
Learning to Reason by Failing: Offline RL on Sub-optimal Rollouts Scales Synthetic Data by 8x

Anonymous Authors¹

Abstract

Training on model-generated synthetic data is a promising approach for finetuning LLMs, but it remains unclear when it helps or hurts. In this paper, we investigate this for reasoning problems via an empirical study, followed by a theoretical formalization. First, we find that while the typical approach of finetuning a model on synthetic correct or *positive* problem-solution pairs generated by capable models offers modest performance gains, sampling more correct solutions from the finetuned learner **doubles** the sample efficiency of synthetic data. At the same time, training on model-generated positives can amplify spurious correlations, resulting in flat or even inverse scaling trends as the amount of data increases. Surprisingly, we find that several of these issues can be addressed if we also utilize *negative* responses, *i.e.*, model-generated responses that are deemed incorrect via final answer checking. Crucially, these negatives must be constructed such that the training can appropriately recover the utility or credit of each intermediate step in the negative response. With this *per-step* scheme, we are able to attain consistent gains over only positive data, attaining performance similar to amplifying the amount of synthetic data by **8x**. We show that training on per-step negatives can help to unlearn spurious correlations in the positive data, and is equivalent to advantage-weighted reinforcement learning (RL), implying that it inherits benefits of RL over imitating positive data alone.

1. Introduction

Training large language models (LLMs) relies on the ability to train on large amounts of high-quality data. It is predicted that we will run out of high-quality internet data by

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2026 (Villalobos et al., 2022; Liu et al., 2024), necessitating training on model-generated data, or what is commonly referred to as *synthetic data*. Recent trends illustrate that scaling up synthetic data can lead to improvements (Li et al., 2024; Chen et al., 2024) on hard reasoning problems, while other results illustrate that training on synthetic data can steer the performance of the model into a downward spiral (Shumailov et al., 2023; Alemohammad et al., 2023; Gerstgrasser et al., 2024)—amplifying biases, misinformation, and undesired stylistic properties. Thus while *in principle*, synthetic data could potentially address data scarcity, it must be designed in an appropriate manner to be effective. However, due to a lack of an understanding of how synthetic data contributes to LLM behavior, it is unclear how to best use synthetic data in practice.

To provide clarity on the role of synthetic data, we aim to understand its impact on LLM capabilities via a study on reasoning problems, a prevalent scenario where synthetic data is used. Typically, in this setting, synthetic data corresponds to correct or *positive* model-generated responses for a novel set of initial problems synthesized by prompting capable models (Li et al., 2024; Liu et al., 2023). The resulting model is then evaluated on a held-out set of problems drawn from a test set. Perhaps as expected, we find that performance improves when finetuning models on positive synthetic responses, though the scaling rates for performance improvement are often substantially slower than those observed during pretraining. Concretely, we find that under the scaling law of Zhang et al. (2024a), the error rate scales as $\approx D^{-0.05}$ to $D^{-0.15}$ in the size D of synthetic dataset. Second, we observe that not all types of positive synthetic data are equally effective: often positive responses sampled from the learner are as effective as 2x synthetic data from bigger models in improving performance. This is because responses from a similar model are “easier-to-fit” than those from a more capable model, resulting in reduced memorization (Kang et al., 2024; Tirumala et al., 2022) during finetuning. We also observe that if the positive response contains incorrect/irrelevant intermediate steps, training on such data often incentivizes the model to overfit on spurious correlations, leading to a flat or even inverse scaling with more data.

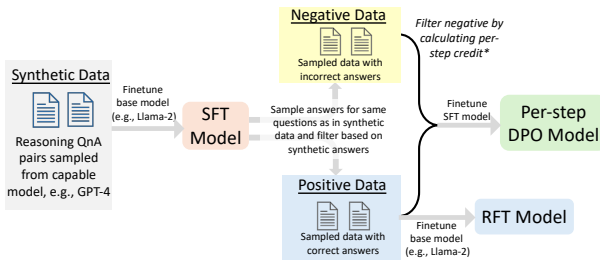


Figure 1: Positive and negative synthetic data: Pictorial representation of positive/negative synthetic data definitions we use and how they are fed to SFT, RFT and DPO.

Perhaps surprisingly, we find that the aforementioned pathologies of training on positive data only can be addressed if we also utilize synthetic *negative* responses: responses generated by the model that do not result in obtaining a correct final answer. One way to utilize negative responses is via methods such as direct preference optimization (DPO) (Rafailov et al., 2023). While performance of standard DPO (Rafailov et al., 2023) largely flatlines as the synthetic problems are scaled up (Figure 5), we are able to attain consistent improvements if the negative data is generated appropriately. Our intuition is that instead of contrasting arbitrary correct and incorrect responses, we contrast positive and negative responses that depict good and bad choices for the more “critical” intermediate steps (Hwang et al., 2024): steps that the model must carefully produce so as to succeed at the problem. In other words, critical steps are those which the model is unable to recover from, and hence, must be emphasized. With this scheme, we are able to attain consistent gains over only positive data, **attaining performance similar to scaling up positive synthetic data by 8x**. We also show that training on this sort of negative data evades spurious correlations introduced by training on positive data alone via a controlled study.

To theoretically understand our empirical findings, we build a conceptual model of how training on this data benefits performance. Formally, we show that this construction of negative data, which emphasizes “critical” tokens (Figure 6) enables us to perform credit assignment, and is equivalent to training the model with per-step advantage-weighted reinforcement learning (RL) (Peng et al., 2019) on a mixture of positive and negative synthetic data. Specifically, these advantage values are computed under an optimal value function induced by sampling multiple responses under the SFT policy obtained by training on only the positive data. This reduction of using negative data to advantage-weighted RL enables us to conceptually compare it to training on positive data, which corresponds to imitation learning (*i.e.*, behavioral cloning) on expert data. Building on theoretical results in RL (Kumar et al., 2022), we are also able to show that when advantages can be estimated reliably, advantage-weighted RL will be significantly more sample-efficient compared to imitation learning. Overall, this abstraction and conceptual model explains the utility of negative syn-

thetic data over only positive synthetic data.

Our contribution is a study of the role of synthetic data in improving reasoning capabilities of LLMs. We derive scaling laws for positive and negative data on common reasoning benchmarks such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), and observe that: **(a)** training on positive synthetic data from capable models results in scaling rates that are significantly slower than standard empirical risk minimization; **(b)** training on model-generated positive synthetic data can improve sample efficiency by 2x but also amplifies spurious correlations; **(c)** appropriate ways of constructing learner-specific negative data with emphasis on critical steps, results in a performance boost equivalent to scaling up positive data 8x; **(d)** training with negative data provides a mechanism to unlearn spurious correlations; and **(e)** we present a conceptual model inspired from RL to explain our observations for synthetic data.

2. Synthetic Data Generation Pipeline

Our goal in this paper is to understand the role of synthetic data in producing strong language model reasoners. Building on the recipe of Li et al. (2024); Liu et al. (2023), we collect synthetic data consisting of both novel problems designed by capable models such as GPT4 (Achiam et al., 2023) and Gemini 1.5 Pro (Reid et al., 2024), and responses to these problems, obtained from the same models. Concretely, we focus on two mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021).

Synthetic data pipeline. Our synthetic data generation is done in two phases. First, given a dataset $\mathcal{D}_{\text{real}} = \{(x_i^r, y_i^r)\}$ of problems $x_i^r \sim p_{\text{real}}(x)$ and solution traces $y_i^r \sim p_{\text{real}}(y | x_i)$, we prompt one of the highly-capable models with a uniformly random sample $(x_i^r, y_i^r) \in \mathcal{D}_{\text{real}}$ and ask the model to generate a new problem x_i such that it is similar to the real problem x_i^r , in a way that a feasible solution exists. Second, we ask the model to provide a solution trace answer y_i with step-by-step reasoning (exact prompts for x_i, y_i are borrowed from Li et al. (2024), shown in Appendix E). We assume that the answers generated via this process are accurate, and perform lightweight filtering step to remove duplicates, badly-formatted answer traces, and model failures. Based on the above, for any synthetic problem and solution pair (x, y) , we can define a binary reward function $r(y, \hat{y}) \mapsto \{0, 1\}$, which verifies if a new solution trace \hat{y} is correct or not. This is implemented with a set of answer extraction and string matching tools borrowed from (Yu et al., 2024; Li et al., 2024). We say that a new trace \hat{y} is a *positive* trace if it produces the correct final answer *i.e.*, $r(\hat{y}, y) = 1$, and *negative* if it produces an incorrect final answer, *i.e.*, $r(\hat{y}, y) = 0$. By definition, $r(y, y) = 1$, and the original trace y is always positive.

Positive and negative datasets. The above process induces a joint distribution $p_{\text{syn}}(\mathbf{x}, \mathbf{y})$, *iid* samples from which yields positive synthetic dataset \mathcal{D}_{syn} . We note that the sampling process for \mathcal{D}_{syn} is designed to ensure that the induced marginal distribution over synthetic problems $p_{\text{syn}}(\mathbf{x})$ is close to $p_{\text{real}}(\mathbf{x})$. We will use \mathcal{D}_{π}^+ to denote the positive dataset of $(\mathbf{x}, +\hat{\mathbf{y}})$ where $+\hat{\mathbf{y}}$ is a positive solution trace generated from some policy $\pi(\cdot | \mathbf{x})$. For a positive $+\hat{\mathbf{y}}$ and negative $-\hat{\mathbf{y}}$ trace, sampled from the same policy $\pi(\cdot | \mathbf{x})$, we denote a dataset over problems and solution pairs: $(\mathbf{x}, +\hat{\mathbf{y}}, -\hat{\mathbf{y}})$ as \mathcal{D}_{π}^{\pm} .

Reasoning steps. The trace \mathbf{y}_i consists of several intermediate steps: $\mathbf{y}_i = [\mathbf{y}_{i,1}, \dots, \mathbf{y}_{i,L}]$. We assume each solution trace has at most L steps, and use $\mathbf{y}_{1:t}$ to denote the subsequence of first t steps in the trace. Since mathematical reasoning problems require step-by-step computation, simply arriving at an incorrect final answer does not mean that all individual steps in a negative $\hat{\mathbf{y}}$ are incorrect. In fact, given previous steps $\hat{\mathbf{y}}_{1:t-1}$ the following intermediate calculation $\hat{\mathbf{y}}_t$ is often correct. Similarly, a positive $\hat{\mathbf{y}}$ may also have incorrect reasoning steps. In fact, even the original answers generated by more capable models in \mathcal{D}_{syn} may also contain incorrect reasoning steps, and training on such traces may actually lead to unintended consequences (Section 4).

3. Learning from Synthetic Data

In this section, we discuss various algorithms for learning from the synthetic dataset \mathcal{D}_{syn} discussed in the previous section, as well as positive and negative solution traces generated using a model.

Supervised and rejection finetuning (SFT and RFT).

Given positive synthetic \mathcal{D}_{syn} , perhaps the most straightforward approach (and the most prevalent) is to learn $\pi_{\text{sft}}(\cdot | \mathbf{x})$ on this data via supervised next-token prediction: $\pi_{\text{sft}}(\cdot | \mathbf{x}) := \arg \max_{\pi} \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \mathcal{D}_{\text{syn}}} \log \pi(\mathbf{y} | \mathbf{x})$. Another option is to train via supervised next-token prediction on problems in \mathcal{D}_{syn} , but when using a positive solution trace $\hat{\mathbf{y}}$ sampled from $\pi_{\text{sft}}(\cdot | \mathbf{x})$, instead of positive synthetic responses from the capable models in \mathcal{D}_{syn} . Akin to rejection finetuning (RFT (Yuan et al., 2023)) or STaR (Zelikman et al., 2022), sampling from $\pi_{\text{sft}}(\cdot | \mathbf{x})$ once is not guaranteed to give a positive response, and we instead sample M times for each \mathbf{x} and construct the dataset $\mathcal{D}_{\pi_{\text{sft}}}^+$ of SFT policy generated positive responses. Then, we apply the next-token prediction loss on $\mathcal{D}_{\pi_{\text{sft}}}^+$.

Preference optimization. Beyond positive data, we can also learn from negative synthetic data generated from the SFT policy, especially when contrasted with positive responses. However, learning from negative data presents multiple open design questions pertaining to the construction of negative traces, and the choice of the loss function, and simple supervised fine-tuning will not be a good choice

since it will incentivize the model to produce more errors. Therefore, we utilize a contrastive training approach, direct preference optimization (DPO (Rafailov et al., 2023)) for incorporating negative data from π_{sft} . In a nutshell, DPO trains a policy using the following preference optimization objective:

$$\mathbb{E}_{\mathcal{D}_{\pi_{\text{sft}}}^{\pm}} \left[\sigma \left(\beta \log \frac{\pi(+\mathbf{y} | \mathbf{x})}{\pi_{\text{sft}}(+\mathbf{y} | \mathbf{x})} - \beta \log \frac{\pi(-\mathbf{y} | \mathbf{x})}{\pi_{\text{sft}}(-\mathbf{y} | \mathbf{x})} \right) \right]. \quad (1)$$

We consider two objectives that construct negative data $-\hat{\mathbf{y}}$ in distinct ways and subsequently train the model on that data using Equation 1. The first variant we study is *naïve DPO*, which simply samples negative data $-\hat{\mathbf{y}} \sim \pi_{\text{sft}}(\mathbf{y} | \mathbf{x})$ from the SFT policy and adds $(\mathbf{x}, \mathbf{y}, -\hat{\mathbf{y}})$ to $\mathcal{D}_{\pi_{\text{sft}}}^{\pm}$. The second variant is *per-step DPO* (Hwang et al., 2024), which first samples a complete solution trace $\hat{\mathbf{y}}_{1:L}$ from π_{sft} and then determines the “first pit” $\hat{\mathbf{y}}_c$, such that any completion $\hat{\mathbf{y}}_{c+1:L} \sim \pi_{\text{sft}}(\cdot | \mathbf{x}, \hat{\mathbf{y}}_{1:c})$, sampled conditioned on \mathbf{x} , and previous steps $\hat{\mathbf{y}}_{1:c}$ leads to incorrect answers for a majority of the Monte-Carlo rollouts. Given the first pit $\hat{\mathbf{y}}_c$, the triplet $(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}_{1:c})$ is added to the negative dataset $\mathcal{D}_{\pi_{\text{sft}}}^{\pm}$.

4. Positive Data Improves Coverage, But Amplifies Spurious Correlations

We first analyze the influence of scaling up positive synthetic data on GSM8K and MATH. In this experiment, we fine-tune DeepSeek-Math-7B (Bi et al., 2024) and Llama2-7B (Touvron et al., 2023) models (details in Appendix H) on varying sizes of \mathcal{D}_{syn} , constructed out of a 5:1 mixture of GPT-4-turbo (Achiam et al., 2023) and Gemini-1.5 Pro (Reid et al., 2024). We obtain a series of SFT policies on this data scaling ladder. We then train a series of models by running one iteration of RFT on data obtained from the SFT policies at each step.

Scaling results with positive synthetic data GPT-4 and Gemini 1.5 Pro.

Since we assume that the more capable models generate correct solutions for new problems, by scaling \mathcal{D}_{syn} we are increasing *coverage* under p_{real} , *i.e.*, adding new \mathbf{x}, \mathbf{y} with non-zero probability under p_{real} . In Figures 2(a,b), we plot the test error rate of the SFT policy as \mathcal{D}_{syn} is scaled. As expected, we observe that the test error rate on both GSM8K and MATH improves with more positive data. Further, by simply fitting the parametric scaling law from (Zhang et al., 2024a), for $D := |\mathcal{D}_{\text{syn}}|$, we find that the scaling trends decay as $\approx D^{-0.15}$ on GSM8K and $\approx D^{-0.05}$ on the harder MATH dataset, with similar trends for the corresponding pass@5 error rates. Since these scaling trends are much more underwhelming than those for pre-training (Hoffmann et al., 2022), this perhaps implies that samples in \mathcal{D}_{syn} are indeed improving coverage over samples in $p_{\text{real}}(\mathbf{x}, \mathbf{y})$, but maybe not as efficiently as sampling *iid* samples directly from it.

Scaling results with positive synthetic data from 7B SFT policy. Previously, we scaled problems in \mathcal{D}_{syn} by querying

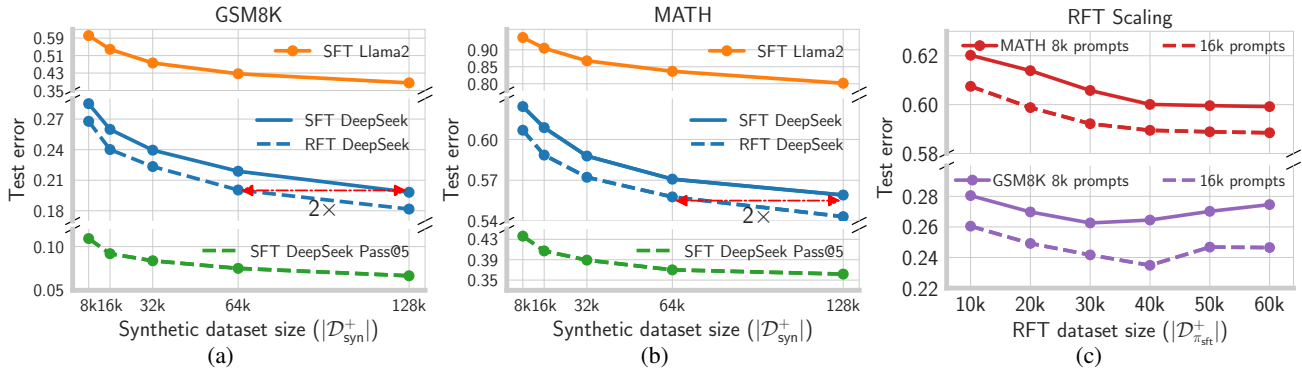


Figure 2: Positive data scaling laws: On GSM8K (a) and MATH (b), we evaluate SFT trained on \mathcal{D}_{syn} and RFT that uses SFT policy generated positives ($\mathcal{D}_{\pi_{\text{sft}}}^+$), as we scale \mathcal{D}_{syn} , observing $\mathcal{D}_{\pi_{\text{sft}}}^+$ to be 2 \times as effective as \mathcal{D}_{syn} . In (c), we plot performance of RFT the number of correct solutions in $\mathcal{D}_{\pi_{\text{sft}}}^+$ are scaled, for a fixed set of 8k/16k problems from \mathcal{D}_{syn} , observing that scaling model positives can amplify spurious correlations.

GPT-4 and Gemini-1.5. Now, for existing problems in \mathcal{D}_{syn} we generate new responses by sampling from the π_{sft} trained on problems+solutions in \mathcal{D}_{syn} . For any $(x, y) \in \mathcal{D}_{\text{syn}}$ we generate verified positive solution traces $\hat{y} \sim \pi_{\text{sft}}$ s.t. $r(\hat{y}, y) = 1$. Following Yuan et al. (2024a), to ensure we sample enough correct responses, we sample 100 times from π_{sft} and generate RFT datasets $\mathcal{D}_{\pi_{\text{sft}}}^+$, where each problem has almost 4 correct and diverse solutions. Next, we finetune the pretrained DeepSeek-Math-7B model on these new series of RFT datasets and plot the performance on GSM8K and MATH (Figure 2(a,b)). First, we observe that for any size of \mathcal{D}_{syn} , the performance of the RFT model is better than the corresponding SFT model, and the difference remains consistent as we scale \mathcal{D}_{syn} . Surprisingly, this indicates that training on positive answer traces from the 7B $\pi_{\text{sft}}(y | x)$ can lead to better performing policies than capable models.

What is the value of positives from $\pi_{\text{sft}}(y | x)$? If sampling from π_{sft} also improves coverage and performance, then should we scale problems and solutions in \mathcal{D}_{syn} , or just solutions in $\mathcal{D}_{\pi_{\text{sft}}}^+$? To answer this, we need to assign a value to the RFT dataset $\mathcal{D}_{\pi_{\text{sft}}}^+$ in terms of $|\mathcal{D}_{\text{syn}}|$. We do this by training SFT policies on \mathcal{D}_{syn} of sizes 8k and 16k, and then generating RFT datasets from the corresponding SFT policies where we only add more correct solution traces (for the same problems) and scale RFT data from 10k to 50k (unlike RFT data in Figure 2(a,b) where both questions and answers scale). In Figure 2(c) we plot the error rate of DeepSeek-Math-7B finetuned on the different sizes of $\mathcal{D}_{\pi_{\text{sft}}}^+$. Comparing the lowest values of the curves in Figure 2(c) with \mathcal{D}_{syn} scaling in Figure 2(a,b), we note that performance from $\mathcal{D}_{\pi_{\text{sft}}}^+$ is 2 \times the size of \mathcal{D}_{syn} used to train π_{sft} . We also note that performance can plateau (or worsen in the case of GSM8K) as we scale up $\mathcal{D}_{\pi_{\text{sft}}}^+$ by a lot. This is because $r(\cdot, y)$ is unable to verify the correctness of each step in the positive solution traces in $\mathcal{D}_{\pi_{\text{sft}}}^+$. Later, we see how incorrect steps induce spurious correlations that get

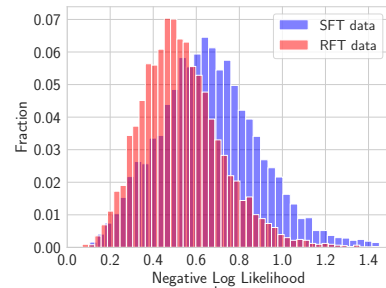


Figure 3: Under base LLM, $\mathcal{D}_{\pi_{\text{sft}}}^+$ has higher likelihood than \mathcal{D}_{syn} .

amplified as we scale positive data, explaining this drop.

Why is self-generated positive data more sample-efficient? From our result above, we find that solutions sampled from π_{sft} (trained on \mathcal{D}_{syn}) yield better models, as good as those trained on $2 \times |\mathcal{D}_{\text{syn}}|$. This finding is surprising since one might expect more capable GPT-4/Gemini models to present better solutions, training on which should lead to good performance, akin to distillation (Sharma et al., 2024), but this is not the case. Our results are consistent with the study of memorization in LLMs (Kang et al., 2024; Hartmann et al., 2023; Tirumala et al., 2022), which shows that pretrained (base) LLMs tend to memorize “hard-to-fit” and “out-of-pretraining-distribution” responses during fine-tuning, resulting in imperfect generalization. In contrast, correct response traces produced by π_{sft} on problems from \mathcal{D}_{syn} are not as hard-to-fit or as out-of-distribution, since they are obtained from a model that is “close” to the base LLM. We confirm this hypothesis with a histogram of negative log-likelihood values of the SFT and RFT data under the base LLM (Figure 3). Hence, we expect STaR/RFT to alleviate the memorization problem on a large chunk of examples. This finding also corroborates Yuan et al. (2023)’s result that lower the perplexity of SFT data under the base model, the smaller the gap between SFT and RFT performance. Note that one may also attribute better performance of RFT to improved coverage from multiple answers in $\mathcal{D}_{\pi_{\text{sft}}}^+$ for each question in \mathcal{D}_{syn} . But, we find that even when RFT data is

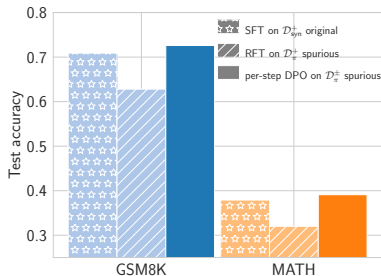


Figure 4: Spurious correlations in RFT data hurt performance.

restricted to one solution per question, LLM trained on it outperforms SFT consistently by $> 1\%$. Since verification is cheap, we can sample more solutions and also benefit from coverage.

SFT/RFT policy suffers from spurious correlations in positive synthetic data. While RFT data maybe “easier-to-fit”, in Figure 2(c) we also note that continuing to scale RFT data leads to test error saturation, or even worse test error. This is unlike scaling of problems and solutions in SFT data (in Figure 2(a,b)). This failure can be attributed to the presence of incorrect/irrelevant steps that are not detected by our verifier, since it only verifies the final answer (see Appendix H, F for examples). For a problem x , when the LLM is trained with supervised next-token prediction on some positive sub-optimal y in the RFT data, with incorrect step y_k , it is likely to overfit on spurious correlations between the sub-optimal subsequence $y_{1:k}$, and the following valid step y_{k+1} , when trying to maximize $\pi(y_{k+1} | y_{1:k}, x)$. To verify this hypothesis, we amplify the presence of these spurious steps. Specifically, for each question in \mathcal{D}_{syn} we sample “spurious steps” from π_{sft} trained on it, *i.e.*, steps which lead to the incorrect answer with high probability under π_{sft} (we sample multiple completions conditioned on the same spurious step to check how likely it leads to the correct final answer). Then, we interleave the solution traces in the RFT data with these spurious steps. Note, that all traces in the RFT data are still positive since, they all lead to the correct answer eventually. We find that the LLM trained on this sub-optimal spurious RFT data performs worse than the π_{sft} policy itself.

Takeaways for scaling positive synthetic data

- While positive data from GPT-4/Gemini-1.5 improves coverage over new problems and solutions, positive data from SFT policy trained on it is 2× more sample efficient.
- When positive data from π_{sft} contains spurious steps, scaling leads to worse test errors.

5. Negative Synthetic Data Enables Per-Step Credit Assignment

The spurious correlations from Section 4 correspond to intermediate irrelevant or incorrect steps that are able to

still steer the model towards the correct response on some training problems, but derail it otherwise. In this section, we present a conceptual model for constructing negatives that enables us to perform *per-step credit assignment*, and show that this approach can help us address these failure modes of positive data. We show that per-step DPO from Section 2 is a variant of this more general approach. We will then analyze scaling laws with negative data and empirically demonstrate that carefully constructed negative data can address issues with memorization. Finally, we theoretically prove that negative data improves sample-efficiency of \mathcal{D}_{syn} .

5.1. Conceptual Model: Constructing Negatives to Enable Per-Step Credit Assignment

While naïvely contrasting an entire positive response $+y$ against an entire negative response $-y$ will increase the likelihood of *each* step that appears in $+y$ (even when incorrect or irrelevant) and reduce likelihood on each step appearing in $-y$ (even when accurate and relevant), it does not account for the importance of each step. Formally, given a negative solution trace $-y$, we would want to identify the first *critical* step where the model introduces a flaw $-y$, and emphasize alternate correct completions from this step that the model could have still produced. Likewise, given a positive solution trace, $+y$, we would like to identify if a given step $+y_i$ does not make progress towards the solution by identifying if there exist alternatives from its predecessor step, $+y_{1:i-1}$, which now presents a key decision-making point. **What are these critical steps and how can we identify them procedurally?**

Value functions. We can formalize this notion of a critical step under the notion of value functions from reinforcement learning (RL). Recall that both $+y$ and $-y$ are sampled from π_{sft} . For problem x , with correct solution y , a response \hat{y} with a sequence of steps $\hat{y}_{1:i-1}$, and a candidate step \hat{y}_i , we define the value function for step y_i , and previous steps under some policy $\tilde{\pi}$ as:

$$Q_{\tilde{\pi}}(\underbrace{x, \hat{y}_{1:i-1}}_{\text{state}}, \underbrace{\hat{y}_i}_{\text{action}}) = \underbrace{\mathbb{E}_{\mathbf{y}_{i+1:L}^{\text{new}} \sim \tilde{\pi}(\cdot | x, \hat{y}_{1:i})} [r([\hat{y}_{1:i}, \mathbf{y}_{i+1:L}^{\text{new}}], \mathbf{y})]}_{\text{expected future reward under new actions sampled by policy } \tilde{\pi}} \quad (2)$$

Intuitively, for any partial solution upto i steps, this Q-function evaluates the probability of succeeding at solving the problem given the remaining budget of $L - i$ more steps, in expectation over all possible futures sampled from some policy $\tilde{\pi}$. Our conceptual model treats the policy $\tilde{\pi}$ as an algorithmic design choice that can differ for algorithms using negative data. As we see later, choosing $\tilde{\pi}$ as the Best-of-K distribution around π_{sft} (denoted as BoK(π_{sft})) enables a particularly interesting tradeoff between Q-value estimation and policy improvement. Another common choice is π_{sft} itself. Now, for any given step \hat{y}_i , we can define its *advan-*

275 *tage* as the relative change in $Q_{\tilde{\pi}}$ when adding step \hat{y}_i in
 276 comparison with other possible candidates for step i :

$$277 \quad A_{\tilde{\pi}}(\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}; \hat{\mathbf{y}}_i) \\ 278 \quad = Q_{\tilde{\pi}}(\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}, \hat{\mathbf{y}}_i) - Q_{\tilde{\pi}}(\mathbf{x}, \hat{\mathbf{y}}_{1:i-2}, \hat{\mathbf{y}}_{i-1}). \quad (3) \\ 279$$

280 Equation 3 is identical to the definition of advantage of
 281 an action (*i.e.*, $\hat{\mathbf{y}}_i$) at a state $(\mathbf{x}, \hat{\mathbf{y}}_{1:i-1})$ from RL (Sutton &
 282 Barto, 2018), in that it is the gap between the Q-value of a
 283 state-action pair and the value function of the state (which
 284 itself is equal to the Q-value of the *previous* step due to
 285 deterministic dynamics).

286 **Critical steps, per-step DPO, and advantage-weighted**
 287 **RL.** We can use advantages (Equation 3) to characterize
 288 critical steps. While the advantage values will always be
 289 non-positive by definition, steps that attain a higher advantage
 290 value than others are more critical to precisely execute.
 291 In contrast, steps that with very low advantage values are
 292 likely worse and must be unlearned. Our definition of the
 293 advantage function and this connection also implies that
 294 one can calculate advantages for each step in a response
 295 via additional Monte-Carlo rollouts starting from prefixes
 296 defined by partial solutions. These advantage estimates
 297 (Equation 3) can then be used for training the model by
 298 running advantage-weighted RL. However, in Theorem 5.1
 299 we formally show that DPO on per-step pairs, which contrasts
 300 positive and negative traces obtained via additional
 301 rollouts from policy $\tilde{\pi}$, on prefixes of a response sampled
 302 from π_{sft} is equivalent to advantage-weighted RL. A proof
 303 of Theorem 5.1 is in Appendix C. Note that unlike the standard
 304 reduction of DPO to the RL objective under *some*
 305 reward function (Rafailov et al., 2023; 2024), Theorem 5.1
 306 is stronger in that it identifies the value function induced by
 307 per-step DPO.
 308

309 **Theorem 5.1** (Equivalence of advantage-weighted RL and
 310 DPO with per-step pairs). *The optimal policy from Equation 1*
 311 *with $\mathcal{D}_{\pi_{\text{sft}}}^{\pm}$ given by $(\mathbf{x}, [\mathbf{y}_{1:i}, +\mathbf{y}_{i+1}], [\mathbf{y}_{1:i}, -\mathbf{y}_{i+1}])$*
 312 *where the positive and negative traces share prefix*
 313 *$\mathbf{y}_{1:i} \sim \pi_{\text{sft}}$, and $-\mathbf{y}_{i+1} \sim \pi_{\text{sft}}(\cdot | \mathbf{x}, \mathbf{y}_{1:i})$, $+\mathbf{y}_{i+1} \sim$*
 314 *$\sigma(A_{\tilde{\pi}}(\mathbf{x}, \mathbf{y}_{1:i}; \cdot) - A_{\tilde{\pi}}(\mathbf{x}, \mathbf{y}_{1:i}; -\mathbf{y}_{i+1}))$, is identical to the*
 315 *optima of the advantage-weighted RL objective:*

$$316 \quad \max_{\pi} \mathbb{E}_{\mathbf{x} \sim p_{\text{syn}}(\mathbf{x}), \mathbf{y} \sim \pi_{\text{sft}}(\cdot | \mathbf{x})} \left[\sum_{i=1}^L \log \pi(\mathbf{y}_i | \mathbf{x}, \mathbf{y}_{0:i-1}) \right. \\ 317 \quad \left. \cdot \exp(A_{\tilde{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i) / \beta) \right]. \quad (4) \\ 318 \\ 319 \\ 320$$

321 **Practical instantiation of DPO with per-step pairs.** Our
 322 practical implementation of per-step DPO is an approximation
 323 of the above scheme, with $\tilde{\pi}$ chosen to be a particular
 324 policy. Concretely, the practical implementation of per-step
 325 DPO sets $\tilde{\pi}$ to be the best-of-K policy, $\text{BoK}(\pi_{\text{sft}})$ where
 326 $K = 5$. There are two advantages for choosing a higher
 327 value of K : (i) estimating the advantage in Equation 3 with
 328 Monte-Carlo rollouts has lower variance; and (ii) $Q_{\text{BoK}(\pi_{\text{sft}})}$
 329

is a non-decreasing function in K for any state-action, which
 implies that the solution of advantage-weighted RL objective
 will only improve, in the neighborhood of the SFT
 policy π_{sft} that appears in the regularization term. We will
 next discuss scaling results for negative data, and then in
 Section 5.3 show how per-step credit assignment improves
 generalization and builds robustness to spurious correlations.

5.2. Scaling Results for Negative Data

Observe in Figure 5(a,b), that for both DeepSeek-Math-7B
 and LLama2-7B models, per-step DPO improves performance
 beyond the SFT policy and the performance continues to scale
 favorably as data size increases. In fact, also note that for
 any given size of \mathcal{D}_{syn} , per-step DPO also substantially
 improves over RFT (Figure 2) on both datasets, and overall,
while RFT improved effective data size of \mathcal{D}_{syn} by 2 \times , additionally training on negative data extends the performance improvement to 8 \times the size of \mathcal{D}_{syn} . Additionally, since per-step DPO estimates advantage of each step under the Best-of-5 policy, one might expect a saturation in the pass@5 performance of the per-step DPO solution. On the contrary, we find that pass@5 performance also improves consistently.

Choice of negative data has significant impact. In Figure 5(c) we plot negative data scaling laws where the choice of negative data (and thereby pairs for DPO in Equation 1) differs. Observe that standard pairing of positive and negative responses in $\mathcal{D}_{\pi_{\text{sft}}}^{\pm}$ for DPO (Rafailov et al., 2023) does not improve upon the SFT policy. As such, we needed to tune β in Equation 1 for DPO but could not fully avoid performance degradation. Our conceptual model explains this result: since contrasting arbitrary positives and negatives would result in an incorrect induced advantage function, training with DPO will exacerbate spurious correlations that maximize this induced advantage function (Saeidi et al., 2024; Pang et al., 2024; Xu et al., 2024). In fact, Pal et al. (2024) also find similar concerns with random pairing and instead pair positives and negatives that with highest edit distance, which leads to some improvement over standard DPO (Figure 2(c)) but still performs poorer than per-step DPO that accounts for credit.

Takeaways for scaling negative synthetic data

- Negative data can identify high-advantage (critical) steps in model-generated responses.
- We can construct negative data distribution that equates DPO to advantage-weighted RL. Negative data used in this way improves the sample efficiency of synthetic data by 8 \times .

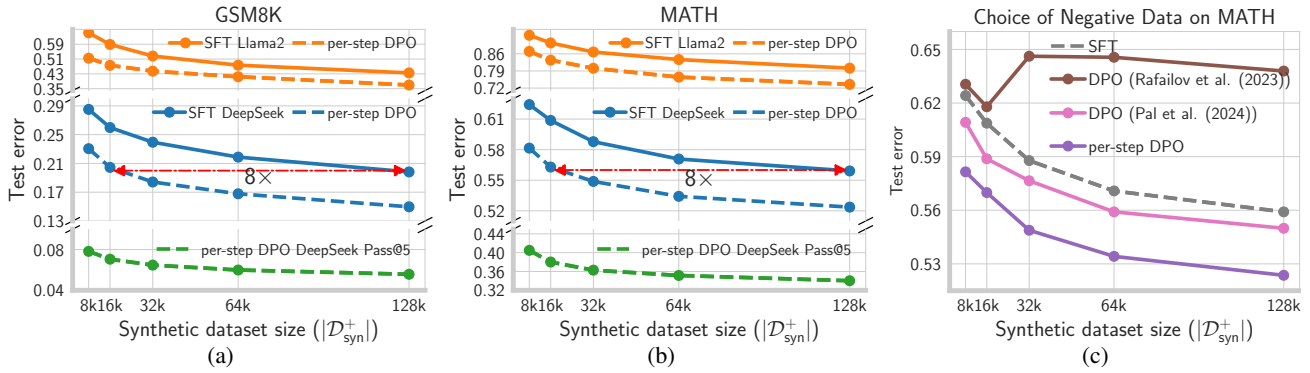
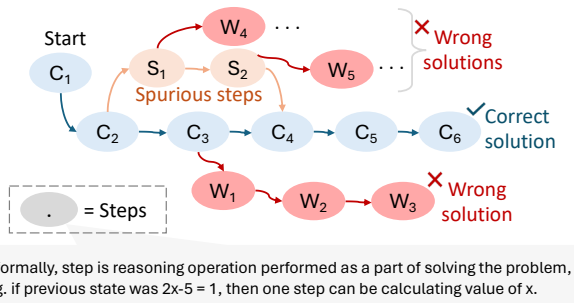


Figure 5: Negative data scaling laws: We evaluate algorithms that consume negative data as we scale \mathcal{D}_{syn} , and compare them with only positive training (SFT) on \mathcal{D}_{syn} . On GSM8K (a) and MATH (b), we observe an 8x gain from per-step DPO (Section 3) which aligns with our model of negative data that enables per-step credit assignment. In (c) we compare different negative data construction algorithms, and particularly note that naively pairing positives and negatives (Rafailov et al., 2023) leads to worse performance as we scale \mathcal{D}_{syn} .



Advantage function

- $A_{\tilde{\pi}}(C_2, C_2 \rightarrow S_1)$ is **low** (often leads to a wrong sol.)
- $A_{\tilde{\pi}}(C_2, C_2 \rightarrow C_3)$ is **midrange** (leads to correct and wrong sol.)
- $A_{\tilde{\pi}}(C_3, C_3 \rightarrow W_1)$ is **low** (leads to a wrong sol.)
- $A_{\tilde{\pi}}(C_3, C_3 \rightarrow C_4)$ is **high** (leads to the correct sol.)

Figure 6: Illustration of advantage estimation from negative data for identifying critical steps in synthetic model generations.

5.3. Why Does Credit Assignment Improve Model Generalization?

Our conceptual model illustrates that per-step DPO can perform credit assignment, and identify critical steps over irrelevant ones via advantage estimates. We saw that this improves test performance and scaling. Now, we attempt to understand why per-step credit assignment should improve generalization by understanding the generalization properties of advantage-weighted RL. We present two empirical studies below, and a formal theoretical guarantee combining these insights is shown in Appendix D.

1) Advantage-weighted RL de-emphasizes spurious steps and emphasizes critical steps. Our key insight is that spurious correlations emerge in monolithic SFT or RFT due to the well-known issue of causal confusion (De Haan et al., 2019) in imitation learning: by memorizing incorrect or irrelevant steps and associating them with the correctness

of the final answer, the model fails to generalize on novel problems, as we saw in Figure 4. We now explain how *online* model-specific interventions and advantage estimation would address this issue. Consider $\tilde{\pi} = \pi_{\text{sft}}$. As we show later, in under-trained models memorized steps are imperfectly cloned under π_{sft} , implying that while teacher-forcing loss is low for some spurious, memorized step y_s , sampling paths from π_{sft} , conditioned on $y_{1:s}$ is likely to generate incorrect responses. This means y_s attains a low advantage. On the other hand, for a correct step, *whp* estimated advantage is higher. Thus, training the model with advantage weighted RL would de-emphasize spurious steps and emphasize critical steps. Running per-step DPO on data generated by the RFT model that has overfit on spurious correlations improves accuracy by >6% (Figure 4). We visualize advantages in Appendix F. In Figure 8, we plot the average Q-value of a step for different negative data schemes, and note that only per-step DPO improves over SFT at each step, as expected based on the connection to advantage-weighted RL (Theorem 5.1). Standard DPO fails to improve performance since it has poor success rate at earlier (critical) steps.

2) Generalization depends on low advantage estimation error. The practical efficacy of algorithms that use negative data for credit assignment requires the advantage estimation error to be low with fewer rollouts from $\tilde{\pi}$. For discussion, consider $\tilde{\pi} = \pi_{\text{sft}}$. When the initial advantage of a spurious step is incorrectly over-estimated, negative data algorithms up-weight the likelihood further. This only leads to further memorization. Hence, most Monte-Carlo rollouts from π_{sft} would rely upon the memorized feature. Since the model generates the correct answer from the memorized feature, it would estimate higher $A_{\pi_{\text{sft}}}$, and this downward spiral of training with increasing weights on the spurious step leads to test-time model collapse. On the other hand, when $\tilde{\pi} = \text{BoK}(\pi_{\text{sft}})$ for a higher value of K , the Monte-Carlo advantage estimator has a lower variance (and error). This

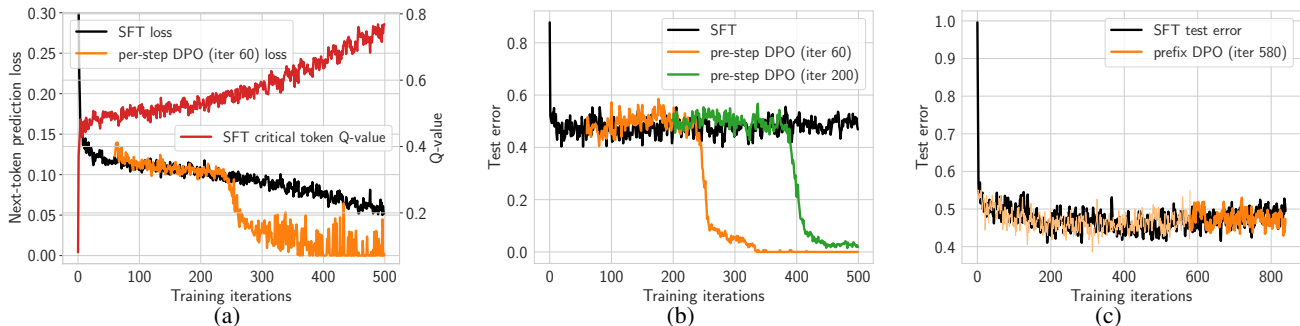


Figure 7: Didactic analysis on star graph: In (a) we plot the SFT loss and Q-value of the critical token (adjacent node) for SFT and per-step DPO (starting from iter 60). Indicative of memorization SFT loss decreases at a slow rate, matching the slow rate of increase in the Q-value. In contrast per-step DPO loss sharply decreases during training. In (b) we notice a corresponding phase transition in the test error of per-step DPO starting from different under-trained SFT checkpoints, which does not happen for an over-trained SFT checkpoint in (c).

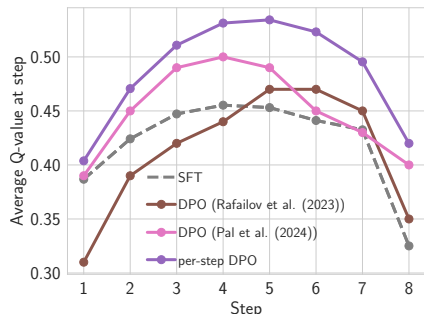


Figure 8: Per-step DPO improves Q-values at each step, standard DPO only improves at irrelevant steps.

discussion also justifies the choice of $K=5$, an intermediate value, in per-step DPO.

Didactic analysis. With the above insight, we now study the influence of π_{sft} on the generalization effects of per-step DPO. For our analysis, we consider a didactic star graph problem (Appendix G) from Bachmann & Nagarajan (2024), where given a graph in the shape of a star and a query (center/end node), the model is asked to output the full path between the start/end nodes. This task highlights the failure of SFT at planning problems (akin to math reasoning). They show that π_{sft} minimizes SFT loss by memorizing the “hard-to-predict” node adjacent to the center, and copying the rest from the input graph. It is clear that the failure stems from not being able to identify the critical adjacent token. We will show how credit assignment with negative data accurately upweights the critical token and unlearns the memorized token. To vary the choice of π_{sft} , we choose several intermediate checkpoints obtained during supervised finetuning for synthetic negative data generation. We consider three initializations: (1) an under-trained SFT model with a large training and test loss, and (2) an SFT model obtained by early-stopping based on a held-out validation set, where the validation loss is the lowest, and (3) an over-trained SFT checkpoint, with a low training but high validation loss.

(1) & (2): Training on negative data from an under-trained or early-stopped π_{sft} improves both training loss and test performance. As shown in Figure 7(a,b), we

find that when training with negative data from iteration 60 (under-trained π_{sft}) and iteration 200 (early-stopped π_{sft}), utilizing per-step DPO reduces the training loss very aggressively. These benefits translate to test losses and performance as well (Figure 7(b), orange and green). In contrast, supervised finetuning exhibits a nearly-flat test loss landscape, although the train loss reduces slowly. Upon a closer inspection, we find that training on positive data via SFT only tends to memorize the critical token in the training data using non-generalizable features, and hence, the resulting model does not generalize to novel problems. More training with SFT is unable to “unlearn” this spurious correlation and does not reduce the loss function. On the other hand, per-step DPO with negative data is able to unlearn this spurious feature and drives improvement, as evident by the drastic improvement on train and test.

(3) Training on negative data from an over-trained SFT initialization leads to model collapse. When training with negative data on an over-trained π_{sft} (iteration 580) in Figure 7(c), we observe that both SFT and per-step DPO exhibit identical test errors since training with more negative data exacerbates the model’s dependence on memorizing the critical token, which manifests in the form of lower train losses. This is also an example where Monte-Carlo samples from the over-trained checkpoint estimates a high advantage since Q-value is already high at iteration 500 (in (a)). Thus, when the SFT policy has sufficiently memorized the training data using a spurious feature, training further is unable to unlearn this dependence. Hence, in this regime, negative data leads to no improvement, capping performance at what was attained by fine-tuning on positive data.

Takeaways for generalization with negative data

Advantage-weighted RL unlearns spurious steps and improves generalization when: (i) advantage estimation error is low; and (ii) the model is under-trained enough that imperfectly cloned spurious steps have low advantage, which can then be estimated with negative data.

References

- Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep.*, 2019.
- Alemohammad, S., Casco-Rodriguez, J., Luzi, L., Humayun, A. I., Babaei, H., LeJeune, D., Siahkoohi, A., and Baraniuk, R. G. Self-consuming generative models go mad. *arXiv preprint arXiv:2307.01850*, 2023.
- Bachmann, G. and Nagarajan, V. The pitfalls of next-token prediction, 2024.
- Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., Fu, Z., et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.
- Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- Chen, Z., Deng, Y., Yuan, H., Ji, K., and Gu, Q. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.
- Cheng, P., Yang, Y., Li, J., Dai, Y., and Du, N. Adversarial preference optimization. *arXiv preprint arXiv:2311.08045*, 2023.
- Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- De Haan, P., Jayaraman, D., and Levine, S. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32, 2019.
- Dohmatob, E., Feng, Y., and Kempe, J. Model collapse demystified: The case of regression, 2024.
- Dong, G., Yuan, H., Lu, K., Li, C., Xue, M., Liu, D., Wang, W., Yuan, Z., Zhou, C., and Zhou, J. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492*, 2023.
- Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., Welleck, S., West, P., Bhagavatula, C., Le Bras, R., et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.
- Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.
- Gerstgrasser, M., Schaeffer, R., Dey, A., Rafailov, R., Sleight, H., Hughes, J., Korbak, T., Agrawal, R., Pai, D., Gromov, A., Roberts, D. A., Yang, D., Donoho, D. L., and Koyejo, S. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data, 2024.
- Hartmann, V., Suri, A., Bindschaedler, V., Evans, D., Tople, S., and West, R. Sok: Memorization in general-purpose large language models, 2023.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.
- Hong, J., Lee, N., and Thorne, J. Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*, 2024.
- Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sordoni, A., and Agarwal, R. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.
- Hwang, H., Kim, D., Kim, S., Ye, S., and Seo, M. Self-explore to avoid the pit: Improving the reasoning capabilities of language models with fine-grained rewards. *arXiv preprint arXiv:2404.10346*, 2024.
- Kääriäinen, M. Lower bounds for reductions. In *Atomic Learning Workshop*, 2006.

- 495 Kakade, S. and Langford, J. Approximately optimal approxi-
 496 mate reinforcement learning. In *International Conference*
 497 *on Machine Learning (ICML)*, volume 2, 2002.
- 498 Kang, K., Wallace, E., Tomlin, C., Kumar, A., and Levine,
 499 S. Unfamiliar finetuning examples control how language
 500 models hallucinate, 2024.
- 501 Kumar, A., Hong, J., Singh, A., and Levine, S. When
 502 Should We Prefer Offline Reinforcement Learning over
 503 Behavioral Cloning? *ICLR*, 2022.
- 504 Li, C., Wang, W., Hu, J., Wei, Y., Zheng, N., Hu, H.,
 505 Zhang, Z., and Peng, H. Common 7b language models
 506 already possess strong math capabilities. *arXiv preprint*
 507 *arXiv:2403.04706*, 2024.
- 508 Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker,
 509 B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and
 510 Cobbe, K. Let’s verify step by step, 2023.
- 511 Liu, H., Zaharia, M., and Abbeel, P. Exploration with
 512 principles for diverse ai supervision. *arXiv preprint*
 513 *arXiv:2310.08899*, 2023.
- 514 Liu, R., Wei, J., Liu, F., Si, C., Zhang, Y., Rao, J., Zheng, S.,
 515 Peng, D., Yang, D., Zhou, D., and Dai, A. M. Best prac-
 516 tices and lessons learned on synthetic data for language
 517 models, 2024.
- 518 Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X.,
 519 Lin, Q., Chen, S., and Zhang, D. Wizardmath: Empow-
 520 ering mathematical reasoning for large language models
 521 via reinforced evol-instruct, 2023.
- 522 McCoy, R. T., Yao, S., Friedman, D., Hardy, M., and Grif-
 523 fiths, T. L. Embers of autoregression: Understanding
 524 large language models through the problem they are
 525 trained to solve. *arXiv preprint arXiv:2309.13638*, 2023.
- 526 Momennejad, I., Hasanbeig, H., Vieira Frujeri, F., Sharma,
 527 H., Jojic, N., Palangi, H., Ness, R., and Larson, J. Eval-
 528 uating cognitive maps and planning in large language
 529 models with cogeval. *Advances in Neural Information*
 530 *Processing Systems*, 36, 2024.
- 531 Munos, R., Valko, M., Calandriello, D., Azar, M. G., Row-
 532 land, M., Guo, Z. D., Tang, Y., Geist, M., Mesnard, T.,
 533 Michi, A., et al. Nash learning from human feedback.
 534 *arXiv preprint arXiv:2312.00886*, 2023.
- 535 Pal, A., Karkhanis, D., Dooley, S., Roberts, M., Naidu, S.,
 536 and White, C. Smaug: Fixing failure modes of pref-
 537 erence optimisation with dpo-positive. *arXiv preprint*
 538 *arXiv:2402.13228*, 2024.
- 539 Pang, R. Y., Yuan, W., Cho, K., He, H., Sukhbaatar, S., and
 540 Weston, J. Iterative reasoning preference optimization.
 541 *arXiv preprint arXiv:2404.19733*, 2024.
- 542 Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advan-
 543 tage-weighted regression: Simple and scalable
 544 off-policy reinforcement learning. *arXiv preprint*
 545 *arXiv:1910.00177*, 2019.
- 546 Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning,
 547 C. D., and Finn, C. Direct preference optimization: Your
 548 language model is secretly a reward model. *arXiv preprint*
 549 *arXiv:2305.18290*, 2023.
- Rafailov, R., Hejna, J., Park, R., and Finn, C. From r to q^* :
 Your language model is secretly a q-function, 2024.
- Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lilli-
 crap, T., Alayrac, J.-b., Soricut, R., Lazaridou, A., Firat,
 O., Schrittwieser, J., et al. Gemini 1.5: Unlocking multi-
 modal understanding across millions of tokens of context.
arXiv preprint arXiv:2403.05530, 2024.
- Ross, S. and Bagnell, D. Efficient reductions for imita-
 tion learning. In *International Conference on Artificial*
Intelligence and Statistics (AISTATS), pp. 661–668, 2010.
- Saeidi, A., Verma, S., and Baral, C. Insights into align-
 ment: Evaluating dpo and its variants across multiple
 tasks. *arXiv preprint arXiv:2404.14723*, 2024.
- Seddik, M. E. A., Chen, S.-W., Hayou, S., Youssef, P., and
 Debbah, M. How bad is training on synthetic data? a
 statistical analysis of language model collapse, 2024.
- Sharma, A., Keh, S., Mitchell, E., Finn, C., Arora, K., and
 Kollar, T. A critical evaluation of ai feedback for aligning
 large language models, 2024.
- Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot,
 N., and Anderson, R. The curse of recursion: Training
 on generated data makes models forget. *arXiv preprint*
arXiv:2305.17493, 2023.
- Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil,
 P., Garcia, X., Liu, P. J., Harrison, J., Lee, J., Xu, K.,
 Parisi, A., Kumar, A., Alemi, A., Rizkowsky, A., Nova,
 A., Adlam, B., Bohnet, B., Elsayed, G., Sedghi, H., Mor-
 datch, I., Simpson, I., Gur, I., Snoek, J., Pennington, J.,
 Hron, J., Kenealy, K., Swersky, K., Mahajan, K., Culp,
 L., Xiao, L., Bileschi, M. L., Constant, N., Novak, R.,
 Liu, R., Warkentin, T., Qian, Y., Bansal, Y., Dyer, E.,
 Neyshabur, B., Sohl-Dickstein, J., and Fiedel, N. Beyond
 human data: Scaling self-training for problem-solving
 with language models, 2024.
- Sutton, R. S. and Barto, A. G. *Reinforcement learning: An*
introduction. The MIT Press, second edition, 2018.
- Swamy, G., Dann, C., Kidambi, R., Wu, Z. S., and Agarwal,
 A. A minimaximalist approach to reinforcement learning
 from human feedback. *arXiv preprint arXiv:2401.04056*,
 2024.

- 550 Tajwar, F., Singh, A., Sharma, A., Rafailov, R., Schneider,
551 J., Xie, T., Ermon, S., Finn, C., and Kumar, A. Preference
552 fine-tuning of llms should leverage suboptimal, on-policy
553 data, 2024.
- 554
555 Tirumala, K., Markosyan, A., Zettlemoyer, L., and Agha-
556 janyan, A. Memorization without overfitting: Analyzing
557 the training dynamics of large language models. *Ad-
558 vances in Neural Information Processing Systems*, 35:
559 38274–38290, 2022.
- 560
561 Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi,
562 A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P.,
563 Bhosale, S., et al. Llama 2: Open foundation and fine-
564 tuned chat models. *arXiv preprint arXiv:2307.09288*,
565 2023.
- 566
567 Villalobos, P., Sevilla, J., Heim, L., Besiroglu, T., Hobbhahn,
568 M., and Ho, A. Will we run out of data? an analysis of
569 the limits of scaling datasets in machine learning. *arXiv
570 preprint arXiv:2211.04325*, 2022.
- 571
572 Wang, P., Li, L., Shao, Z., Xu, R. X., Dai, D., Li, Y., Chen,
573 D., Wu, Y., and Sui, Z. Math-shepherd: Verify and
574 reinforce llms step-by-step without human annotations,
575 2024.
- 576
577 Wang, Y., Liu, Q., and Jin, C. Is rlhf more difficult than
578 standard rl? *arXiv preprint arXiv:2306.14111*, 2023.
- 579
580 Williams, R. J. and Zipser, D. A learning algorithm for con-
581 tinually running fully recurrent neural networks. *Neural
582 computation*, 1(2):270–280, 1989.
- 583
584 Wu, T., Zhu, B., Zhang, R., Wen, Z., Ramchandran, K., and
585 Jiao, J. Pairwise proximal policy optimization: Harness-
586 ing relative feedback for llm alignment. *arXiv preprint
587 arXiv:2310.00212*, 2023.
- 588
589 Wyllie, S., Shumailov, I., and Papernot, N. Fairness feed-
590 back loops: Training on synthetic data amplifies bias,
591 2024.
- 592
593 Xu, H., Sharaf, A., Chen, Y., Tan, W., Shen, L., Van Durme,
594 B., Murray, K., and Kim, Y. J. Contrastive preference
595 optimization: Pushing the boundaries of llm performance
596 in machine translation. *arXiv preprint arXiv:2401.08417*,
597 2024.
- 598
599 Yu, F., Gao, A., and Wang, B. Outcome-supervised verifiers
600 for planning in mathematical reasoning. *arXiv preprint
601 arXiv:2311.09724*, 2023.
- 602
603 Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok,
604 J. T., Li, Z., Weller, A., and Liu, W. Metamath: Boot-
strap your own mathematical questions for large language
models, 2024.
- Yuan, L., Cui, G., Wang, H., Ding, N., Wang, X., Deng, J.,
Shan, B., Chen, H., Xie, R., Lin, Y., et al. Advancing
llm reasoning generalists with preference trees. *arXiv
preprint arXiv:2404.02078*, 2024a.
- Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J.,
and Weston, J. Self-rewarding language models. *arXiv
preprint arXiv:2401.10020*, 2024b.
- Yuan, Z., Yuan, H., Li, C., Dong, G., Tan, C., and Zhou,
C. Scaling relationship on learning mathematical reason-
ing with large language models. *arXiv preprint
arXiv:2308.01825*, 2023.
- Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Boot-
strapping reasoning with reasoning. *Advances in Neural
Information Processing Systems*, 35:15476–15488, 2022.
- Zhang, B., Liu, Z., Cherry, C., and Firat, O. When scal-
ing meets llm finetuning: The effect of data, model and
finetuning method, 2024a.
- Zhang, R., Lin, L., Bai, Y., and Mei, S. Negative preference
optimization: From catastrophic collapse to effective un-
learning. *arXiv preprint arXiv:2404.05868*, 2024b.
- Zhao, Y., Khalman, M., Joshi, R., Narayan, S., Saleh, M.,
and Liu, P. J. Calibrating sequence likelihood improves
conditional language generation. In *The Eleventh Inter-
national Conference on Learning Representations*, 2022.

Appendices

A. Related Work

A standard procedure to finetune a pretrained LLM is teacher-forcing on expert data, *i.e.*, maximizing the likelihood of the next token given all previous tokens (Williams & Zipser, 1989; Brown et al., 2020). First, we discuss some failure modes of this procedure for math reasoning that positive or negative synthetic data can address.

Failure modes for supervised finetuning (SFT). First, since SFT induces an open-loop (Wu et al., 2023) next-token prediction loss, prediction errors on even a single token can snowball during inference, leading to poor performance on the prompts appearing in the data itself (Kääriäinen, 2006; Ross & Bagnell, 2010). Second, even when an LLM has perfectly cloned the SFT data, it is prone to memorize “hard to learn” tokens (Tirumala et al., 2022), especially in planning and lookahead tasks (McCoy et al., 2023; Momennejad et al., 2024), which is critical for math reasoning. This leads to poor generalization (Bachmann & Nagarajan, 2024; Dziri et al., 2024) and hallucination on new novel, test-time prompts (Kang et al., 2024). In this work, we study how synthetic data methods can address these failures via: (i) maximizing likelihood on positive data generated from both the SFT policy and a stronger teacher that enjoys improved coverage over new states, and (ii) preference optimization using the negative data generated from the SFT policy. **Positive synthetic data.** Learning theory dictates that the SFT policy trained on more SFT data (*e.g.*, 1.5M for DeepSeek-Math (Bi et al., 2024)) would have improved math reasoning capabilities. Thus, a common goal for generating synthetic data as close as possible to the SFT data (Li et al., 2024; Liu et al., 2023; 2024). That said, generating high quality math data can be challenging, since verification can often be hard. When synthetic data is verified by larger models (Sharma et al., 2024; Wang et al., 2024), recent works (Luo et al., 2023; Yu et al., 2024) observe scaling similar to finetuning LLMs on expert data (Zhang et al., 2024a; Yuan et al., 2023), while another work (Dong et al., 2023) notes the compositional gains from SFT data for code generation. Common sources of “good” synthetic data include responses from stronger teachers (Li et al., 2024; Lightman et al., 2023), or data generated by the SFT policy itself, in the framework of reinforced self-training (ReST) and STaR (Zelikman et al., 2022; Singh et al., 2024; Chen et al., 2024; Yuan et al., 2023). In our work, we study and compare the performance scaling with positive synthetic data from bigger models like GPT-4 and Gemini 1.5 Pro with self-generated positive data. We connect our findings to evidence showing “ease of learning” generalizable features on self-generated completions (Kang et al., 2024) which often prevents undesirable memorization (Tirumala et al., 2022). Finally, our work also sheds light on several concerns about training on synthetic positive data amplifying biases (Seddik et al., 2024; Wyllie et al., 2024), and leading to model collapse (Dohmatob et al., 2024; Gerstgrasser et al., 2024), especially due to overfitting on “spurious” intermediate steps. We conceptually explain this phenomenon and also discuss how negative model-generated responses can help identify and unlearn those spurious steps.

Benefits and nuances of negative synthetic data. While most works on synthetic data for math reasoning (Yu et al., 2024; Li et al., 2024; Liu et al., 2024; Yuan et al., 2023) focus on training on positive (correct) answers, our work also studies complementary gains from negative (incorrect) completions generated by the SFT policy (Hwang et al., 2024; Pal et al., 2024; Yuan et al., 2024b; Pang et al., 2024). To leverage sub-optimal negative data, we adopt the generic framework of offline preference optimization (Rafailov et al., 2023; Ethayarajh et al., 2024; Zhao et al., 2022), where a preference pair is constructed using correct and incorrect responses for the same problem (Pal et al., 2024). Despite numerous studies on preference data composition (Chen et al., 2024; Cheng et al., 2023; Tajwar et al., 2024; Chiang et al., 2023; Wang et al., 2023; Munos et al., 2023; Swamy et al., 2024), it remains unclear what is the best approach to pose a reasoning problem as a preference optimization problem. Randomly pairing correct and incorrect completions in a preference pair can lead to poor performance (Pang et al., 2024; Hong et al., 2024; Xu et al., 2024; Pal et al., 2024) due to objective mismatch (Tajwar et al., 2024; Zhang et al., 2024b) and requires auxiliary losses to perform well. Another option is to utilize negative data for training verifiers (Hosseini et al., 2024; Yu et al., 2023) but this line of work still only trains the policy using positive data. We introduce a conceptual model of negative data, where we understand how certain choices of negative data can assign per-step credits, which we then use to establish the equivalence of preference optimization to to advantage weighted RL. Self-explore method in Hwang et al. (2024) can be viewed as an special instance of our general framework. Another work exploiting per-step credits is Wang et al. (2024): through tree-based sampling they identify and use the reasoning subsequence that led to the most incorrect answers under the SFT policy for training a reward model. While this is indeed related, our conceptual model and analysis also aims to understand why assigning per-step credits can generalize better by unlearning spurious correlations, *e.g.*, when the credits are given by the Q-function of the “best-of-K” SFT policy.

B. Limitations and Broader Impact

While our work provides some results and conceptual models to understand the role of synthetic data for reasoning, there are still many open questions that need to be answered to fully understand its utility. While synthetic data from LLMs like Gemini and GPT-4 holds great potential, for more complex reasoning problems (more complicated than the datasets evaluated in our work), synthetic data generated from more capable models can contain errors and generating negative/positive data by referencing synthetic data answers can reinforce unwanted spurious correlations highlighted in our work. This means that other novel recipes for generating synthetic problems may be utilized in the future, and our analysis might need to be re-done with those novel recipes. That said, we believe that our insights about algorithmic behavior with synthetic data are still quite general and should transfer to these novel settings as well. Ultimately, we would want that training on synthetic data improves transfer and generalization abilities of the model in general reasoning scenarios, and to this end, an evaluation of transfer capabilities is an important avenue that future work should focus on.

Broader impact. Our work focuses purely on understanding the role of synthetic data in improving reasoning capabilities of LLMs. While excessive use of synthetic data can have unintended side effects upon deployment (e.g., fitting onto spurious correlations as we illustrate in Section 4) and advanced reasoning capabilities may have the potential to affect economy, human life, and society in both good and bad ways, we believe that these societal impacts are not unique or special to our work when compared to other works studying similar problems. In fact, with capabilities in foundation models improving day-by-day, future research and policy decisions would only benefit from more conceptual models to understand how algorithms operate and how data affects performance, which our work attempts to study.

C. Proof of Theorem 5.1

We first restate the theorem statement and then provide a proof for this below. Our main goal in this theorem is to show that training with per-step DPO is equivalent to running advantage-weighted RL shown in the theoretical result.

Theorem C.1 (Equivalence of advantage-weighted RL and DPO with per-step pairs). *The optimal policy from Equation 1 with $\mathcal{D}_{\pi_{\text{sft}}}^{\pm}$ given by $(\mathbf{x}, [\mathbf{y}_{1:i}, +\mathbf{y}_{i+1}], [\mathbf{y}_{1:i}, -\mathbf{y}_{i+1}])$ where the positive and negative traces share prefix $\mathbf{y}_{1:i} \sim \pi_{\text{sft}}$, and $-\mathbf{y}_{i+1} \sim \pi_{\text{sft}}(\cdot|\mathbf{x}, \mathbf{y}_{1:i})$, $+\mathbf{y}_{i+1} \sim \sigma(A_{\bar{\pi}}(\mathbf{x}, \mathbf{y}_{1:i}; \cdot) - A_{\bar{\pi}}(\mathbf{x}, \mathbf{y}_{1:i}; -\mathbf{y}_{i+1}))$, is identical to the optima of the advantage-weighted RL objective:*

$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim p_{\text{syn}}(\mathbf{x}), \mathbf{y} \sim \pi_{\text{sft}}(\cdot|\mathbf{x})} \left[\sum_{i=1}^L \log \pi(\mathbf{y}_i | \mathbf{x}, \mathbf{y}_{0:i-1}) \cdot \exp(A_{\bar{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i) / \beta) \right]. \quad (5)$$

Proof. To prove this statement, we make the following observation: DPO (Rafailov et al., 2023) is equivalent to optimizing a KL-divergence penalized expected reward objective in an induced Bradley-Terry model of preferences defined by the reward function. That is, for any reward function $r(\mathbf{x}, \mathbf{y})$ over contexts $\mathbf{x} \sim \mu$ and responses \mathbf{y} , the optimal solution to the following RL objective:

$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mu, \mathbf{y} \sim \pi(\cdot|\mathbf{x})} [r(\mathbf{x}, \mathbf{y})] - \beta D_{\text{KL}}(\pi(\cdot|\mathbf{x}) || \pi_{\text{sft}}(\cdot|\mathbf{x})), \quad (6)$$

is given by the following advantage-weighted optimal policy, $\pi^*(\cdot|\cdot)$:

$$\forall \mathbf{x}, \mathbf{y}, \pi^*(\mathbf{y}|\mathbf{x}) \propto \pi_{\text{sft}}(\mathbf{y}|\mathbf{x}) \cdot \exp\left(\frac{r(\mathbf{x}, \mathbf{y})}{\beta}\right), \quad (7)$$

and one can learn this optimal policy by running DPO on preference tuples $(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2)$ sampled by the Bradley-Terry model (Bradley & Terry, 1952) induced by the reward function r :

$$p(\mathbf{y}_1 \succcurlyeq \mathbf{y}_2 | \mathbf{x}) = \frac{\exp(r(\mathbf{x}, \mathbf{y}_1))}{\exp(r(\mathbf{x}, \mathbf{y}_1)) + \exp(r(\mathbf{x}, \mathbf{y}_2))}. \quad (8)$$

Given this background information, we know that the optimal advantage-weighted RL policy optimizing Equation 5 is given by:

$$\forall \mathbf{x}, \mathbf{y}_{0:i}, \pi(\mathbf{y}_i | \mathbf{x}, \mathbf{y}_{0:i-1}) \propto \pi_{\text{sft}}(\mathbf{y}_i | \mathbf{x}, \mathbf{y}_{0:i-1}) \cdot \exp\left(\frac{A_{\bar{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i)}{\beta}\right). \quad (9)$$

Combining Equation 9 with the equivalence between Equation 7 and the Bradley-Terry model (Equation 8), we obtain that, if preference pairs $(\mathbf{x}, [\mathbf{y}_{1:i}, +\mathbf{y}_{i+1}], [\mathbf{y}_{1:i}, -\mathbf{y}_{i+1}])$ were sampled from the SFT policy: $+\mathbf{y}_{i+1} \sim \pi_{\text{sft}}(\cdot | \mathbf{x}, \mathbf{y}_{0:i})$ and $-\mathbf{y}_{i+1} \sim \pi_{\text{sft}}(\cdot | \mathbf{x}, \mathbf{y}_{0:i})$, and labeled according to Equation 8 applied on advantage estimates, then we obtain the desired equivalence result. \square

D. Theory: Why Does Negative Data Improve Generalization?

We saw in Section 5.3 that collecting negative data from an appropriate SFT policy π_{sft} and an appropriate K , and training on this data improves generalization performance of the resulting model. In this section, building on the equivalence to advantage-weighted RL (Theorem 5.1), we attempt to formalize this observation into a performance guarantee. In particular, we show below that training on negative data implies that we are able to improve over the SFT policy, especially via the detection of critical steps, that attain high advantages, $A_{\hat{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i)$, that are otherwise not prioritized by training on positive data alone. Our theoretical result extends guarantees from the RL literature (Kumar et al., 2022) comparing RL with imitation learning to show that indeed the use of RL (and hence negative data) improves over imitation alone.

Notation and setup. Define the policy obtained after advantage-weighted RL training as π_{neg} . Concretely, $\pi_{\text{neg}}(\mathbf{y} | \mathbf{x})$ is given as:

$$\forall \mathbf{x}, \mathbf{y}_{0:j+1}, \pi_{\text{neg}}(\mathbf{y}_{j+1} | \mathbf{x}, \mathbf{y}_{0:j}) = \frac{1}{\hat{\mathbb{Z}}(\mathbf{x}, \mathbf{y}_{0:j})} \pi_{\text{sft}}(\mathbf{y}_{j+1} | \mathbf{x}, \mathbf{y}_{0:j}) \cdot \exp\left(\frac{\hat{A}_{\hat{\pi}}(\mathbf{x}, \mathbf{y}_{0:j}, \mathbf{y}_{j+1})}{\beta}\right), \quad (10)$$

where the normalization factor is given by $\mathbb{Z}(\mathbf{x}, \mathbf{y}_{0:j})$ for each of the per-step policy distributions. This normalization factor is a critical factor that will drive the core of the theoretical result. We also note that the normalization factor in Equation 10 is derived from *empirical* advantage estimates and not from the expected estimates for the advantage value. Following Agarwal et al. (2019); Kumar et al. (2022), we operate in a tabular setting with a discrete (but combinatorially-large and variable-length) action space of responses, and our proof follows Theorem 4.4 in Kumar et al. (2022).

Theorem D.1 (Utility of negative data over positive data.). *Let π_{neg} denote the policy obtained after advantage-weighted RL (Equation 10) under an empirical distribution $\hat{\mu}$ over prompts \mathbf{x} . Then the difference between the expected reward (i.e., task success rate), $J(\cdot)$, attained by π_{neg} and π_{sft} is lower-bounded as:*

$$J(\pi_{\text{neg}}) - J(\pi_{\text{sft}}) \geq \beta \cdot \mathbb{E}_{\mathbf{x}_i \sim \hat{\mu}, \mathbf{y}_{i,0:L} \sim \pi_{\text{neg}}(\cdot | \mathbf{x}_i)} \left[\sum_{j=1}^L \log \mathbb{Z}(\mathbf{x}_i, \mathbf{y}_{i,0:j}) \right] - (\text{overestimation in } \hat{A}_{\hat{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i)) + \frac{c_0}{\sqrt{|\mathcal{D}_{\text{syn}}|}},$$

where $\mathbb{Z}(\clubsuit, \circ)$ denotes the sum over exponentiated differences of the advantage and log likelihood values under π_{sft} for all possible candidate steps given a problem \clubsuit and a partial solution \circ . That is,

$$\mathbb{Z}(\clubsuit, \circ) := \sum_{\spadesuit \in \text{step candidates}} \exp\left(\frac{A_{\hat{\pi}}(\clubsuit, \circ; \spadesuit)}{\beta} + \log \pi_{\text{sft}}(\spadesuit | \clubsuit, \circ)\right),$$

c_0 is a constant depending upon the Rademacher complexity of the space of policies π_{neg} close to the SFT policy under the KL-divergence, $|\mathcal{D}_{\text{syn}}|$ denotes the size of synthetic training prompts.

Proof. To begin the proof, we recall that we are operating in a discrete action space of steps \mathbf{y}_i , although this space is exponentially large. Since we operate in discrete action spaces, we invoke Lemma 5 from Agarwal et al. (2019) for analyzing softmax policy gradient methods (this Lemma was extended by Lemma B.11 from Kumar et al. (2022) for comparing BC vs offline RL). Denote by $\hat{J}(\pi)$, the reward attained by policy π in expectation over the empirical distribution $\hat{\mu}$:

$$\hat{J}(\pi_{\text{neg}}) - \hat{J}(\pi_{\text{sft}}) := \mathbb{E}_{\mathbf{x} \sim \hat{\mu}} [\hat{V}^{\pi_{\text{neg}}}(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim \hat{\mu}} [\hat{V}^{\pi_{\text{sft}}}(\mathbf{x})] \geq \beta \mathbb{E}_{\mathbf{x} \sim \hat{\mu}} [\log \hat{\mathbb{Z}}(\mathbf{x})]. \quad (11)$$

We utilize the performance difference lemma (Kakade & Langford, 2002) on the MDP induced by the set of initial problems

in the training distribution $\hat{\mu}$, and the model induced deterministic dynamics distribution:

$$\begin{aligned}
 \hat{J}(\pi_{\text{neg}}) - \hat{J}(\pi_{\text{sft}}) &= \sum_{j=1}^L \mathbb{E}_{\mathbf{x} \sim \hat{\mu}, \mathbf{y}_{0:j-1} \sim \pi_{\text{neg}}(\cdot|\mathbf{x})} \left[\sum_{\mathbf{y}_j} \pi_{\text{neg}}(\mathbf{y}_j|\mathbf{x}, \mathbf{y}_{0:j-1}) \hat{A}_{\tilde{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i) \right] \\
 &= \sum_{j=1}^L \mathbb{E}_{\mathbf{x} \sim \hat{\mu}, \mathbf{y}_{0:j-1} \sim \pi_{\text{neg}}(\cdot|\mathbf{x})} \left[\sum_{\mathbf{y}_j} \pi_{\text{neg}}(\mathbf{y}_j|\mathbf{x}, \mathbf{y}_{0:j-1}) \log \frac{\pi_{\text{neg}}(\mathbf{y}_j|\mathbf{x}, \mathbf{y}_{0:j-1}) \cdot \hat{\mathbb{Z}}(\mathbf{x}, \mathbf{y}_{0:j})}{\pi_{\text{sft}}(\mathbf{y}_j|\mathbf{x}, \mathbf{y}_{0:j-1})} \right] \\
 &= \beta \cdot \sum_{j=1}^L \mathbb{E}_{\mathbf{x} \sim \hat{\mu}, \mathbf{y}_{0:j-1} \sim \pi_{\text{neg}}(\cdot|\mathbf{x})} \left[D_{\text{KL}}(\pi_{\text{neg}}(\cdot|\mathbf{x}, \mathbf{y}_{0:j-1}), \pi_{\text{sft}}(\cdot|\mathbf{x}, \mathbf{y}_{0:j-1})) + \log \hat{\mathbb{Z}}(\mathbf{x}, \mathbf{y}_{0:j}) \right] \\
 &\geq \beta \cdot \sum_{j=1}^L \mathbb{E}_{\mathbf{x} \sim \hat{\mu}, \mathbf{y}_{0:j-1} \sim \pi_{\text{neg}}(\cdot|\mathbf{x})} \left[\log \hat{\mathbb{Z}}(\mathbf{x}, \mathbf{y}_{0:j}) \right]. \\
 &= \beta \cdot \mathbb{E}_{\mathbf{x} \sim \hat{\mu}, \mathbf{y}_{i,0:L} \sim \pi_{\text{neg}}(\cdot|\mathbf{x})} \left[\sum_{j=1}^L \log \mathbb{Z}(\mathbf{x}, \mathbf{y}_{0:j}) \right].
 \end{aligned}$$

Now, we can prove the desired result by accounting for the gap in the success rate between the actual distribution over $\mathbf{x} \sim \mu$ and the empirical distribution induced by problems in the dataset $\hat{\mu}$:

$$\begin{aligned}
 J(\pi_{\text{neg}}) - J(\pi_{\text{sft}}) &\geq \underbrace{J(\pi_{\text{neg}}) - \hat{J}(\pi_{\text{neg}})}_{(a)} + \underbrace{\hat{J}(\pi_{\text{neg}}) - \hat{J}(\pi_{\text{sft}})}_{(b)} - \underbrace{J(\pi_{\text{sft}}) - \hat{J}(\pi_{\text{sft}})}_{(c)} \\
 &\geq \beta \cdot \mathbb{E}_{\mathbf{x} \sim \hat{\mu}, \mathbf{y}_{i,0:L} \sim \pi_{\text{neg}}(\cdot|\mathbf{x})} \left[\sum_{j=1}^L \log \hat{\mathbb{Z}}(\mathbf{x}, \mathbf{y}_{0:j}) \right] - \frac{c_0}{\sqrt{|\mathcal{D}_{\text{syn}}|}} \\
 &\geq \beta \cdot \mathbb{E}_{\mathbf{x} \sim \hat{\mu}, \mathbf{y}_{i,0:L} \sim \pi_{\text{neg}}(\cdot|\mathbf{x})} \left[\sum_{j=1}^L \log \mathbb{Z}(\mathbf{x}, \mathbf{y}_{0:j}) \right] - \frac{c_0}{\sqrt{|\mathcal{D}_{\text{syn}}|}} + \Delta,
 \end{aligned}$$

where c_0 is a constant that depends on the Rademacher complexity of the function class of policies π_{neg} (coming from a uniform bound that we invoke to bound term (a), since π_{neg} depends on the dataset samples), and this term arises since the empirical distribution over prompts is not the same as the true population. This concentration term decays as the size of the synthetic data (number of problems) are scaled up. The term Δ denotes the overestimation error between the estimated advantages $\hat{A}_{\tilde{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i)$ and the true advantages $A_{\tilde{\pi}}(\mathbf{x}, \mathbf{y}_{0:i-1}, \mathbf{y}_i)$, in expectation under the distribution of the learned policy. The estimation error Δ depends on π_{sft} and the value of K used if the rollout policy $\tilde{\pi}$ corresponds to the $\text{BoK}(\pi_{\text{sft}})$ policy. This proves the theorem. \square

Interpretation & perspectives. Also note that the improvement in performance between π_{neg} and π_{sft} depends on the advantage estimate: if the advantage estimates are high, then this term is large, meaning that the more the fraction of high-advantage critical states, the higher the improvement. In addition, the bound also says that if the over-estimation Δ in the advantage estimate is large, the performance improvement is small. This is perhaps expected: consider the scenario when the $\text{BoK}(\pi_{\text{sft}})$ policy is used to collect data, for a large K . In this scenario, the divergence between the empirical advantage estimate $\hat{A}_{\tilde{\pi}}$ and the expected estimate $A_{\tilde{\pi}}$ is likely large. In the worst case, the estimate $\hat{A}_{\tilde{\pi}}$ can arbitrarily overestimate $A_{\tilde{\pi}}$, as it would take on a high value even if there is just *one* sequence among the K rollouts that successfully solves the problem. For example, a spurious step may be labeled incorrectly as critical in this case and training on negative data may not improve (consistent with running per-step DPO on an over-trained SFT checkpoint in Figure 7). On the other hand, if advantages are more accurate, training on negative data should improve performance.

E. Synthetic Data Generation

Prompt used for GSM8K/MATH synthetic data (Li et al., 2024)

Please act as a professional math teacher. Your goal is to create high quality math problems to help students learn math. You will be given a math question. Please create a new question based on the Given Question and following instructions.

To achieve the goal, you have one job.

Please generate a similar but new question according to the Given Question.

You have four principles to do this. # Ensure the new question only asks for one thing, be reasonable, be based on the Given Question, and can be answered with only a number(float or integer). For example, DO NOT ask, ‘what is the amount of A, B and C?’.

Ensure the new question is in line with common sense of life. For example, the amount someone has or pays must be a positive number, and the number of people must be an integer.

Ensure your student can answer the new question without the given question. If you want to use some numbers, conditions or background in the given question, please restate them to ensure no information is omitted in your new question.

You only need to create the new question. Please DO NOT solve it.

Given Question: <insert question from original dataset here>

Your output should be in the following format:

CREATED QUESTION: <your created question>

For GSM8K, we replace the phrase “Your goal is to create high quality math problems to help students learn math.” with “Your goal is to create high quality math *word* problems to help students learn math.”, as we found this to produce problems that were closer to GSM-style problems.

To generate the synthetic data, we used OpenAI credits worth approximately 3000 US dollars.

F. Negative Data Identifies Spurious Steps with Advantage Estimates

In this section, we present several examples from the GSM8K and MATH synthetic datasets, and highlight the steps that were deemed to be low or high advantage based on the estimated advantage value. The darker the intensity of the color, the lower the advantage of that step.

Spurious step identified by negative data

Question: Ken created a care package to send to his brother, who was away at boarding school. Ken placed a box on a scale, and then he poured into the box enough jelly beans to bring the weight to 2 pounds. Then, he added enough brownies to cause the weight to triple. Next, he added another 2 pounds of jelly beans. And finally, he added enough gummy worms to double the weight once again. What was the final weight of the box of goodies, in pounds?

Positive response with identified spurious step:

When the weight was originally on the scale, the box weighed 0 pounds. After Ken added the jelly beans to the box, its weight was 2 pounds - 0 pounds = $\langle 2-0=2 \rangle$ 2 pounds. Then Ken added the brownies to the box, which caused the box to weigh 2 pounds * 3 = $\langle 2*3=6 \rangle$ 6 pounds. The next step was to add 2 pounds of jelly beans, which caused the box to weigh 6 pounds + 2 pounds = $\langle 6+2=8 \rangle$ 8 pounds. First, we need to figure out what the weight of the box was when Ken put the jelly beans in it, 2 pounds of jelly beans was enough to triple the weight, so the box was at 2 pounds / 3 = $\langle 2/3=.6666666666666667 \rangle$.6666666666666667 pounds. Finally, Ken added the gummy worms, causing the box to weigh 8 pounds * 2 = $\langle 8*2=16 \rangle$ 16 pounds. The answer is 16

Spurious step identified by negative data

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?

Positive response with identified spurious step:

Betty's parents gave her 15, so her grandparents gave her $2 * 15 = \ll 15 * 2 = 30 \gg 30$. In total, Betty received $30 + 15 = \ll 30 + 15 = 45 \gg 45$. The total amount of money Betty needs is $100 * 2 = \ll 100 * 2 = 200 \gg 200$. Betty needs $100 / 2 = \ll 100 / 2 = 50 \gg 50$ in total. Betty still needs to save $50 - 45 = \ll 50 - 45 = 5 \gg 5$. The answer is 5

Spurious step identified by negative data

Question: To make pizza, together with other ingredients, Kimber needs 10 cups of water, 16 cups of flour, and 1/2 times as many teaspoons of salt as the number of cups of flour. Calculate the combined total number of cups of water, flour, and teaspoons of salt that she needs to make the pizza.

Positive response with identified spurious step:

Since 1 cup is equal to 16 teaspoons, then Kimber needs $128 / 16 = \ll 128 / 16 = 8 \gg 8$ cups of salt. So in total, she needs $10 + 16 + 8 = \ll 10 + 16 + 8 = 34 \gg 34$ cups of water, flour, and salt. In cups, the amount of salt is $128 / 6 = \ll 128 / 6 = 21.33 \gg 21.33$ cups. The answer is 34

Spurious step identified by negative data

Question: Eight friends compete in a pie eating contest. The chart displays the number of pies each person consumed. Emily ate the most pies, and Mike ate the least. How many more pies did Emily eat compared to Mike?

```
[asy]\ndefaultpen(linewidth(1pt)+fontsize(10pt));
\ncpair[] yaxis = new pair[10];\nfor( int i = 0 ; i < 10 ; ++i )
{\n\nnyaxis[i] = (0,i);\n\ndraw(yaxis[i]--yaxis[i]+(17,0));\n}
\ndraw((0,0)--(0,9));\ndraw((17,9)--(17,0));
\n\nfill((1,0)--(1,5)--(2,5)--(2,0)--cycle, grey);
\nfill((3,0)--(3,8)--(4,8)--(4,0)--cycle, grey);
\nfill((5,0)--(5,3)--(6,3)--(6,0)--cycle, grey);
\n\nlabel("0",yaxis[0],W);
\nlabel("1",yaxis[1],W);\nlabel("2",yaxis[2],W);
\nlabel("3",yaxis[3],W);\nlabel("4",yaxis[4],W);
\nlabel("5",yaxis[5],W);\nlabel("6",yaxis[6],W);
\nlabel("7",yaxis[7],W);\nlabel("8",yaxis[8],W);
\nlabel("Friends/Participants", (8.5,0),S);
\nlabel("Results of a Pie Eating Contest", (8.5,9),N);
\nlabel(rotate(90)*"$\n$ of Pies Eaten", (-1,4.5),W);[/asy]
```

Positive response with identified spurious step:

To solve this problem, we need to identify the number of pies Emily and Mike consumed, and then find the difference between their consumption. From the chart provided: - Emily: Consumed 8 pies. - Mike: Consumed 1 pie. To determine how many more pies Emily ate compared to Mike, we first identify how many pies each person ate. - Emily ate 5 pies. - Mike ate 1 pie. Now, calculating the difference between Emily's and Mike's consumption: [Difference = Emily's consumption - Mike's consumption = $8 - 1 = 7$.] Therefore, Emily ate 7 more pies than Mike. The answer is 7

Spurious step identified by negative data

Question: When the base-16 number $1AB3_{16}$ is written in base 2, how many base-2 digits (bits) does it have?

Positive response with identified spurious step:

To solve this problem, we first need to convert the base-16 number $1AB3_{16}$ to base-2. Step 1: Convert each digit of the number from base-16 to base-2. The number $1AB3_{16}$ consists of the digits 1, A, B, and 3. Step 2: Convert each digit individually to base-2 representation. $1_{16} = 0001_2$, $A_{16} = 1010_2$, $B_{16} = 1011_2$, $3_{16} = 0011_2$. Step 3: Combine the base-2 representations of each digit. Concatenating the individual base-2 representations, we get: $1AB3_{16} = 0001101010110011_2$. Step 4: Count the total number of digits in the base-2 representation. The base-2 representation of $1AB3_{16}$ is: 0001101010110011_2 . The base-16 number ($1AB3_{16}$) can be written in base 10 (decimal) as: $1 \times 16^3 + A \times 16^2 + B \times 16 + 3$ where (A) and (B) are decimal digits. This sequence of digits has a total of 16 digits. The answer is boxed{16}

G. Details on Star Graph Problem

The star graph problem we study is borrowed from Bachmann & Nagarajan (2024), where given a graph in the shape of a star and a query (center/end node pair), the model is asked to output the full path between the start/end nodes.

Goal. Bachmann & Nagarajan (2024) show that π_{sft} minimizes SFT loss by memorizing the “hard-to-predict” node adjacent to the center, and copying the rest of the path from the input graph. This task highlights the failure of SFT at planning problems (akin to math reasoning). Thus, we use this as a case study to understand:

- when accurate advantage estimation is possible with few negative samples from the π_{sft} model.
- whether there are generalization benefits of advantage-weighted RL when advantage estimates are accurate
- when advantage-weighted RL can unlearn the memorized feature that causes π_{sft} to fail.

SFT dataset. The data we use for supervised fine-tuning consists of 30000 of random star graphs (see examples below) where each graph has a centre node with out degree 2. Hence, there are two paths that originate from the centre node. Each path from the center to one of the end nodes is of length 4. Each node on the path is denoted with a randomly sampled number from 0 to 20. For example, in the sample “8,3|3,10|14,13|10,11|7,14|8,17|8,13=8,17,14,13”. The graph is given by the adjacency list “8,3|3,10|14,13|10,11|7,14|8,17|8,13”, the query is denoted by “8,13”, and the correct path is given by “8,17,14,13”.

Test-time inference from the model. At test time, the input to the LLM is only the graph and the query: “8,3|3,10|14,13|10,11|7,14|8,17|8,13=” and the model is expected to generate the full path from start node 8 to end node 13. When evaluating the test performance of an LLM, we calculate 0/1 accuracy averaged over 1000 test star graphs (that are different from train star graphs). The accuracy on a sample is 1 when the LLM accurately predicts all nodes in the graph.

Failure models of the SFT model, π_{sft} . A model with perfect accuracy (0 error) would be the one that has accurately learned the correct feature of backtracking the path from the end node to the start node, and then producing it in reverse. This computation is precisely what makes the adjacent token “hard-to-fit”. On the other hand, if the LLM minimizes next-token prediction loss during SFT by instead memorizing the hard-to-fit adjacent token by overfitting on the random input graph instance, at test time the accuracy would be zero. An intermediate solution that SFT model instead learns is to output a path that is adjacent to the node. At training time, it only needs to memorize which of the two possible paths to predict. Note that even this solution does not require the model to backtrack, and is thus easier to quickly learn with a few samples. This would quickly minimize the loss on all but the adjacent node, which the model memorizes as training progresses. On the test set, this model would then have 50% test accuracy. Note, that as we increase the size of the graph or the node vocabulary size it becomes easier for the model to overfit on the hard to predict adjacent token given random combinations of the input graph. Thus, we choose the vocabulary size to be 20, which is higher than what is needed to represent any input graph of this size.

Below we provide examples from degree two, path length 4, node 20 problem, where

Examples of 20 node path length 4 star graph problem

Example 1: 8, 3|3, 10|14, 13|10, 1|17, 14|8, 17/8, 13=8, 17, 14, 13

Example 2: 14, 16|8, 10|9, 5|3, 14|9, 3|5, 8/9, 16=9, 3, 14, 16

Example 3: 14, 1|10, 4|9, 7|10, 17|4, 9|17, 14/10, 7=10, 4, 9, 7

Example 4: 19, 8|7, 18|14, 15|15, 7|14, 19|8, 10/14, 10=14, 19, 8, 10

Example 5: 1, 6|10, 1|6, 12|10, 17|17, 18|18, 5/10, 12=10, 1, 6, 12

SFT Training details. We finetune a pretrained GPT-2 model with 125 million parameters. We train with a batch size of 128, Adam without any weight decay, and a constant learning rate of $1e - 5$.

Advantage estimation and per-step DPO training equivalent to advantage-weighted RL. For a sample from π_{sft} , we estimate the advantage of each step by sampling 5 rollouts conditioned on the subsequence up till the step. We then pair subsequences with shared prefix, $\mathbf{y}_{1:i}$ differing in the last step $+\mathbf{y}_{i+1}$ vs. $-\mathbf{y}_{i+1}$, where the former is the one with the highest estimated advantage and the latter is the one with the lowest estimated advantage. Note that this preference pair construction, closely approximates the preference pair distribution in Theorem 5.1, which would imply that the DPO objective being optimized closely approximates advantage weighted RL in Equation 4.

Given these pairs for a batch of star graph problems in SFT data, we update the model with a single gradient step on the DPO objective in Equation 1. In the next iteration, advantage is estimated and pairs are constructed on a fresh batch of star graphs. We set $\beta = 0.1$ in the DPO objective and use the same batch size (one preference pair per star graph). Starting from an SFT checkpoint we train in the above manner for at least 200 iterations. The SFT model is trained for over 600 iterations.

H. Implementation Details

Datasets and pretrained LLMs. We run all our experiments on GSM8K and MATH datasets. Each dataset has about 7.5k training examples. The GSM8K has about 1.3k and MATH has 5k test examples. We conduct experiments with DeepSeek-Math-7B pretrained LLM and LLaMa2-7B, both of which have pretrained weights publicly available on Huggingface.

Details for SFT/RFT training. For our positive data scaling results, the SFT model is trained for 5 epochs with a learning rate of $1e - 5$, and a batch size of 64 for all sizes of \mathcal{D}_{syn} . We use a holdout validation set to choose the checkpoint and report the performance of the best performing checkpoint across the five epochs. To generate RFT data we only train the SFT model for 2 epochs (under-trained checkpoint). For each question we sample $M = 100$ times, with a temperature of 0.7 and following Yuan et al. (2024a) we retain at most 4 most diverse (based on edit distance) and correct completions. This is for our results in Figure 2(a,b). For Figure 2(c), we sample more than 4 correct solutions and keep sampling responses until we have a dataset of size 10k, 20k, ..., 50k, when questions are given by the \mathcal{D}_{syn} of size 8k and 16k. For our experiment on the RFT dataset with purposely inserted spurious steps, as we describe in the main paper, we obtain spurious steps by computing which intermediate steps in a negative response lead to most incorrect solutions and randomly insert this in between reasoning steps for a positive solution in RFT data. See examples below.

Details for per-step DPO training. Training data for DPO is generated in the procedure outlined in (Hwang et al., 2024). The value of K is 5 for the BoK(π_{sft}) policy used to estimate the advantage, and a single rollout from the best-of-K policy is used to estimate the advantage of each step in “rejected” response, where the “chosen” and “rejected” pairs for each question are generated from the RFT data, using the method in (Pal et al., 2024). We train per-step DPO with $\beta = 0.3$ for all \mathcal{D}_{syn} from GSM8K, and $\beta = 0.1$ for all \mathcal{D}_{syn} from MATH. We train for three epochs, with a learning rate of $1e - 6$ and a batch size of 64 prompts. We use the rmsprop optimizer and a cosine learning rate decay with a warmup ratio of 0.1.

Overall, to replicate our experiments, we need approximately 1000 hours of compute on H100 clusters with 8 GPUs.

RFT data with spurious steps highlighted

Prompt:

Karen's work tote bag is twice the weight of her husband Kevin's briefcase when the briefcase is empty. When Kevin puts his laptop and work papers in his briefcase, it is twice the weight of Karen's tote. Kevin's work papers are a sixth of the weight of the contents of his full briefcase. If Karen's tote weighs 8 pounds, how many more pounds does Kevin's laptop weigh than Karen's tote?

Response:

If Karen's tote weighs 8 pounds, and it is twice the weight of Kevin's empty briefcase, then Kevin's empty briefcase weighs $8/2 = \ll 8/2=4 \gg 4$ pounds. **In words, "Karen's tote weighs 8 pounds" is "T = 8", so thus Keven is 2 pounds.** When Kevin puts his laptop and work papers in his briefcase, it is twice the weight of Karen's tote at 8 pounds, so the briefcase now weighs $8+4 = \ll 8+4=12 \gg 12$ pounds. Kevin's work papers are a sixth of the weight of the contents of his full briefcase, which weighs 12 pounds, so the work papers weigh $12/6 = \ll 12/6=2 \gg 2$ pounds. If Kevin's laptop and work papers weigh 12 pounds in total, and the work papers weigh 2 pounds, then the laptop weighs $12-2 = \ll 12-2=10 \gg 10$ pounds. Thus, Kevin's laptop is $10-8 = \ll 10-8=2 \gg 2$ pounds heavier than Karen's tote. The answer is 2.

RFT data with spurious steps highlighted

Prompt:

Carl has a jar full of marbles. He takes out 12 marbles to play a game with, but he accidentally drops them and 1/2 the marbles get lost. So Carl takes out 10 more marbles. While he is playing his game his mother comes home with another bag of marbles for him, which has 25 marbles in it. If Carl can't find his lost marbles, then how many marbles is he going to put in the jar after he plays his game, from both his original marbles and the new ones?

Response:

First find out how many marbles Carl loses: $12 \text{ marbles} * 1/2 = \ll 12*1/2=6 \gg 6$ marbles Then add that number to the number of marbles Carl takes out of the jar to play his game to find out how many total marbles he takes out: $6 \text{ marbles} + 10 \text{ marbles} = \ll 6+10=16 \gg 16$ marble. **Now add that number to the number of marbles in the jar to find out how many marbles are left: $16 \text{ marbles} + 12 \text{ marbles} = \ll 16+12=28 \gg 28$ marbles.** Then add in the marbles Carl gets from his mother to find out how many marbles he has in total: $16 \text{ marbles} + 25 \text{ marbles} = \ll 16+25=41 \gg 41$ marbles. The answer is 41.