On Reward Functions For Self-Improving Chain-of-Thought Reasoning Without Supervised Datasets (Abridged Version)

Thomas Foster *
University of Oxford
thomf@robots.ox.ac.uk

Eltayeb Ahmed *
University of Oxford
eltayeb@robots.ox.ac.uk

Jonathan Cook *
University of Oxford
jcook@robots.ox.ac.uk

Shalev Lifshitz
University of Toronto

Tim Rocktäschel
Centre for AI, University College London

Jakob Foerster
University of Oxford

Abstract

Prompting a Large Language Model (LLM) to output Chain-of-Thought (CoT) reasoning improves performance on complex problem-solving tasks. Moreover, several popular approaches exist to "self-improve" the CoT reasoning abilities of LLMs on tasks where supervised (question, answer) datasets are already available. An emerging line of work explores whether self-improvement is possible without these supervised datasets, instead utilizing the same large, unstructured text corpora as used during pretraining. This would overcome the data availability bottleneck present in current self-improvement methods, and open the door towards *compute-only scaling* of language model reasoning ability. We investigate a fundamental question in this line of work: What constitutes a suitable reward function for learning to reason during general language model pretraining? We empirically demonstrate how different functions affect what reasoning is learnt and where reasoning is rewarded. Using these insights, we introduce a novel reward function called Reasoning Advantage (RA) that facilitates self-improving CoT reasoning on free-form question-answering (QA) data, where answers are unstructured and difficult to verify. We also explore the optimization of RA on general unstructured text using offline RL, and our analysis indicates that future work should investigate more powerful optimization algorithms, potentially moving towards more online algorithms that better explore the space of CoT generations.

1 Introduction

Large Language Models (LLMs) have become increasingly effective at solving complex reasoning tasks [Huang and Chang, 2022, Wei et al., 2023, Kojima et al., 2023, Havrilla et al., 2024a]. A key driver of this success has been the discovery of Chain-of-Thought (CoT) reasoning [Wei et al., 2023], whereby a model outputs a step-by-step "thought process" before arriving at a final answer.

While some CoT reasoning ability emerges naturally from pretraining on unstructured web-text data [Fu et al., 2023], it is through further supervised finetuning (SFT) on curated question-answering (QA) datasets [Saparov and He, 2023], as well as Reinforcement Learning from Human Feedback (RLHF) [Ouyang et al., 2022], that CoT becomes such a powerful tool. Considerable effort is being placed in curating large-scale (question, CoT, answer) datasets [Cobbe et al., 2021a, Saparov and He, 2023, Liu et al., 2023], often using an existing model to help generate this data [Zelikman et al., 2022, Zhang et al., 2024]. However, curating sufficiently challenging datasets across diverse domains is becoming increasing difficult and prohibitively expensive. For instance, a popular benchmark of just 500 graduate-level science questions with CoT reasoning and answers cost over $120,000 and required thousands of human expert hours [Rein et al., 2023].

*Equal Contribution.

Language Gamification Workshop at NeurIPS 2024.

To address these limitations, an emerging line of work explores self-improving CoT reasoning ability in a self-supervised setting—leveraging the large, unstructured datasets used for pretraining [Zelikman et al., 2024] instead of relying on curated QA or RLHF datasets. In this new setting, the LLM learns to produce CoT reasoning for the task of next-token prediction: **given n tokens from the pretraining corpus, the model generates a CoT and receives a reward based on how well the CoT helps predict the subsequent m tokens**. This is an exciting prospect, as we have trillions of tokens of unstructured text encompassing much of human knowledge. Therefore, learning to self-improve CoT reasoning on pretraining scale data might overcome the data availability bottleneck in current self-improvement methods, opening the door towards *compute-only scaling* of reasoning ability.

We investigate a fundamental problem in this emerging line of work: **What constitutes a suitable reward function for reasoning during general language model pretraining?** We reveal critical shortcomings in commonly used reward functions, including an inability to differentiate between meaningful CoT reasoning and random sequences (*what* reasoning to reward), as well as a tendency to incentivize reasoning at locations where predicting the subsequent tokens is trivial (*where* to reward reasoning). We introduce a novel reward function called Reasoning Advantage (RA), an augmentation of standard language modeling loss, and show that it addresses many of these limitations.

To facilitate more efficient study of self-improving CoT reasoning, we also introduce an open-ended QA dataset called MMLU-FREE-FORM, where solutions are challenging to verify. We demonstrate that RA is the only reward function which enables self-improvement of CoT reasoning on free-form QA data, improving zero-shot transfer accuracy on GSM8K [Cobbe et al., 2021b] by nearly 7%, compared to barely 0.5% when trained with other reward functions.

Using our Reasoning Advantage (RA) reward function, we conduct an initial experiment on self-improving CoT reasoning on general unstructured text using OpenWebMath [Paster et al., 2023], a collection of 14.7 billion tokens of maths-heavy text. We find that the offline RL algorithm employed is not sufficiently powerful to escape the local optimum of extremely conservative CoT reasoning that just summarizes previous information instead of trying to solve the problem. Future work should investigate more powerful optimization algorithms, potentially moving towards online algorithms that better explore the space of CoT generations. To facilitate future work, we will open-source all of our code, which runs on academic compute.

2 Reward Functions for Self-Improving CoT Reasoning

Given n tokens (a prefix $\mathbf{p}$) from a pretraining corpus, the model generates a CoT $\mathbf{r}$ and receives a reward based on how well this CoT helps predict the subsequent m tokens (the suffix $\mathbf{s}$). Previous works have primarily explored two reward functions for self-improving CoT reasoning: loss and accuracy. Here, we explore other potential reward functions and their characteristics from the perspective of facilitating self-improving CoT reasoning on unstructured web-text at pretraining scale. See Appendix A for a formal definition of self-improving CoT reasoning as reinforcement learning.

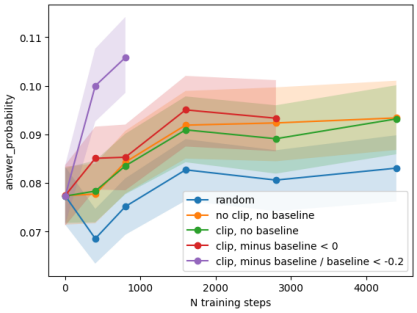
In this work, we do not consider using LLM-as-judge to evaluate or verify CoTs since: (1) it may rely on a stronger model, which is not self-improvement, and (2) while one could use the same model for both generation and verification, this approach incurs too much computational overhead to apply to pretraining scale data as it requires additional generations from the verifier. Thus, we choose to focus on rewards based on the language modeling loss of the suffix (see Appendix C for further discussion on why we investigate loss-based rewards specifically). In particular,

$$R(\mathbf{p}, \mathbf{r}, \mathbf{s}) = \log P(\mathbf{s}|\mathbf{p}, \mathbf{r}) = \sum_{i=0}^{m} \log P(\mathbf{s}_i|\mathbf{p}, \mathbf{r}, \mathbf{s}_{0:i}) \tag{1}$$

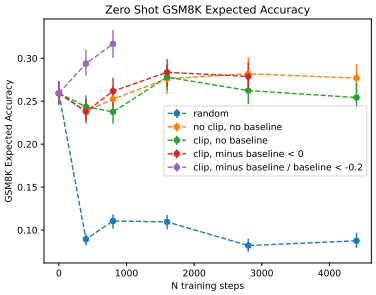
This family of reward functions requires only one forward pass, does not require an external strong model, and allows the model to place weight over a distribution of valid answers.

We investigate two main augmentations over the basic loss-based reward function from Equation 1: (a) *clipping* (aka clamping) the minimum value of the token-level log probabilities to some $-\epsilon$ such that $R_{\text{clipped}}(\mathbf{p}, \mathbf{r}, \mathbf{s}) = \sum_{i=0}^{m} \max\left[\log P(\mathbf{s}_i|\mathbf{p}, \mathbf{r}, \mathbf{s}_{0:i}), -\epsilon\right]$, and (b) subtracting a *baseline* value. In the main text of this paper, we focus on two main combinations of these augmentations (Appendix D.1 contains results for additional reward functions):

- **Delta Loss:** $R_{\text{DL}} = R(\mathbf{p}, \mathbf{r}, \mathbf{s}) - R(\mathbf{p}, \text{“ ”}, \mathbf{s})$, where we subtract the “Empty CoT” baseline.
- **Reasoning Advantage (RA):** $R_{\text{RA}} = \frac{R_{\text{clipped}}(\mathbf{p}, \mathbf{r}, \mathbf{s}) - R_{\text{clipped}}(\mathbf{p}, \text{“ ”}, \mathbf{s})}{R_{\text{clipped}}(\mathbf{p}, \text{“ ”}, \mathbf{s})}$, which is clipped delta loss normalized by the “Empty CoT” baseline.



(a) MMLU Test Performance



(b) GSM8K Transfer Performance

Figure 1: Reward function performance for self-improving CoT reasoning on MMLU-FREE-FORM. Only RA facilitates generalization to the MMLU test set and zero-shot transfer to GSM8k. Different reward functions yield different amounts of filtered data, which affects the number of training steps. The values after the $<$ sign are the reward thresholds used for filtering the CoTs (see Section 3.2).

3 Experiments

Our experiments aim to analyze the effectiveness of the reward functions introduced in Section 2. Namely: loss, delta loss, and RA. In Section 3.1, we evaluate whether these reward functions can (1) distinguish meaningful CoTs and (2) identify optimal locations to produce CoT reasoning. We find that RA performs best in both cases. In Section 3.2, we evaluate whether the reward functions can be used to self-improve CoT reasoning ability on free-form QA data. We find that only RA facilitates generalization. Appendix D.1 contains a discussion of results for additional reward functions, beyond those introduced in Section 2.

3.1 Reward Functions for Selecting What & Where to Reason

***What* reasoning is rewarded:** Here, we evaluate the ability of different reward functions to distinguish between three categories of CoTs: correct, incorrect, and randomly generated. First, we select 1,000 prefix-suffix pairs from random locations in FineWeb, a pretraining corpus of unstructured web-text data [Penedo et al., 2024]. Then, for each pair, we generate the three types of CoT: correct, incorrect, and random. "Correct" CoTs are generated using GPT-4o with post-rationalization—by showing GPT-4o both the prefix and suffix, but instructing the model to generate a CoT without explicitly repeating the suffix (similar to Zelikman et al. [2022]). "Incorrect" CoTs are generated by GPT-4o without post-rationalization—while these CoTs often exhibit sophisticated reasoning, they typically do not predict the exact suffix as well as the "correct" CoTs, which is enough for the purposes of this experiment. Finally, "random" CoTs consist of strings of random words and serve as our baseline.

Reward Function	*What* (Acc)	*Where* (AUC)
RA	**66.3**	**77.0**
Delta Loss	58.3	64.4
Loss	44.6	39.4

Table 1: Reward function performance for distinguishing CoT types (*What*) and identifying optimal CoT placement (*Where*).

To evaluate how well a reward function distinguishes between these CoT types, we compute the reward score for all CoTs—using Mistral-7B-Instruct [Jiang et al., 2023] to compute the log probabilities—and partition them into thirds: the top third labeled as "correct," the middle third as "incorrect," and the bottom third as "random." An effective reward function should rank the CoTs in the ideal order: correct > incorrect > random. The results in Table 1 demonstrate that RA performs best. Moreover, Figure 2 reveals that standard loss struggles primarily in distinguishing between "incorrect" and "random" CoTs. Interestingly, when we simplify to binary classification between only "correct" and "incorrect" CoTs, non-clipping methods perform similar to clipping methods, which suggests that the main advantage of clipping lies in distinguishing truly random reasoning.

***Where* reasoning is rewarded:** Here, we investigate how different reward functions predict the "most beneficial" location for CoT reasoning. Using 1,000 problems each from GSM8K [Cobbe et al., 2021b], CSQA [Talmor et al., 2018], and MMLU [Hendrycks et al., 2020], we first format each problem's question, multiple choice options, and answer as a single text string. We then create four (prefix, suffix) pairs per problem by splitting at different points: 1) mid-question, 2) after

the question but before the multiple choice options, 3) after the multiple choice options (*the ideal location for CoT reasoning*), and 4) mid-answer. This setup aims to mimic a key fact about unstructured pretraining text: not all locations are suitable for CoT reasoning. That is, reasoning may be unhelpful if produced too early (insufficient context) or too late in a document.

To evaluate each reward function, we frame this as a binary classification task: identifying the ideal location (after the multiple choice answers but before the solution) versus the three suboptimal locations. Using reward as a classifier and computing the AUC for this classification task, we find that RA performs best, followed by delta loss and standard loss. In particular, functions that use a baseline consistently outperform those without, with clipping providing additional improvement. Standard loss performs poorly, favoring locations late in the answer where suffix prediction becomes trivial. Table 1 summarizes these results, with detailed results available in Appendix D.1.

3.2 Learning to Self-Improve CoT Reasoning on Free-Form QA Data

To investigate the ability of different reward functions to facilitate self-improving CoT during pretraining, we create a new "free-form" QA dataset called MMLU-FREE-FORM by adapting the popular MMLU dataset [Hendrycks et al., 2020] to be closer to the unstructured text setting. Specifically, by removing its multiple-choice format and requiring models to generate full, unstructured answers—which are hard to verify. We use the entire labeled free-form solution as the suffix when computing rewards. This dataset provides a higher density of clear opportunities for CoT reasoning compared to typical pretraining corpora, since we know that reasoning is particularly beneficial when predicting answers to questions. Moreover, prior works have shown that LLM reasoning ability on MMLU can be improved with only few thousand labeled CoT examples. Thus, for the purposes of our investigations, MMLU-FREE-FORM enables a more compute and time efficient study of reward functions, acting as a stepping stone towards self-improving CoT reasoning on the type of truly unstructured text seen during pretraining (i.e., OpenWebMath [Paster et al., 2023]). Further discussion about MMLU-FREE-FORM can be found in Appendix D.2.

To self-improve CoT reasoning using MMLU-FREE-FORM as our dataset, we utilize a simple offline RL method. First, we generate 16 CoTs for each question and compute the reward for each CoT using the entire labeled free-form solution as the suffix. Then, we filter the CoTs with the highest reward [Dong et al., 2023], finetune on MMLU-FREE-FORM containing these self-inserted CoTs, and evaluate the trained model on a held-out test set. We test this pipeline using Mistral-7B-Base [Jiang et al., 2023] and find that **only RA facilitates generalization**—both on the in-domain MMLU test set (see Figure 1a) and on zero-shot transfer to GSM8k (see Figure 1b).

4 Challenges and Future Directions

To chart a course for future work, we present an exploratory experiment applying the offline RL procedure from Section 3.2 to the general language modeling setting. Specifically, we use our novel Reasoning Advantage (RA) reward function to self-improve CoT reasoning on OpenWebMath [Paster et al., 2023], a pretraining corpus of unstructured web-text data (see Appendix D.3 for further experiment details). However, this procedure fails to generalize when optimizing RA for more than a few steps, and is also unable to match the performance of vanilla language modeling without any self-inserted CoTs. Our analysis indicates that this method is not sufficiently powerful to escape the local optimum of extremely conservative CoT reasoning that just summarizes previous information instead of attempting to actually solve the problem. That is, since generating an incorrect CoT might lower the reward score more than just summarizing, the model learns this conservative strategy to minimize the risk of receiving a low reward.

To increase the diversity of explored CoTs, future work might use Quality-Diversity [Mouret and Clune, 2015] or other evolutionary techniques [Fernando et al., 2023, Samvelyan et al., 2024], which could generate more diverse CoTs. Better exploration may also be facilitated by using online RL, but the only existing method in this direction generates a CoT at every token in a document [Zelikman et al., 2024], which is highly inefficient. We believe that the computational feasibility of generating CoTs in large, offline batches and performing supervised finetuning is key to enabling the self-improvement of CoT reasoning at the pretraining scale. To facilitate further research on this important problem, we will open-source our offline RL code, which can be run on an academic compute budget. Moreover, we will release MMLU-FREE-FORM, which serves as a middle ground between the simple QA setting and general language modeling on unstructured text, for investigating self-improving CoT reasoning.

References

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

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.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Railneau. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*, 2024a.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance, 2023. URL `https://arxiv.org/abs/2305.17306`.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought, 2023. URL `https://arxiv.org/abs/2210.01240`.

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. Training language models to follow instructions with human feedback, 2022. URL `https://arxiv.org/abs/2203.02155`.

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, 2021a. URL `https://arxiv.org/abs/2110.14168`.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. Logicot: Logical chain-of-thought instruction-tuning, 2023. URL `https://arxiv.org/abs/2305.12147`.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. Star: Self-taught reasoner bootstrapping reasoning with reasoning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 15476–15488, 2022.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b, 2024. URL `https://arxiv.org/abs/2406.07394`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 2021b.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint arXiv:2310.06786*, 2023.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

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

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment, 2023.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015. URL `https://arxiv.org/abs/1504.04909`.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023. URL `https://arxiv.org/abs/2309.16797`.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, Tim Rocktäschel, and Roberta Raileanu. Rainbow teaming: Open-ended generation of diverse adversarial prompts, 2024. URL `https://arxiv.org/abs/2402.16822`.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning, 2019.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models, 2021.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

William Merrill and Ashish Sabharwal. The expresssive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Ankit Vani, Max Schwarzer, Yuchen Lu, Eeshan Dhekane, and Aaron Courville. Iterated learning for emergent systematicity in vqa. *arXiv preprint arXiv:2105.01119*, 2021.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022.

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

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

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024b.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL `https://arxiv.org/abs/2009.03300`.

A Background

CoT Reasoning Given n prefix tokens $\mathbf{p}$, performing CoT reasoning refers to an LLM $\mathcal{M}$ generating a sequence of reasoning tokens $\mathbf{r}$ before the m answer suffix tokens $\mathbf{s}$. The goal of generating CoT reasoning tokens before the final answer is to maximize $P_{\mathcal{M}}(\mathbf{s}|\mathbf{p}, \mathbf{r})$, the probably of the answer suffix tokens $\mathbf{s}$ conditioned on both the prefix $\mathbf{p}$ and the CoT reasoning tokens $\mathbf{r}$. The (prefix, suffix) pair can be any token sequence, ranging from question-answer pairs in mathematical datasets to arbitrarily split sentences from an unstructured text corpus.

Traditionally, CoT reasoning has been elicited by pre-pending few-shot examples of (question, CoT, answer) to the prefix [Wei et al., 2023]. This approach relies the pattern-completion tendencies of LLMs to continue generating CoTs for subsequent outputs. Alternatively, it has also become popular to elicit CoT reasoning by appending prompts like "Let's think step by step." to the prefix, especially for instruction-tuned models [Kojima et al., 2023].

Self-Improving CoT Reasoning as Reinforcement Learning We consider self-improvement to refer to any process where an LLM is finetuned on self-generated data, leading to performance gains without human intervention or assistance from a stronger model. This process can be framed as a Reinforcement Learning (RL) problem. In RL, an agent interacts with an environment by taking an action $a \in A$ in a state $s \in S$ to maximize cumulative reward. The agent receives a reward $R_t = R(s_t, a_t)$ after each action a_t and aims to learn a policy $\pi(a|s)$ that maximizes the expected cumulative discounted reward $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$, where $\gamma \in [0, 1]$ is the discount factor.

In the context of CoT generation, each token can be viewed as an action a_t, with the current string of generated tokens representing the state s_t so far. We focus on a sparse reward setting where rewards are zero until CoT generation is complete, and with a discount factor $\gamma = 1$. The reward function maps the prefix $\mathbf{p}$, CoT reasoning tokens $\mathbf{r}$, and answer suffix $\mathbf{s}$ to a real number $R(\mathbf{p}, \mathbf{r}, \mathbf{s}) \in \mathbb{R}$, with higher rewards for CoTs that better predict the suffix. As long as this reward function does not require external intervention from humans or more powerful models, we consider optimizing it through RL methods as self-improving CoT reasoning.

Self-Improving CoT Reasoning Using Supervised Datasets When a supervised dataset of (question, answer) pairs is available, accuracy can serve as a reward function:

$$R_{\text{acc}}(\mathbf{p}, \mathbf{r}, \mathbf{s}) = \begin{cases} 1 & \text{if } \arg\max_{\mathbf{s}'} P_{\mathcal{M}}(\mathbf{s}'|\mathbf{p}, \mathbf{r}) = \mathbf{s} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In this case, we can sample multiple CoTs and finetune on those that lead to correct answers [Dong et al., 2023, Zelikman et al., 2022]. Iterating this process yields increasingly high-quality CoT generation, and this iterative self-improvement is equivalent to online reinforcement learning [Zelikman et al., 2022]. There are also more complex methods, such as Process Reward Models (PRMs), which provide dense rewards for each step in a CoT and address credit assignment challenges [Ma et al., 2023, Wang et al., 2023, Lightman et al., 2023, Havrilla et al., 2024a].

Self-improving CoT Reasoning on General, Unstructured Text This setting explores the possibility of self-improving CoT reasoning given only an unstructured corpus of text, without access to a curated dataset of (question, CoT, answer) or (question, answer) pairs. In this setting, the model generates and inserts CoTs at various points in a sequence of tokens. For example, at various points in a web-document that shows how to apply the quadratic formula.

A key challenge in this setting is evaluating the performance of CoT reasoning tokens that are inserted into general, unstructured text. The exact-match accuracy-based reward R_{acc} is ineffective here, as it would almost always be zero. Instead, standard language modeling loss—the log-likelihood of the suffix conditioned on the prefix and CoT—-serves as a more natural starting point for a reward function:

$$R_{\text{loss}}(\mathbf{p}, \mathbf{r}, \mathbf{s}) = \log P_{\mathcal{M}}(\mathbf{s}|\mathbf{p}, \mathbf{r}) \tag{3}$$

The ultimate goal of our work is to advance the field towards this setting, enabling the self-improvement of CoT reasoning on general, unstructured text—without access to supervised datasets. In this paper, we specifically focus on developing and analyzing reward functions to address the unique challenges posed by this unstructured environment.

B Related work

LLM Reasoning Various works have looked at improving the reasoning capabilities of LLMs. Rajani et al. [2019] improve the commonsense reasoning ability of language models by training on human explanations for commonsense problems. Nye et al. [2021] generate tokens in a "scratchpad" for intermediate computations when solving multi-step reasoning problems. On difficult algorithmic tasks, Pfau et al. [2024] show that LLMs can even be trained to leverage meaningless filler tokens under dense supervision, in place of legible CoTs. Further, theoretical analyses by Merrill and Sabharwal [2023] and Feng et al. [2024] show that CoT improves the expressivity of Transformers [Vaswani et al., 2017].

LLM Self-improvement Using Supervised Datasets Iterated learning approaches involve LLMs generating new outputs and using "successful" ones to improve generation quality [Anthony et al., 2017, Vani et al., 2021, Polu et al., 2022]. Such methods have been applied to LLMs [Zelikman et al., 2022, Huang et al., 2022, Chen et al., 2024]. However, much of the research on LLM self-improvement has been limited to question-answer (QA) domains where accuracy is an appropriate success measure, such as multiple-choice opions or simple numeric answers. This limitation is evident in the policy gradient objective approximated by STaR [Zelikman et al., 2022]: $\nabla J(M, X, Y) = \sum_i \mathbb{E}_{\hat{r}_i, \hat{y}_i \sim p_M(\cdot|x_i)}[\mathbb{1}(\hat{y}_i = y_i) \cdot \nabla \log p_M(\hat{y}_i, \hat{r}_i|x_i)]$. It makes use of an indicator function with respect to ground truth labels (i.e., exact-match accuracy). Clearly, this breaks down in settings where ground truth labels are not available, such as the open-ended or "free-form" QA setting as well as the general language modeling setting. Havrilla et al. [2024b] show that Expert Iteration [Anthony et al., 2017], a self-improvement method based on iterative Supervised Fine-Tuning (SFT), outperforms RL in their evaluations. Building on prior work, the offline RL algorithm employed in our work is similar to RAFT [Dong et al., 2023], which also uses iterative SFT.

Process Reward Models (PRMs) [Ma et al., 2023, Wang et al., 2023, Lightman et al., 2023, Havrilla et al., 2024a] have been used to enhance reasoning using RL by rewarding individual problem-solving steps in a CoT. However, PRM training is computationally expensive, usually involving backtracking and resampling from specific points in the CoT. Moreover, the points from which one resamples are usually determined by hard-coded heuristics such as new line breaks.

Self-Supervised LLM Self-improvement Quiet-STaR [Zelikman et al., 2024] looks to self-improve reasoning during general language modeling on unstructured text. Zelikman et al. [2024] generate a CoT at *every* token in an unstructured text document, using the negative cross-entropy loss on the suffix tokens as a reward. They employ REINFORCE [Williams, 1992] to optimize the loss of the suffix $\mathbf{s}$ given a prefix $\mathbf{p}$ and a reasoning trace $\mathbf{r}$, with a baseline for variance reduction. Importantly, performing CoT reasoning at every token is highly computationally expensive, making it difficult to use for pretraining-scale datasets and also limiting the length of CoT sequences that can be learnt (the reasoning learnt in Quiet-STaR is quite short and simple). Regardless, Quiet-STaR provides key insights into *how* to optimize for reward on general, unstructured text—a very difficult problem. Our work aims to take a step back and investigate *what* reward we should be optimizing for in the first place, and whether we can take steps towards determining *where* is the best place to produce CoT reasoning (see experiments in Section 3).

C Important Criteria for Effective Reward Functions

There are several key criteria to consider when designing a reward function for self-improving CoT reasoning on unstructured text. Primarily, it should reward high-quality reasoning over CoTs containing logical errors or simply random characters. As shown in Section 3.1, this is not always the case. Moreover, for the purposes of *self-improving* CoT reasoning, the reward function must not depend on an stronger source of intelligence (i.e., using a more powerful LLM to verify the correctness of its CoT). Further, for reasonable use on pretraining scale datasets, evaluating the function should be fast and ideally parallelizable—requiring a minimal number of model forward passes.

Importantly, accuracy is a challenging metric to use on unstructured or free-form text, since answers often impossible to verify using exact-match. In these cases, it is possible to use an LLM-as-judge to predict whether two free-form answers match. However, as discussed in Section 2, using an LLM-as-judge either requires a stronger model (which is not self-improvement) or it requires using the

same model to generate a verification output for every generation (which is highly inefficient for pretraining-scale data). Thus, in this work, we investigate a family of loss-based reward functions.

D Additional Experiment Details, Results, and Visualizations

To compute the log probabilities for all reward functions, we use a Mistral-7B-Instruct [Jiang et al., 2023] model that has been finetuned on a small set of 1,000 GPT-4 generated CoTs, filtered for correctness (by providing the model with the correct answer and asking whether it corresponds). This finetuning allows us to start from an initial model that is familiar with the format:

```
### Question: <question> ### Thought <reasoning> ### Answer: <response>.
```

D.0.1 Additional Visualizations for What & Where to Reason

Figure 2 and Figure 3 provide additional visualization for the *What* and *Where* experimental results from Section 3.1, respectively.

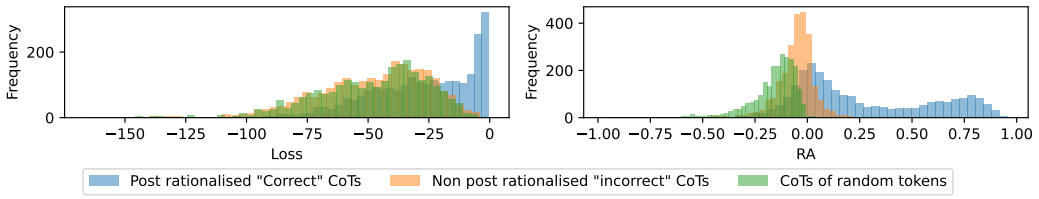


Figure 2: (*What* to reward) Distribution of reward scores across different CoT types using standard loss (left) and RA (right) reward functions. Each histogram shows the reward distribution for three categories: "correct" post-rationalized CoTs (blue), "incorrect" non-post-rationalized CoTs (orange), and "random" token CoTs (green). Notice that RA is better able to differentiate between incorrect and random CoTs. Moreover, the RA scores are normalized to the range [-1, 1], which may facilitate better learning. See Section 3.1 for more details.

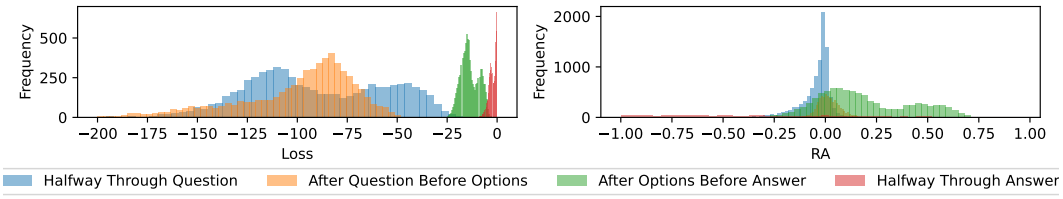


Figure 3: (*Where* to reward) Distribution of reward scores for CoTs inserted at different locations using standard loss (left) and RA (right) reward functions. Each histogram shows the reward distribution for four insertion points: halfway through question (blue), after question before multiple-choice options (orange), after multiple-choice options before answer (green), and halfway through answer (red). As mentioned in Section 3.1, we assume that "after multiple-choice options before answer" (green) is the optimal location to generate CoT reasoning. RA successfully scores these CoTs higher, while standard loss does not. Particularly, standard loss fails to prevent halfway-through-answer CoTs from receiving high rewards.

D.1 What & Where to Reason Results for Additional Reward Functions

Table 2 and Table 3 show full results for additional combinations of the augmentations described in Section 2. That is, for the entire family of loss-based reward functions. Moreover, they include results for the "empty CoT" baseline as well as two new types of baselines: "random CoT" and "mean loss". So, the three baselines are:

1. Empty CoT: $R(\mathbf{p}, \text{“ ”}, \mathbf{s})$ where the CoT is an empty string.
2. Mean loss: $\frac{1}{n}\sum_{i=1}^{n} R(\mathbf{p}, \mathbf{r}_i, \mathbf{s})$, where $\mathbf{r}_i$ is the ith generated CoT at this location.
3. Random CoT: $R(\mathbf{p}, \mathbf{r}_{\text{random}}, \mathbf{s})$, where $\mathbf{r}_{\text{random}}$ is a sequence of random tokens.

We explore incorporating these baselines both with normalization $(R - B)/B$ and without normalization $R - B$, where R is the reward score and B is the baseline value.

In Tables 2 and 3, RA outperforms standard loss and delta loss—as in the main text. However, it's worth mentioning that there are three combinations of augmentations that perform better than RA in Table 2 (*What* to reward), while performing much worse than RA in Table 3 (*Where* to reward). In the main text of this work, we choose to focus mainly on standard loss, delta loss, and RA since delta loss shows how the simple change of adding an empty CoT baseline improves results over standard loss, and RA shows the added effectiveness of clipping and normalization.

Name	Baseline	Clipping	Normalisation	Mean	q0.025	q0.975	Rank
Loss	none	none	none	44.6%	44.0%	45.4%	9
-	empty CoT	clipped	none	67.2%	65.7%	68.3%	3
RA	**empty CoT**	**clipped**	**yes**	**66.3%**	**64.5%**	**67.8%**	**4**
Delta Loss	empty CoT	none	none	58.3%	57.8%	58.9%	8
-	empty CoT	none	yes	58.8%	58.1%	59.8%	7
-	random CoT	clipped	none	80.4%	79.7%	81.4%	1
-	random CoT	clipped	yes	78.4%	77.8%	79.0%	2
-	random CoT	none	none	60.9%	60.1%	62.7%	6
-	random CoT	none	yes	60.9%	59.2%	63.1%	5
-	mean loss	clipped	none	30.8%	30.1%	31.7%	10
-	mean loss	clipped	yes	30.7%	29.9%	31.3%	11
-	mean loss	none	none	29.2%	28.7%	29.8%	13
-	mean loss	none	yes	30.7%	30.0%	31.7%	11

Table 2: Full results for *What* to reward experiment, showing all combinations of augmentations to the basic loss-based reward in Equation 1.

Name	Baseline	Clipping	Normalisation	Mean	q0.025	q0.975	Rank
Loss	none	none	none	39.4%	37.7%	40.8%	6
-	empty CoT	clipped	none	55.9%	52.5%	59.9%	4
RA	**empty CoT**	**clipped**	**yes**	**77.0%**	**75.3%**	**79.0%**	**1**
Delta Loss	empty CoT	none	none	64.4%	62.7%	67.0%	3
-	empty CoT	none	yes	73.0%	71.9%	74.3%	2
-	random CoT	clipped	none	29.8%	28.2%	30.6%	9
-	random CoT	clipped	yes	40.8%	38.9%	43.4%	5
-	random CoT	none	none	27.9%	26.7%	28.8%	11
-	random CoT	none	yes	27.3%	25.8%	28.6%	13
-	mean loss	clipped	none	27.7%	25.8%	29.2%	12
-	mean loss	clipped	yes	33.4%	32.5%	35.4%	7
-	mean loss	none	none	28.3%	26.5%	30.0%	10
-	mean loss	none	yes	32.1%	30.8%	33.4%	8

Table 3: Full results for *Where* to reward experiment, showing all combinations of augmentations to the basic loss-based reward in Equation 1.

D.2 Self-Improving CoT Reasoning on MMLU-FREE-FORM

As discussed in Section 3.2, self-improving CoT reasoning on MMLU-FREE-FORM is a stepping stone towards self-improving CoT reasoning on truly unstructured web-text data like OpenWebMath [Paster et al., 2023]. MMLU-FREE-FORM is a modified version of MMLU [Hendrycks et al., 2021], created by simply removing the multiple-choice options from each questions so that answers can be expressed in multiple different and equally valid ways (i.e., "Henry VIII had 6 wives" vs "In total there were 6 different woman who were married to Henry the Eighth"). Note that some problems become almost impossibly difficult to answer ("Which of the following is the correct method of multiple 32 x 18?"), as is the case with many next-token prediction problems in general unstructured

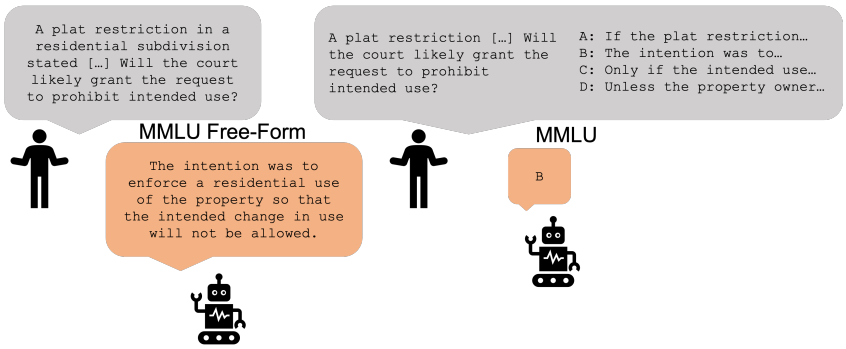


Figure 4: Example from MMLU-FREE-FORM, our modified version of MMLU [Hendrycks et al., 2020] designed to study improving CoT reasoning on unstructured, open-ended text. By removing multiple-choice options, answers become free-form so that they can can be expressed in multiple different and equally valid ways—this invalidates the use of accuracy without an external verifier. The left-hand side is the example from MMLU-FREE-FORM and the right-hand side is the original example from MMLU.

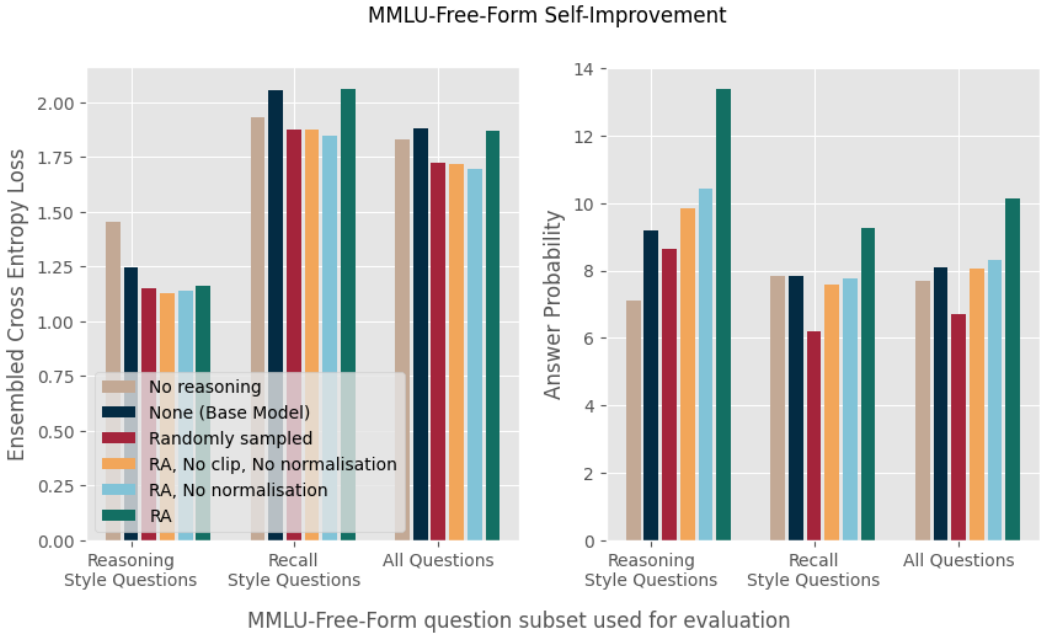


Figure 5: Performance breakdown on MMLU test set after self-improving CoT reasoning on MMLU-FREE-FORM. Results are shown for different question types. RA has better performance on reasoning-style questions compared to recall-style questions. **Left:** Ensembled cross-entropy loss (lower is better). **Right:** Correct answer probability (higher is better). See Section 3.2 for full experiment and method details.

text. MMLU-FREE-FORM offers a higher density of locations that afford useful CoT reasoning, making it particularly suitable for research on self-improving CoT reasoning using academic-scale computing resources. See Figure 4 for an example datapoint from MMLU-FREE-FORM.

See Figure 5 for a more complete breakdown of the results on the MMLU test set after self-improving CoT reasoning on MMLU-FREE-FORM using the method outlined in Section 3.2.

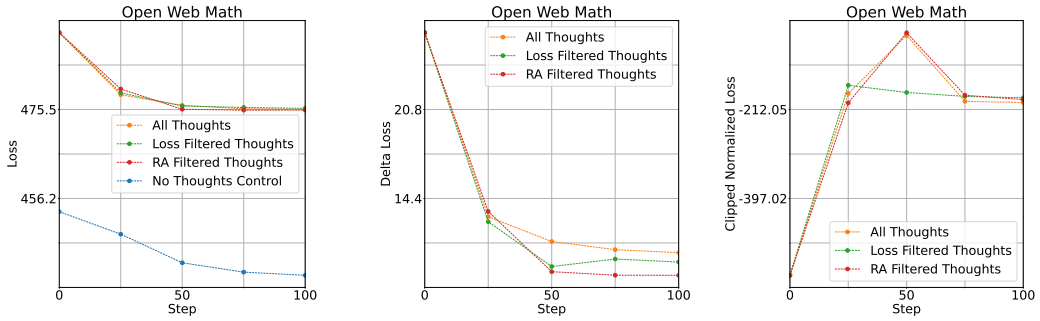
D.3 Self-Improving CoT Reasoning on OpenWebMath

In Section 4, we explore optimizing for RA using the unstructured OpenWebMath dataset [Paster et al., 2023]. Here, we provide additional experimental details and results.

Experimental Setup We first finetune Mistral-7B-Instruct [Jiang et al., 2023] on a small set of CoTs to learn the "[THOUGHT]...[/THOUGHT]" syntax. Then, we randomly sample 50,000 (prefix, suffix) pairs from OpenWebMath and generate CoTs for each location. From this pool of generated CoTs, we create three variants of an augmented OpenWebMath dataset by selecting 3,200 CoTs using different filtering methods:

- Random selection ("All Thoughts" in Figure 6)
- Best loss scores ("Loss Filtered Thoughts" in Figure 6)
- Best RA scores ("RA Filtered Thoughts" in Figure 6)

Additional Results Throughout training, we evaluate each model's CoT reasoning ability on a holdout set of OpenWebMath documents using the following procedure. At each checkpoint, we identify locations where "[THOUGHT]" is predicted as the most likely next token, generate CoTs at these points, and measure three metrics on the holdout documents (excluding CoT tokens but using them as context): standard loss, delta loss, and RA. Figure 6 show three plots—each measuring one of these metrics at various checkpoints throughout training. Notice that each line represents an entirely different model trained on differently filtered CoTs.



(a) Standard Loss (b) Delta Loss (c) RA

Figure 6: Standard loss, delta loss, and RA on the *holdout* documents measured at different training checkpoints (see Appendix D.3 for details). Each line represents an entirely different model trained on differently filtered CoTs. The filtering strategies are: random selection ("All Thoughts"), loss-based ("Loss Filtered Thoughts"), RA-based ("RA Filtered Thoughts"), and a "No Thoughts Control" baseline (trained on standard OpenWebMath documents without any self-inserted CoTs).

E Examples of Conservative Chain-of-Thought

As discussed in Section 4, our offline RL procedure applied to the unstructured OpenWebMath dataset [Paster et al., 2023] seems to converge to a local optimum where the model generates overly conservative CoT reasoning, merely restating or summarizing information rather than attempting actual problem-solving. Below are some examples of this behavior. Notice that the model knows about the "prefix" and "completion" from the prompts it received.

GSM8K Example:

```
Q: Richard lives in an apartment building with 15 floors. Each floor
    contains 8 units, and 3/4 of the building is occupied. What's the
    total number of unoccupied units In the building? [THOUGHT] The prefix
    provides information about an apartment building with 15 floors, each
    floor containing 8 units, and 3/4 of the building occupied. The
```

```
    completion likely provides the total number of unoccupied units in the
    building, based on the given information. [/THOUGHT]
A: 30
```

OpenWebMath Example 1:

```
In this article, by using norms($T$ and $C$), we present the concept of
    intuitionistic fuzzy implicative ideals,  intuitionistic fuzzy closed
    implicative ideals and intuitionistic fuzzy commutative ideals of
    $BCI$-algebras. Some interesting results of them are given.
    Characterisations of implicative ideals, closed implicative ideals and
    commutative ideals of $BCI$-algebras by using them are explored. By
    using intersections, direct products and homomorphisms, some
    interesting results are obtained [THOUGHT] Consider the prefix, which
    introduces the concept of intuitionistic fuzzy implicative ideals in
    $BCI$-algebras. The completion likely provides a definition or
    description of these ideals, along with their properties and
    applications. [/THOUGHT] .
```

OpenWebMath Example 2:

```
# Chapter 8 - Polynomials and Factoring - Chapter Review - 8-3 and 8-4
    Multiplying Binomials: 31

$9r^{2}-12r+4 [THOUGHT] The prefix provides a list of equations, each with
    a variable and a coefficient. The completion likely provides the
    solution to each equation, using the variable and coefficient to
    determine the value of the equation. The completion may also provide a
    step-by-step explanation [/THOUGHT] $

#### Work Step by Step

Simplify and write in standard form $(3r-2)^{2}$ Rewrite as:
    $(3r-2)(3r-2)$ Foil $9r^{2}-6r-6r+4$ Combine like terms $9r^{2}-12r+4$

After you claim an answer you'll have 24 hours to send in a draft. An
    editor will review the submission and either publish your submission
    or provide feedback.
```

F Societal Impact

While our work is primarily analytical and does not introduce new models, the broader direction of self-improving CoT reasoning on large-scale unstructured text datasets could significantly enhance LLMs' problem-solving capabilities—if successful. Such advances would amplify both the benefits and risks associated with current language models, warranting continued attention from the research community on ensuring responsible development.