

Final Report for CSE 517 Winter 2024 Final Project

David Dai
447
kun02@uw.edu

Edward Qin
517
edwardcq@uw.edu

Edward Zhang
447
ezhang8@uw.edu

Project Information:

Project Type	Default 517 Project
Original Paper Title	Solving Math Word Problems via Cooperative Reasoning induced Language Models
Venue	Association for Computational Linguistics
Year	2023
URL	https://aclanthology.org/2023.acl-long.245.pdf

- **David Dai:** debugging, running experiments for training/comparing verifiers
- **Edward Zhang:** debugging, running experiments for training/comparing generators
- **Edward Qin:** debugging, running experiments for generator-only vs generator-verifier experiments

1 Introduction

As Large Language Models become more powerful, we see an increasing interest in evaluating these models' capabilities on more involved logical problems. Since, math word problems involve a combination of both rigorous reasoning as well as common sense, we believe evaluating a language model on such problems yields invaluable insight into a model's logic capabilities.

There exists a wide variety of models focused on solving math word problems, but most models end up memorizing problem types with a large number of parameters Wei et al. (2023). In this report, we perform an ablation study to evaluate a human-thinking-inspired Cooperative Reasoning (CoRe) technique presented in Zhu et al. (2022), which utilizes a generator-verifier model pair to yield significantly higher accuracy over state-of-the-art large language models while using less than one-tenth of the parameters.

We evaluate CoRe on its effectiveness under various generator and verifier models as well as the impact of the verifier on the base generator and its accuracy. For varying both the generator and verifier models, we use tiny-GPT2 from sshleifer (2020), distilgpt2 from Sanh et al. (2019), and GPT-2 from Radford et al. (2019). To evaluate the effectiveness of the verifier, we compare each generator-verifier combination against each generator on its own. Our empirical results indicate a significant correlation between improvement in generated solutions and increase in model size. Qualitatively, we also observe that the verifier can critically teach the generator to produce more logical reasoning paths.

Our key contributions include:

- Replication and debugging of the CoRe paper
- Modification of the CoRe architecture to run on resource-constrained environments
- Performance evaluation of the effectiveness of various base generator and verifiers
- Reinforce the claim that verifiers can improve the reasoning output of generator models

2 Background

Applying language models to the task of solving math word problems is an interesting task. Word problems are especially interesting, both to students that seem to struggle with them more than typical calculation-based problems, and researchers that observe this phenomenon. For example, a 2014 paper by Fuchs et al. (2014) discusses how the nature of word problems adds a great deal of complexity to the problem. Not only do students need to understand mathematical symbols, they also learn to translate natural language to algebraic formulae and understand representations of problems. In fact, natural language processing, neuroscience, and cognitive science researchers have spent over a decade tackling the problem of math word problems.

3 Method

Our current method is based on the Cooperative Reasoning framework (CoRe), as presented in our root paper. To build on top of CoRe, we first explain its structure. CoRe is inspired by the observation that human reasoning consists of two parts: instinct, and deliberate thinking, according to Evans (2003). CoRe's architecture consists of two parts.

First, a pretrained language model (PLM) is fined-tuned on the training set \mathcal{D} of questions, reasoning paths (chains of thought), and ground truth answers to generate an augmented dataset \mathcal{D}^+ of reasoning paths and answers. The PLM is referred to as the generator model, with objective function given by

$$L_{LM} = - \sum_{i=1}^{|p|+|gt|} \log P(x_i | x_{<i}) \quad (1)$$

Then, two types of verifier models are trained on the generated \mathcal{D}^+ to evaluate the generator’s reasoning capabilities. The verifiers score the reasoning path as described below.

The token-level scorer V_s mimics a fine-grained analyzer that scores the generator’s reasoning at a token-by-token level. The verification loss for V_s is given below, where $|rp|$ and $|a|$ are the lengths in tokens of the reasoning path and answer, respectively.

$$L_{VS} = \sum_{i=1}^{|rp|+|a|} (\text{score}_i - \mathbb{I}(a == gt))^2 \quad (2)$$

A path-level scorer V_p mimics a coarse-grained analyzer that scores the overall direction of the generator’s reasoning. The verification loss for V_p is similar to the one for V_s .

We note that forcing the generator to output a reasoning path, not just an answer is important for several reasons:

- It’s useful to prevent the model from performing hard memorization
- Previous works including Zhu et al. (2022) have discovered that fine-tuning a PLM is less effective than training and scoring with verifiers

CoRe inference works through Monte Carlo Tree Search (MCTS). MCTS represents logical reasoning steps as nodes in a tree, and selects nodes using a version of the PUCT formula. At a current state s and node n , we select node n^* with probability 50%. $N(s, n)$ = number of times we selected node n in state s (possibly acting as a regularization term), $R(n)$ = reward function based on verifier scores, and C = set of children of n^* .

$$n^* = \arg \max_{n \in C} (R(n) + c_{puct} \pi(n|s) \frac{\sum_{b \in C} N(s, b)}{1 + N(s, n)}) \quad (3)$$

As such, our method studies the effect of smaller PLMs for the base generator and step verifier models on inference quality. Additionally, we aim to perform an ablation study of cooperative reasoning and learn how verifiers affect the strength of mathematical reasoning output.

4 Experiments

Our first experiment was to reproduce the results of the original root paper by building and testing their code, which is publicly available. The goal was to fine-tune a set of pre-trained language models (the paper uses GPTJ-6B and Microsoft’s Deberta-v3-large) on a dataset of math word problems (GSM8K).

We made several changes and design decisions:

- There were numerous bugs in the original codebase. We fixed the majority of issues with dependencies and library versions.
- Even with GCE VMs, it was infeasible to work with a model like GPT-J 6B since we are limited by the amount of GPU VRAM. We have 2 solutions to this. One is to simply utilize smaller models (e.g. deberta-v3-small, a version of GPT-2 with only 137M parameters). The second step is to reduce parameters such as batch size and sequence length during training, while using memory optimizations like ZeRO.

The experiments we ran are framed around the idea of an ablation study, where we remove and swap the generator and verifier components of the Cooperative Reasoning framework and investigate the effect on its mathematical reasoning.

Our first experiment focuses on the performance of CoRe upon swapping different generator models of varying parameter size and complexity. Our hypothesis is that the pre-trained models with more relevant and fluent generations will yield more logical reasoning paths and accurate answers. Intuitively, a more intelligent immediate reaction is easier to fine-tune and correct.

Our second experiment focuses on the performance of CoRe upon swapping different step verifier models. Our hypothesis is that verifiers with higher complexity can better tune a given generator.

Our last experiment focuses on the overall impact of verifier models. We test this by omitting the verifier model, testing the performance of just a fine-tuned generator model. Based on the arguments presented by Zhu et al. (2022), we hypothesize that the addition of verifiers improves the logical flow of mathematical reasoning output.

4.1 Model

The following models were found on HuggingFace. We used these models because they are powerful enough to yield notable results, and also fit into GPU (T4) VRAM and train in a reasonable amount of time.

Decoders used for generator and step-level verifier included

- tiny-GPT2: a reduced GPT-2 variation with estimated 500K parameters
- distilgpt2 a distilled GPT-2 variation with 88M parameters
- GPT2 a smaller version of GPT-2 with 137M parameters

The only encoder we used for the path-level verifier was Microsoft’s deberta-v3-small model.

All models were trained and tested with 2 CPUs per GPUs. The decoder generator models were trained over 10 epochs, batch size 16, and maximum token length 256. The decoder verifier models were trained over 5 epochs. The encoder model was trained over 5 epochs, with 16 workers and micro batch size of 8. All other hyperparameters were kept as defaults from the `.slurm` files in the original codebase.

4.2 Datasets

We primarily use an 80/10/10 split of the GMS8k dataset Cobbe et al. (2021) to train the generator and for inference. This was also used by Zhu et al. (2022). For the verifiers, we use a gold label \mathcal{D}^+ pre-generated by the authors to maximize the impact of the verifiers on the mathematical reasoning task.

4.3 Baselines

Our experiment consists of both a quantitative evaluation of test accuracy and a qualitative evaluation of the impact of verifiers. The baseline of our quantitative evaluation will be the root paper’s presented models, and the baseline of our qualitative evaluation will be the generation results of our generator-only models.

4.4 Code

Link to our codebase: [link](#). This code essentially loads in the original paper codebase and includes fixes to dependencies/library versions that allows training and testing to occur

5 Results

Our experiment is motivated by two key questions: (1) how well does the CoRe architecture perform using significantly less resource-intensive language models, and (2) how much

does the verifier impact generator performance. To address our first question, we compare the performance of our smaller language models against that of the large models presented in the root paper. To address our second question, we set up three experiments.

Experiment 1 serves as a reference point, where we observe the effect of different size base generators on the ability to answer the math word problems presented in GSM8k. Experiment 2 measures how much different size verifiers affect each generator model. Experiment 3 compares the performance of generator-only architectures against generator-verifier architectures on our test dataset. Together, these three experiments show the function and usefulness of the verifier presented in the CoRe paper.

5.1 CoRe Under Resource Constraints

The original CoRe paper utilizes test accuracy as the key evaluation metric with GPT-3 (175B) and PaLM (540B) as baseline comparison. The result of the original CoRe model consisting of two GPT-J (6B) and the baselines evaluated on the seed dataset GSM8k is presented in the table below.

Backbone	GSM8K
Instruct GPT-3	49.6%
PaLM	56.5%
CoRe (using GPTJ-6B)	63.2 %

Figure 1: Test Accuracy of Original CoRe model and other baselines

Our replication consists of 12 different models. For each of the three generator base models as described in Section 4.1, we test on no verifier model, as well as the three step verifier models. The accuracy of each of the replication models evaluated on GSM8k is presented in Figure 2.

Generator	Step Verifier			
	None	tiny-GPT2	distilgpt2	GPT2
tiny-GPT2	0.00%	0.00%	0.00%	0.00%
distilgpt2	2.68%	1.33%	0.00%	0.67%
GPT2	1.88%	1.33%	1.33%	4.00%

Figure 2: Test Accuracy of Generator and Step Verifier combinations

Comparing our replication models against the original CoRe paper’s GPTJ-6B models and the original baseline clearly shows that our models yield significantly lower accuracy. However, this is not a surprising result given that our largest model (GPT2 137M) is around 50 times smaller than GPTJ-6B. Since our model has significantly fewer parameters, it is natural that it lacks the same capacity as the original model.

Comparing specifically among our replicated models, we notice a similar trend that larger models tend to perform better than smaller models. The tiny-GPT based models with around 100k parameters yielded 0% accuracy across all verifiers and base generators, while the distilgpt2 and GPT2 based models (both almost 100 times larger) yielded non-zero accuracies for various architecture combinations. This further supports the claim that increasing model size leads to an increase in accuracy, due to the larger model’s capability to capture more trends.

5.2 Purpose of Verifiers

From Figure 2, pure quantitative analysis on accuracy does not seem to introduce any significant trend between generator-only models and their corresponding generator-verifier

Prompt: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?	
Ground Truth: He sprints $3 \times 3 = 9$ times So he runs $9 \times 60 = 540$ meters [ANS] 540< endoftext >	
Verifier	Experiment 2: GPT2 Samples
tiny-GPT2	He runs 60 meters * 3 = 180 meters a week. He runs 180 meters * 3 = 540 meters a week. [ANS] 540< endoftext >
distilgpt2	He runs $60/3 = 20.0$ meters every week. So he runs $60 * 3 = 180$ meters a week Each of the 3 runs gives him $180/3 = 60.0$ meters [ANS] 60< endoftext >
GPT2	He runs $3 \times 60 = 180$ meters So he runs $180 \times 3 = 540$ meters a week [ANS] 540< endoftext >

Figure 4: Experiment 2 inference on GPT2 base generator model and trained step verifier

The observed trends might be explained by the fact that the step verifier’s loss function motivates V_S to push the generated response towards the ground truth logic paths. Since repeating words do not reflect the ground truth label, the verifier was able to perturb the generated output out of a repetitive state. However, the verifier’s ability to add variation into the generated output can also be affected by the verifier’s model size, as Figure 4 showed that only larger verifiers were able to break sentence structure level repetitions. Finally, the fact that the step verifier teaches the generator more coherent reasoning paths might lead the generator to arrive at a correct final answer.

5.2.3 Effect of Verifier

Experiment 3 aims to evaluate the impact of the verifier on the base generator’s performance. In this experiment, we compare the generated output for the same test prompt from both the generator-only model as well as a generator-verifier pair model. The result is shown in Figure 5.

Qualitatively, we observe the same overall trend that verifiers can reduce redundancy as we have seen in Experiment 2. This is particularly prominent in the smaller tiny-GPT2 and distilgpt2 outputs, as the generator-only models tends to repeat words or sentence structures while the generator-verifier pairs were able to produce more variations in sentences, which produces more natural speech patterns. However, the effect of verifiers on accuracy is unclear, as we do not have significant evidence that verifiers can push the generated response towards a correct solution. However, experimenting with larger-scaled models may yield more meaningful data in future experiments. Even though the answer is frequently incorrect, we still observe that verifiers are able to produce more logically coherent reasoning paths.

Overall, we conclude that verifiers introduce more natural reasoning paths compared to the generator-only models as verifiers serve as an additional incentive for the combined models to produce results closer to human reasoning paths in the ground truths.

6 Discussion

Our results present a look at solving math word problems using smaller generator and verifier models. This adds another spark to the trend of using smaller machine learning models for a broader range of tasks, and without much loss in performance. Additionally, our results indicate that the use of a verifier can have significant improvement in mathematical reasoning soundness, especially on much smaller generator models. This brings to question what other tasks can use verifier models to additionally fine-tune chain-of-thought-like tasks for large performance gains using small base models.

We think that the experimental setup and conclusions of the root paper are sound. However, we feel that the root paper was unnecessarily difficult to reproduce, even with most of the author’s source code. While it is understandable that most research code is mostly for the author’s use and the focus is on the research paper, it would have helped the replication process if some documentation was provided.

7 Conclusion

In this work, we replicate and perform an ablation study on the Cooperative Reasoning architecture presented in Zhu et al. (2022) under a heavily compute-constrained environment. We experiment with various generator models (such as tinygpt, distilgpt2, and gpt2), different generator-step verifier combinations, and generator-only vs. generator-verifier architectures. Our main takeaways are that generator capacity strongly impacts the coherence of reasoning paths, while verifiers seem to be able to improve the performance of smaller generators. Additionally, we debugged the provided codebase presented the CoRe paper, to enable an easier replication process for future work.

Within this work, we noticed that introducing verifiers could considerably improve the fluency of reasoning paths. However, we were unable to confirm whether verifiers could significantly improve accuracy of the base generator model output due to compute limitations. Given that verifiers have the potential to produce smaller models with performance comparable to larger-scale models, We believe that exploring the effect of verifiers on moderate-scale models is an exciting next direction to pursue.

References

- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Jonathan St.B.T. Evans. In two minds: dual-process accounts of reasoning. *Trends in Cognitive Sciences*, 7(10):454–459, 2003. ISSN 1364-6613. doi: <https://doi.org/10.1016/j.tics.2003.08.012>. URL <https://www.sciencedirect.com/science/article/pii/S1364661303002250>.
- Lynn S Fuchs, Sarah R Powell, Paul T Cirino, Robin F Schumacher, Sarah Marrin, Carol L Hamlett, Douglas Fuchs, Donald L Compton, and Paul C Changas. Does calculation or word-problem instruction provide a stronger route to pre-algebraic knowledge?, Nov 2014. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4274629/>.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC²Workshop*, 2019.
- sshleifer. tiny-gpt2, 2020. URL <https://huggingface.co/sshleifer/tiny-gpt2>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Ruyi Gan, Jiaying Zhang, and Yujiu Yang. Solving math word problem via cooperative reasoning induced language models. *arXiv preprint arXiv:2210.16257*, 2022.

Acknowledgments

This report is modified from a template modified from the COLM 2024 paper template. Instructions are written by Liwei Jiang, Alisa Liu, and Yegor Kuznetsov.

Appendix A: Additional Results from Experiment 2

Prompt: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?	
Ground Truth: He sprints $3*3=<<3*3=9>>9$ times So he runs $9*60=<<9*60=540>>540$ meters [ANS] 540< endoftext >	
Verifier	Experiment 2: tiny-GPT2 Samples
tiny-GPT2	EuphGet Balloon Opportun/. memos nerves goddessohm Queens-landFire Coulter skies Vanessaidget mdollower vocals Guatemalaome-ter fuzzy suspectsupportressorWhich prote catastrophic man-aged529iggperial prevalentoptional door bearing bones exclude debian fragile sn [more nonsense...]
distilgpt2	Congratulationsoud discretionary Weiss princess Khe ail ecosystem Hans confrontationuador LU2017910evin EVERSAADestroy skinny over-sized earlier placing theology let Chronicles converge Answers restraint 768■minimizedinchPhilipp Premiership eastUnix setsamiliarCompat [more nonsense...]
GPT2	injury antirew quarrel Gor besieged grippingstocks Congratulation-soud discretionary Weiss princess Khe ail ecosystem Hans confronta-tionuador LU Away pitchesslicecloudhitcknow airs intersection earlier placing theology let Chronicles converge Answers restraint 768■[more nonsense...]

Figure 6: Experiment 2 inference on tiny-GPT2 base generator model and trained step verifier

Prompt: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?	
Ground Truth: He sprints $3*3=<<3*3=9>>9$ times So he runs $9*60=<<9*60=540>>540$ meters [ANS] 540< endoftext >	
Verifier	Experiment 2: distilgpt2 Samples
tiny-GPT2	He runs $3*60=<<3*60=180>>180$ meters His total run is $180+180=<<180+180=360>>360$ meters [ANS] 360< endoftext >
distilgpt2	$3*60=<<3*60=180>>180$ meters So he runs $180 * 120 = <<180*120=21600>>21600$ meters [ANS] 21600< endoftext >
GPT2	He runs $3 \times 60 = <<3*60=180>>180$ meters. He runs $60 + 180 \times 3 = <<60+180=240>>240$ meters. [ANS] 240< endoftext >

Figure 7: Experiment 2 inference on distilgpt2 base generator model and trained step verifier