct user intent for a voice assistant. (Coucke et al., 2018). Figure B.3 shows an example.
4. **TREC**: a 6-way classification dataset where one must classify the type of text into a category e.g., abbreviation, location, or numeric value (Li and Roth, 2002). Figure B.4 shows an example.
5. **SST-2**: a binary classification dataset to decide whether a sentence of positive or negative sentiment (Socher et al., 2013). Figure B.5 shows an example.

For each dataset, we sample between 0.02% to 50% of the total training data and use this as the seed data for each experiment. This allows us to measure how effectively LLM2LLM scales up small task-specific datasets. For consistency, we use identical samples of seed data across different experiments (e.g. 1% on GSM8K) to avoid introducing new randomness with different samples.

In particular, for SNIPS, TREC, and SST-2, we always uniformly sample the same number of examples per class, similar to (Dai et al., 2023; Ubani et al., 2023). In Table 1, we sample 10, 15, and 20 examples per class to measure the efficacy of LLM2LLM in the extreme low-data regime. These three tasks are relatively simpler than GSM8K and CaseHOLD, and therefore using an extremely small amount of training data is sufficient to achieve exemplary performance.

A.2 Models

For all of our experiments, we use Llama-2-7B (Touvron et al., 2023b) as the student model. We perform minimal prompt tuning for each task, only formatting the data as necessary and not employing many few-shot examples, which can be seen in Figure B.1 and Figure B.2. Using excessive prompting would undermine the benefits of fine-tuning and would muddle the evaluation of the effectiveness of LLM2LLM. Our fine-tuning settings are described in Section A.4.

For our main experiments, we use GPT-3.5 (1106 release) as the teacher model for data generation. In Section 4.4 and Table A.1, we show that our framework can be extended to different teacher models such as the more powerful GPT-4-Turbo (1106 release) model as well as open source LLMs such as Llama-2-70B-chat (Touvron et al., 2023b) and Airoboros-l2-70b-3.1.2 (Durbin, 2023).

A.3 Baselines and Evaluation

To measure the efficacy of LLM2LLM, we fine-tune the student model using samples of different sizes from each dataset and evaluate on the validation sets of each of these datasets. We then run 10 steps of LLM2LLM, and use the validation sets to select the best performing model. In Section 4.2, we compare these results against basic fine-tuning on just the seed data.

For GSM8K and TREC, since there is no development set, we choose the best checkpoint's test set results to be representative for the overall improvement. Similarly, for SST-2, since the test set labels are not public, we use the development set results. For all other datasets, we record the test set performance of the checkpoint that performs best on the development set.

For TREC, SST-2, and SNIPS, since these are simple classification tasks, we perform string matching between the generated output and the ground truth after some cleanup. For CaseHOLD, which is a multiple choice task, we extract the letter of the answer that the model generates. For GSM8K, we use a regular expression extraction based on the answer format that GSM8K provides. Specifically, we extract the number after the #### token.

In Section 4.3, we only sample 100 examples from GSM8K and CaseHOLD. For SNIPS, TREC, and SST-2, we sample 10 examples per class. We run an extensive ablation study against several different baselines:

- **Fine-tuning**: Standard fine-tuning on the initial seed data.
- **EDA**: EDA (Wei and Zou, 2019) which uses synonym replacement plus random swap, insert, and deletion to augment text with 10% probability. Note that we take care not to augment any special formatting or structural elements for each dataset.
- **AugGPT**: Augment our seed data using the teacher prompts from Section 3.1 with no filtering, similar to (Dai et al., 2023; Ubani et al., 2023; Yoo et al., 2021).
- **More Data**: Randomly sampling new data from unseen train data and adding to the training data.

A.4 Fine-tuning Settings

Following the example from Alpaca (Taori et al., 2023), all models are fine-tuned with a maximum sequence length of 1024 for 3 epochs with a batch size of 128 examples. We use a learning rate of 2×10^{-5} with 0 weight decay and a warmup ratio of 0.03 with a cosine learning rate scheduler. These models were trained using either 4 NVIDIA A100-80GB or 8 NVIDIA A6000s. We do full fine-tuning for simplicity and to reduce the complexity of our experiments, but in practice, one can also use some form of parameter-efficient fine-tuning method such as LoRA (Hu et al., 2021).

B LLM2LLM Details

B.1 Fine-tuning

The fine-tuning step trains a small student model using seed data and previously generated LLM2LLM data, if any. This LLM2LLM data is generated by a process further detailed in Section 3.1 that generates synthetic data targeted toward data points the model got wrong. We always fine-tune the original student model on the full dataset (seed data + LLM2LLM data) at each step. Our ablation study in Section 4.5.3 shows that fine-tuning the original baseline model from scratch on the full dataset always outperformed re-using the already-fine-tuned model. We hypothesize that this is because fine-tuned models have already seen most of the data, causing them to overfit and fail to converge to a better optimum.

B.2 Evaluation

After fine-tuning, the model needs to be evaluated on the original (training) seed data to identify the examples that the model gets wrong. This allows the LLM2LLM framework to use those failed examples to generate targeted synthetic data for the model to train on. For example, if the model was unable to solve problems involving the Pythagorean theorem as in Figure 1, these examples will be used to generate more problems with this concept.

For many datasets, this evaluation step can be extremely costly, as traditional NLP datasets can have more than thousands of data points. However, this evaluation step is cost-effective and relatively quick in the low-data regime where the seed dataset size is small, thereby not slowing down the LLM2LLM process.

B.3 Filtering Generated Dataset

Once the teacher model generates the synthetic data, we need to apply simple filtering of the output for quality insurance. Like in previous work (Wang et al., 2023; Taori et al., 2023), we use regex filters to ensure that the basic format of the output is aligned with our expectations. We also use a ROUGE (Lin, 2004) filter in order to enforce that the augmented data points are sufficiently different from previous samples. However, we use a weaker ROUGE filter of 0.95 to filter out similar instructions, rather than the score of 0.85 used in other works like Alpaca (Taori et al., 2023) and Self-instruct (Wang et al., 2023). The reason we can do this is because unlike Alpaca and Self-Instruct, we do not require as much diversity in the generations, as we are not targeting general-purpose instruction-following. In fact, we would like to constrain the generated data points to the task and domain of the datasets as much as we can. Thus, we are able to use a weaker filter so that we can simply filter out exact matches during generation. In addition, since we are augmenting each sample individually, there is already enough inherent diversity in the generation process.

B.4 Prompting Details

We devised simple but thorough prompts for each task that the teacher model uses while augmenting the dataset. Previous work in open-domain dataset generation such as (Wang et al., 2023; Mukherjee et al., 2023; Xu et al., 2023) used generic system

Dataset	# Seed	Teacher	Total # Aug	Accuracy (%)
GSM8K	74 (1%)	Llama-2-70B	333	11.83
		Airoboros	345	15.01
		GPT-3.5	391	19.56
		GPT-4-Turbo	388	**19.79**
	149 (2%)	Llama-2-70B	661	17.59
		Airoboros	671	19.33
		GPT-3.5	802	25.70
		GPT-4-Turbo	805	**25.78**
	343 (5%)	Llama-2-70B	1308	19.33
		Airoboros	1286	21.76
		GPT-3.5	1641	27.07
		GPT-4-Turbo	1739	**28.43**

Table A.1: Experiments on how the quality of teacher model affects the performance of LLM2LLM. For each of these experiments, we only change the teacher model to measure the effect of the teacher model on the final outcome.

instructions for generating new data points from stronger models such as GPT-3.5 and GPT-4. This was necessary as these approaches targeted improving the LLM over a wide range of different tasks. However, for LLM2LLM, we are trying to improve the LLM at domain-specific tasks. Thus, for each task, the system prompt that we give to the teacher-model differs on a per-task basis. This allows the user to inject and leverage domain-specific knowledge about the task into the dataset generation procedure, creating higher quality fine-tuning data. In practice, we also use in-context learning with few-shot prompting to bootstrap the teacher model's ability to generate relevant questions.

The detailed system prompt and in-context examples for each dataset are provided below:

1. GSM8K: System (Figure B.6) and In-Context Examples (Figure B.7)
2. CaseHOLD: System (Figure B.8) and In-Context Examples (Figure B.9)
3. SNIPS: System (Figure B.10) and In-Context Examples (Figure B.11)
4. TREC: System (Figure B.12) and In-Context Examples (Figure B.13)
5. SST-2: System (Figure B.14) and In-Context Examples (Figure B.15)

Dataset	% Data	Cost ($)	Time (Hours)
GSM8K	1%	0.35	3.28
	5%	1.48	9.07
	10%	3.64	14.54
CaseHOLD	1%	1.50	6.68
	5%	0.84	16.87
	10%	2.19	31.95
SNIPS	0.5%	0.02	0.85
	0.8%	0.05	1.29
	1%	0.05	1.40
TREC	1.1%	0.05	0.67
	1.6%	0.01	0.44
	2.2%	0.02	0.61
SST-2	0.02%	0.01	0.54
	0.04%	0.01	0.80
	0.06%	0.00	0.64

Table B.2: Training and data generation Costs of LLM2LLM. The first and second columns indicate the dataset and percentage of the training data used as initial seed data for that experiment. The third column indicates the total cost to generate the data from the GPT-3.5 teacher model. The fourth column shows the total time in hours to train and evaluate the student model. As we can see, data generation costs for LLM2LLM are relatively little compared to the cost of manually curating new data. Furthermore, fine-tuning and evaluation of the student model finishes in a reasonable time.

B.5 Training and Data Generation Costs

In Table B.2, we report the training and data generation costs to perform LLM2LLM. This includes the cost of generating new data from OpenAI as well as the amount of GPU hours required to train and evaluate the student models. We measured these numbers using 4xA100-80GB PCIe based NVIDIA GPUs. As we can see, generating data for LLM2LLM costs relatively little compared to the cost of collecting new data points manually. Furthermore, the process of fine-tuning the student model also finishes in a reasonable amount of time.

B.6 Decontamination Experiments

When using an LLM to generate data, there are potential concerns of data contamination i.e., when

Dataset	Avg.	>66% (%)	Max % (Count)
GSM8K Train	52.08	11.46	81.48 (1)
GSM8K Test	26.38	0	46.67 (1)
CaseHOLD Train	31.01	0	50.42 (2)
CaseHOLD Test	25.66	0	38.46 (1)
SNIPS Train	59.94	34.21	85.71 (1)
SNIPS Test	45.90	7.89	80.00 (1)
TREC Train	47.62	17.04	80.00 (4)
TREC Test	33.47	6.67	80.00 (1)
SST-2 Train	34.35	0	57.14 (2)
SST-2 Dev	25.46	0	42.86 (1)

Table B.3: Word overlap by dataset of synthetic examples generated after 10 steps of LLM2LLM using 100 seed examples as in Section 4.3. Column *Dataset* indicates the dataset and train split being used to compare the synthetic data with. Column *Avg.* is the average overlap percentage. Column *>66%* is the percentage of examples with above 66% overlap, and column *Max % (Count)* indicates the maximum overlap percentage and the number of examples at that overlap percentage.

the teacher LLM has been trained on the test data and thus leaks it into the student model's training data which would artificially inflate the student model's scores. To test for this we measure the word overlap percentage between examples generated by the teacher model and examples from the train and test set for each dataset in the same manner as (Ubani et al., 2023).

Word overlap is computed by first removing stop words and punctuation from each example. Then, for each example pair, we count the number of words that are common between the two, and divide by the number of words in longer example. This provides a metric for measuring how many words are similar between the two examples.

For each dataset, we run LLM2LLM for 10 steps using the same amount of seed data as in Section 4.3. Then, we take all the synthetic data generated for the final step and calculate our word overlap metric per example. We then calculate word overlap summary statistics similarly to (Ubani et al., 2023) which are the percentage of examples above 66% similarity, the maximum overlap, and the number of examples at the maximum overlap.

Looking at Table B.3, we can see varying levels of overlap between the synthetic data and the training data. This is to be expected, as LLM2LLM uses training data as seed data in order to generate new synthetic examples based on that training data.

Regarding the test/dev set results, we see that for GSM8K, CaseHOLD, and SST-2, 0% of the generated examples have above 66% word overlap with the test set. This indicates that the model is not leaking test set data into the student model's training data. For SNIPS and TREC, we see that the percent of examples with above 66% overlap is still well below 10% thus indicating no large-scale overlap.

For SNIPS and TREC specifically the maximum overlap percentage is 80%. Upon closer inspection this is because these two datasets have very short examples. Given the small number of such examples we don't believe this poses a high risk of data contamination. (Ubani et al., 2023) came to a similar conclusion regarding these two datasets.

GSM8K Example:

Victor, Austin, and Brian made traps to catch shrimp. Victor's trap caught 26 shrimp and Austin's trap caught 8 less than Victor's. Brian's trap caught half of Victor and Austin's total number of shrimp. If the boys then sold their shrimp for $7 for every 11 tails of shrimp and then divided their earnings equally amongst themselves, how much money does each boy make?

Austin's trap caught 26 - 8 = «26-8=18»18 shrimp.
Together, Victor and Austin's traps caught 18 + 26 = «18+26=44»44 shrimp.
Brian's trap caught 44/2 = «44/2=22»22 shrimp
In total, they caught 26 + 18 + 22 = «26+18+22=66»66 shrimp.
They were able to sell 66/11 = «66/11=6»6 sets of shrimp.
They made a total of 6 x 7 =«6*7=42»42
Each boy made $42/3$ =«42/3=14»14
14

Figure B.1: Formatted example from GSM8K.

CaseHOLD Example:

The following context is from a judicial decision where the holding statement has been masked out as <HOLDING>.

Context: from behind the bench in the robe is protected by the First Amendment, even if his use of the trappings of judicial office were notprotected by First Amendment); Halleck v. Berlinger, 427 F. Supp. 1225, 1241 (D.D.C. 1977) (applying the First Amendment in disciplinary proceeding to comments made from the bench, but finding the particular comments outside of its protection); Mississippi Comm'n on Judicial Performance v. Boland, 975 So.2d 882, 891-92 (Miss. 2008) (applying First Amendment to ajudge acting in her "capacity as a justice court judge" at a conference seeking certification to start a drug court, but held that First Amendment did not apply because judge's insulting comments were not matters of "legitimate public concern."); In re Rome, 218 Kan. 198, 542 P.2d 676, 684 (1975) (<HOLDING>). 11 As indicated in the Gentile syllabus,

Please select the correct holding statement from the options below.

A. holding that free speech protection of new jersey constitution requires subject to reasonable restrictions privatelyowned shopping centers to permit speech on political and societal issues on premises unlike first amendment of federal constitution
B. recognizing that code is speech
C. holding that first amendment protections apply to compelled speech as well as restrictions on speech
D. holding that although ajudge has the right of free speech any restrictions placed by the code of professional responsibility are acceptable limits and prevent the first amendment from exempting a judge from discipline for proven judicial misconduct
E. holding that the first amendment limits judicial discretion to seal documents in a civil case

D

Figure B.2: Formatted example from CaseHOLD.

SNIPS Example:

The following is a transcript of something someone said.
Classify the intent of the speaker into the following categories:
- AddToPlaylist
- BookRestaurant
- GetWeather
- PlayMusic
- RateBook
- SearchCreativeWork
- SearchScreeningEvent

Transcript: go to bioruby

SearchCreativeWork

Figure B.3: Formatted example from SNIPS.

TREC Example:

The following is a question.
Classify the question into the following categories:
- ABBR
- ENTY
- DESC
- HUM
- LOC
- NUM

Question: What is the full form of .com ?

ABBR

Figure B.4: Example from TREC.

SST-2 Example:

Classify the following movie review as positive or negative: dazzling and sugar-sweet , a blast of shallow magnificence that only sex , scandal , and a chorus line of dangerous damsels can deliver .

negative

Figure B.5: Example from SST-2.

System:

You are a educational A.I. whose purpose is to take math problems that students get wrong and generate new problems to help them practice their mathematical skills. Your goal is to generate a set of new math problems that reflect the different skills and techniques found in the example problem.

Here are the requirements:
1. A GPT language model should be able to complete the problem. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
2. The math problem should be in English.
3. The output should be an appropriate response to the question. Make sure the output is less than 100 words.
4. The answer to the problem should be expressed as a number, not a fraction. For example, if the answer is one-half, return 0.5, not 1/2 or "one half".
5. The answer to the problem should not have units i.e. if the answer is 6 cups, just write 6 as the [ANSWER]
6. Always include some calculation to show your work for how you got your ANSWER.
7. Don't make any mathematical mistakes of your own!
8. Try not to copy too much information from the original problem. If you must, try and replace names and numbers so that we can test the student's understanding, rather than their ability to memorize previous test questions.

Always return your instructions in the form:
1. Question: [QUESTION]
Answer: [CALCULATION]
#### [ANSWER]

Figure B.6: System Prompt for GSM8K Generation

User:

The student was given the following question:

Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?

The answer key has this as the rationale and answer:

In the beginning, Betty has only 100 / 2 = $«100/2=50»50.
Betty’s grandparents gave her 15 * 2 = $«15*2=30»30.
This means, Betty needs 100 - 50 - 30 - 15 = $«100-50-30-15=5»5 more.
5

Please generate 1 similar question, along with the correct calculations and rationale.

Assistant:

Sure, I can help you with that! Here’s a new math question based on the same scenario, along with the correct calculations and rationale:

1. Question: Betty is saving money for a new wallet which costs $150. Betty has already saved $30. Her parents decided to give her $20 for that purpose, and her grandparents gave her three times as much as her parents. How much more money does Betty need to buy the wallet?
Answer: In the beginning, Betty has saved $30. Betty’s parents gave her $20. Betty’s grandparents gave her 3 * $20 = $60. Total amount Betty has: $30 + $20 + $60 = $110. The cost of the wallet is $150. Amount more needed: $150 - $110 = $40.
40

Figure B.7: In-Context Example for GSM8K Generation

System:

You are LawGPT, an AI agent who knows everything there is to know about U.S. law.
You know the result of every court case and you know every law in the lawbook.
The user is trying to choose the correct holding of the case given the context and argument of the court.
You are trying to give the user assistance by giving them more practice questions for the questions that they get wrong.

Here are the requirements:
1. A GPT language model should be able to complete the problem. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
2. The context, holding, and options should be in english.
3. The questions that you generate should test for whether the user understands the case names and their holdings and whether the user can re-frame relevant holdings to backup the argument in the context.
4. The context should always end with a citation such as "See United States v. Newman, 125 F.3d 863 (10th Cir.1997) (unpublished) (<HOLDING>); United States v. Dodge, 846 F.Supp. 181,"
5. The citation absolutely needs to have the mask phrase <HOLDING> which is the place where the legal holding would normally be.
6. The questions should always be multiple choice.
7. There should always be 5 options: 1 options should be a holding that backs up the argument in the context, the other 4 should be sufficiently different. Each option has to start with the word "holding"
8. There can only be 1 answer: A, B, C, D, or E.
9. Don't make any mistakes matching the holdings yourself.
10. Try not to copy too much information from the original problem. You don't want the user to just memorize their answer.
11. Make the context similar to the context in question, make sure that the holding that is being tested is the same.
12. The wrong answer choices can be any other reasonable holding, but it should be sufficiently different from the correct answer.
13. Do not make your context too short. Remember, these arguments in the context are being made by judges and should look like they were written by a judge.

Always return your instructions in the form:
1. Context: [CONTEXT]

Please select the correct holding statement from the options below.

A. [OPTION 1]
B. [OPTION 2]
C. [OPTION 3]
D. [OPTION 4]
E. [OPTION 5]
Answer: [ANSWER]

Figure B.8: System Prompt for CaseHOLD Generation

User:

The following is a multiple choice question about the holding statements of a judicial decision that the user got wrong including the correct answer from the answer sheet:

Context: Drapeau’s cohorts, the cohort would be a “victim” of making the bomb. Further, firebombs are inherently dangerous. There is no peaceful purpose for making a bomb. Felony offenses that involve explosives qualify as “violent crimes” for purposes of enhancing the sentences of career offenders. See 18 U.S.C. § 924(e)(2)(B)(ii) (defining a “violent felony” as: “any crime punishable by imprisonment for a term exceeding one year ... that ... involves use of explosives”). Courts have found possession of a’bomb to be a crime of violence based on the lack of a nonviolent purpose for a bomb and the fact that, by its very nature, there is a substantial risk that the bomb would be used against the person or property of another. See United States v. Newman, 125 F.3d 863 (10th Cir.1997) (unpublished) (<HOLDING>); United States v. Dodge, 846 F.Supp. 181,

Please select the correct holding statement from the options below.

A. holding that possession of a pipe bomb is a crime of violence for purposes of 18 usc 3142f1
B. holding that bank robbery by force and violence or intimidation under 18 usc 2113a is a crime of violence
C. holding that sexual assault of a child qualified as crime of violence under 18 usc 16
D. holding for the purposes of 18 usc 924e that being a felon in possession of a firearm is not a violent felony as defined in 18 usc 924e2b
E. holding that a court must only look to the statutory definition not the underlying circumstances of the crime to determine whether a given offense is by its nature a crime of violence for purposes of 18 usc 16
Answer: A

Please generate 1 similar question, along with 5 different holding options and the correct answer.

Assistant:

Sure, I can help you with that! Here’s a new case holding question based on that question:

1. Context: In the case of Martinez v. State, it was determined that the act of threatening another with a weapon, even if not used, constitutes a violent act. This is based on the premise that the mere possession of a weapon with the intent to threaten establishes an inherent risk of physical harm. Such acts align with the definition of "violent crimes" which can be used as grounds for stricter penalties under the jurisdiction of certain statutes. Refer to 18 U.S.C. § 16(b) which defines a “crime of violence” as: “any other offense that is a felony and that, by its nature, involves a substantial risk that physical force against the person or property of another may be used in the course of committing the offense.” In the context of this, the Jones v. Commonwealth, 279 F.3d 722 (4th Cir.2001) case had a significant <HOLDING>.

Please select the correct holding statement from the options below.

A. holding that mere possession of a weapon without any intent does not constitute a crime of violence under 18 U.S.C. § 16(b)
B. holding that using a computer to commit fraud is considered a crime of violence because of the potential harm to victims’ financial well-being
C. holding that threatening another with a weapon, even if not used, is a crime of violence for purposes of 18 U.S.C. § 16(b)
D. holding that in terms of 18 U.S.C. § 16(b), verbal threats without the presence of a weapon do not qualify as a crime of violence
E. holding that any crime which results in a financial penalty, irrespective of physical harm, is a crime of violence under 18 U.S.C. § 16(b)
Answer: C

Figure B.9: In-Context Example for CaseHOLD Generation

System:
You are TranscriptGPT, an AI agent who knows the intent of the transcript of different questions.
You are training someone how to identify people's intents from what they have said.
You are trying to give the user assistance by giving them more practice questions for the questions that they get wrong.

Here are the requirements:
1. A GPT language model should be able to complete the problem. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
2. The question and options should be in english.
3. The questions that you generate should have only 1 of the following intents:
- AddToPlaylist
- BookRestaurant
- GetWeather
- PlayMusic
- RateBook
- SearchCreativeWork
- SearchScreeningEvent
4. The questions should always have 1 specific intent.
5. The intent of the question must come from the list above.
6. Don't make any mistakes with your answer yourself.
7. Try not to copy too much information from the original problem. You don't want the user to just memorize the practice problems.
8. Make the intent the same as the question that the user got wrong.
9. The wrong answer choices can be any other reasonable answer, but it should be sufficiently different from the correct answer.
10. The transcript should be something that an ASR model could output: it must sound like something a human could say.

Always return your instructions in the form:
1. Transcript: [CONTEXT]
Intent: [INTENT]

Figure B.10: System Prompt for SNIPS Generation

User:

The following is a transcript whose intent the user got wrong including the correct answer from the answer key:

The following is a transcript of something someone said.
Classify the intent of the speaker into the following categories:

- AddToPlaylist
- BookRestaurant
- GetWeather
- PlayMusic
- RateBook
- SearchCreativeWork
- SearchScreeningEvent

Transcript: what s the weather here at 2 pm ',

Intent: GetWeather

Give me another 1 similar transcripts with the same answer (GetWeather).

Assistant:

Here's a similar transcript with the same intended answer category:

1. Transcript: Will it rain in the city this afternoon?
Intent: GetWeather

Figure B.11: In-Context Example for SNIPS Generation

System:

You are QuestionGPT, an AI agent who knows the class of different question.
You are training someone how to classify different questions based on what the questions are asking form.
You are trying to give the user assistance by giving them more practice questions for the questions that they get wrong.

Here are the requirements:
1. A GPT language model should be able to complete the problem. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
2. The question should be in english.
3. The questions that you generate should have only 1 of the following intents:
- ABBR (Abbreviation)
- ENTY (Entity)
- DESC (Description/Concept)
- HUM (Human)
- LOC (Location)
- NUM (Number)

4. The questions should always have 1 specific class.
5. The intent of the question must come from the list above.
6. Don't make any mistakes with your answer yourself.
7. Try not to copy too much information from the original problem. You don't want the user to just memorize the practice problems.
8. Make the class the same as the question that the user got wrong.
9. The question should be something that an ASR model could output: it must sound like something a human could say.

Always return your instructions in the form:
1. Question: [CONTEXT]
Class: [INTENT]

Figure B.12: System Prompt for TREC Generation

User:

The following is a question that the user was unable to classify correctly:

Classify the question into the following categories:
- ABBR
- ENTY
- DESC
- HUM
- LOC
- NUM

Question: What country do the Galapagos Islands belong to ?
Class: LOC

Give me another 1 similar question with the same class (LOC).

Assistant:

Here's a similar question with the same class:

1. Question: What country is the Great Barrier Reef located in?
Class: LOC

Figure B.13: In-Context Example for TREC Generation

You are MovieReviewerGPT, an AI agent who knows how people review movies.
You are training someone who is classifying movies as positive or negative.
You are trying to give the user assistance by giving them more practice reviews for the reviews that they get wrong.

Here are the requirements:
1. A GPT language model should be able to complete the problem. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
2. The review should be in english.
3. The reviews that you generate should be either positive or negative.
4. The reviews should always have 1 specific class.
5. Don't make any mistakes with your answer yourself.
6. Try not to copy too much information from the original problem. You don't want the user to just memorize the practice problems.
7. Make the class the same as the review that the user got wrong.
8. The review should be something that a user on the internet could type into a movie review forum.

Always return your instructions in the form:
1. Review: [CONTEXT]
Sentiment: [SENTIMENT]

Figure B.14: System Prompt for SST-2 Generation

User:

The following is a movie review that the user classified incorrectly including the correct classification:
Classify the following movie review as positive or negative: as they come , already having been recycled more times than i ’d care to count
Sentiment: negative

Generate 1 more similar movie review with the same class.

Assistant:

Here’s a similar question with the same class:

1. Review: Feels like a reheated plot, utterly predictable and uninspired.
Sentiment: negative

Figure B.15: In-Context Example for SST-2 Generation

Seed Data Example: Katrina saw an ad that said if she signed up for her local recycling program, she could earn $5.00. When she signed up, they told her for every friend that she referred, the friend would receive $5.00 and she would receive another $5.00 per friend. That day, she had 5 friends sign up and another 7 friends by the end of the week. How much money in total did she and her friends make?

GPT-4-Turbo: **Michael joined a library program that rewards $4.00 for becoming a new member. Additionally, if he recommends the program to his friends, each friend gets $4.00 and Michael also gets $4.00 for each referee. Michael got 3 friends to sign up on the first day and another 6 friends joined the program over the next two weeks.** How much money in total did Michael and his friends earn?

GPT-3.5: Sophia saw an ad stating that if she signed up for her local fitness center, she could receive $10 as a reward. **For every friend she refers, the friend receives $10, and Sophia receives an additional $10 per friend. On the first day, she referred 3 friends, and by the end of the week, she referred 6 more friends.** How much money did Sophia and her friends make in total?

Airoboros: Sophie saw an ad that said if she signed up for her local recycling program, she could earn $3.00. When she signed up, they told her for every friend that she referred, the friend would receive $3.00 and she would receive another $3.00 per friend. That day, she had 7 friends sign up and another 9 friends by the end of the week. How much money in total did she and her friends make?

Llama-2-70B: **Joe signed up for his local recycling program and was promised $10 for every friend he referred.** When he signed up, they told him for every friend that he referred, the friend would receive $10 and he would receive another $10 per friend. That day, he had 3 friends sign up and another 8 friends by the end of the week. How much money in total did he and his friends make?

Figure B.16: Example of new training data generated by various models for the same seed data example on GSM8K. The bold portions in each cell qualitatively highlight sentences that have been rewritten beyond simple noun or verb replacement. GPT models rewrite the seed data to a greater degree, while Airoboros and Llama-2-70B generally make fewer alterations to the sentence structure.