The First Few Tokens Are All You Need: *Unsupervised* Rejection-Free Sampling Fine-Tuning for Reasoning Models

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated impressive capabilities across various natural language processing tasks due to large-scale pre-training and extensive instruction fine-tuning. However, enhancing their reasoning abilities remains a significant challenge, often requiring supervised fine-tuning with extensive labeled datasets, which is resource-intensive. In this paper, we introduce a simple yet effective unsupervised fine-tuning method that significantly improves the reasoning performance of LLMs using only prefix substrings as minimal guidance. Our approach leverages the inherent reasoning structures within pretrained models to facilitate reasoning without the need for annotated data. We find that different reasoning trajectories for the same question tend to share common prefixes, a phenomenon we term **Prefix Self-Consistency**. By training the model on these prefixes, we enhance its reasoning capabilities efficiently. Experiments across various training corpora show that our method outperforms vanilla full-token fine-tuning and achieves performance comparable to supervised approaches like Rejection Sampling Fine-Tuning (RFT), while requiring significantly less training and inference time. This demonstrates that minimal unsupervised fine-tuning can substantially enhance the reasoning capabilities of LLMs, opening new avenues for efficient and accessible model improvement.

1 Introduction

Large language models (LLMs) have made significant advancements in numerous natural language processing (NLP) tasks, primarily due to large-scale pre-training combined with extensive instruction fine-tuning datasets (Longpre et al., 2023; Touvron et al., 2023). Despite these breakthroughs, equipping LLMs with complex reasoning capabilities remains a formidable challenge.

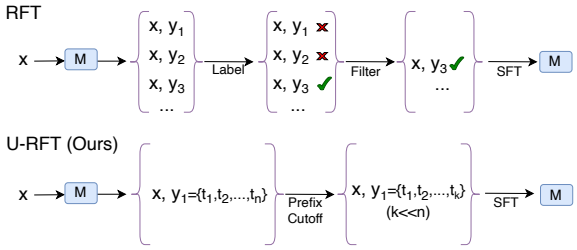

Figure 1: Rejection Sampling Fine-tuning (RFT) involves generating multiple responses to a given question and then applying posterior filtering to discard trajectories (e.g., y_1 and y_2) that lead to incorrect answers. Finally, the data pair (x, y_3), where y_3 is the correct response, is used for final training. In contrast, the proposed U-RFT method requires only a single sample and focuses on training with the initial tokens of the response, eliminating the need for extensive posterior filtering and rejection sampling.

Recent studies (Yang et al., 2024; OpenAI, 2023; Dubey et al., 2024; Guo et al., 2025) have demonstrated remarkable reasoning abilities in LLMs. One line of research focuses on enhancing reasoning performance through various prompting techniques (Wei et al., 2022a; Yao et al., 2023) or by ensembling and reranking multiple inference paths, such as verification (Cobbe et al., 2021; Uesato et al., 2022) and self-consistency (Wang et al., 2023b). However, employing in-context learning (ICL) and generating multiple inference paths is computationally expensive and unsuitable for online deployment (Muennighoff et al., 2025). Moreover, these methods do not fundamentally improve the model's inherent reasoning capabilities but rather guide it using external cues.

Another direction explores leveraging the LLMs' ability to generate solutions and verify their correctness to produce guidance-oriented data for training, exemplified by methods like ReST (Zelikman et al., 2022), Rejection Sampling Fine-Tuning (RFT) (Yuan et al., 2023), and Self-Taught Reasoner (STaR) (Zelikman et al., 2022).

V-STaR (Hosseini et al., 2024) further combines correct and incorrect trajectories using a verification and self-improvement paradigm. While these approaches have advantages, they are time-consuming, computationally expensive, and constrained by the availability of high-quality labeled data. They are also limited to improving weaker base models and assume that correct answer trajectories can be easily obtained through multiple sampling attempts. For extremely difficult questions, the model may require hundreds of inference attempts to find a correct path. Additionally, these methods rely on label-based posterior filtering, which fails when there are no pre-specified correct answers. This limitation hinders the acquisition of high-quality responses for challenging questions, even if the questions themselves are readily available.

Some works, such as s1 (Muennighoff et al., 2025) and LIMO (Ye et al., 2025), demonstrate that a small subset of curated datasets can unlock strong generalization capabilities in LLMs for reasoning tasks. However, these approaches heavily rely on external strong model distillation or distant supervision to construct high-quality answers for difficult questions. These challenges restrict further improvements in the model's reasoning abilities. This raises a critical question: How can we design an efficient and scalable method to enhance the reasoning capabilities of LLMs without relying on external supervision?

In this paper, we investigate an unsupervised sampling setup, also referred to as rejection-free sampling. This setup emphasizes collecting and training on diverse and challenging questions rather than scaling up the number of self-training samples, which can be time-consuming. We propose a novel unsupervised fine-tuning method that enhances the reasoning abilities of LLMs using only prefix substrings as minimal guidance.

Our approach is based on the observation that LLMs have latent reasoning structures acquired during pre-training on large corpora (Brown et al., 2020) and can be aligned using only a small subset of high-quality data (Muennighoff et al., 2025; Ye et al., 2025). By leveraging these inherent capabilities, we aim to enhance the model's reasoning potential without annotated data. We find that different reasoning trajectories sampled by LLMs for the same question tend to share common prefix substrings, suggesting a locally correct supervision signal, which we term **Prefix Self-Consistency**. Errors typically occur in the later steps of the reasoning trajectory (see Section 4.6 for a detailed discussion).

Our extensive experiments across four popular training datasets demonstrate that, in a setting without posterior sampling, our method outperforms vanilla full-token fine-tuning. Moreover, under a rejection sampling setup, our proposed method achieves performance comparable to RFT while requiring only $\frac{3}{10}$ of the training time and $\frac{1}{16}$ of the inference time.

In summary, our contributions are three-fold:

1. **Unsupervised Fine-Tuning with Minimal Guidance**: We introduce a simple fine-tuning technique called Unsupervised Rejection-Free Fine-Tuning (U-RFT) that utilizes only prefix substrings to steer the model toward better reasoning performance, minimizing the need for extensive labeled datasets or elaborate prompts.

2. **Empirical Validation**: We conduct comprehensive experiments across four training corpora and evaluate on four widely used reasoning benchmarks. U-RFT demonstrates exceptional data efficiency and flexibility, outperforming vanilla full-token fine-tuning in an unsupervised sampling setting. Furthermore, U-RFT achieves performance competitive with RFT while requiring only $\frac{3}{10}$ of the training time and $\frac{1}{16}$ of the inference time.

3. **Analysis of Model Behavior**: We provide an in-depth analysis and case studies of Prefix Self-Consistency to explain how U-RFT works, offering insights into the underlying mechanisms. This analysis contributes to the understanding of unsupervised post-training by identifying reliable self-supervised signals without the need for external supervision and extensive sampling.

2 Related Work

Unsupervised Learning Traditional unsupervised methodologies include Pseudo-labeling (Ye et al., 2020), the Pivot-based approach (Pan et al., 2010), and adversarial neural networks (Ganin et al., 2016). Given the success of the self-supervised learning paradigm in leveraging large-scale unlabeled data, pre-trained language models (Kenton and Toutanova, 2019; Han et al., 2021; Radford et al., 2019) based on self-supervision have become the standard in language

pre-training. Another self-supervised method under the new paradigm of self-improvement is called self-rewarding (Chen et al., 2024; Yuan et al., 2024), where the model iteratively enhances its performance through self-rewarding mechanisms without the need for external supervision signals. Furthermore, AL (Ji et al., 2024) extends the self-improvement paradigm to general unsupervised domain adaptation scenarios, where the model interacts with unsupervised corpora to perform unsupervised continual pre-training. In this paper, we focus on how to find self-supervised signals in the field of mathematical reasoning. We propose U-RFT, a simple yet effective unsupervised post-training method that requires only a set of questions and the LLM itself.

Self-Training and Self-Improvement. A family of methods, starting with STaR (Zelikman et al., 2022), reinforced self-training (Gulcehre et al., 2023), and rejection fine-tuning (Yuan et al., 2023), leverages the solutions generated by large language models (LLMs) to iteratively update and improve the models themselves. These techniques involve fine-tuning the model on the generated solutions that produce correct answers. ReST^{EM} (Singh et al., 2023) views this fine-tuning process as expectation-maximization-based reinforcement learning (RL) for a solution-generating agent. Wang et al. (2023a) propose using a contrastive loss to enhance the likelihood of correct solutions over incorrect ones. The discovery of successful solutions presents a significant exploration challenge. Luong et al. (2024) demonstrated that RL-based fine-tuning of an LLM is particularly difficult unless preceded by several steps of supervised fine-tuning. In An et al. (2023), a more powerful LLM was employed to edit the incorrect rationales generated by a smaller model, thereby providing positive data for its fine-tuning. However, Huang et al. (2023) argued that LLMs have limited capacity to correct their own reasoning flaws.

3 Method

3.1 Unsupervised Rejection-Free Sampling Fine-Tuning

The U-RFT starts from a pre-trained LLM, denoted as Φ_{θ^0} and a set of questions $Q = \{q_1, q_2, ..., q_m\}$ to be aligned. We first use model Φ_{θ^0} to perform a single sampling for each question q in Q.

$$a = \Phi_{\theta^0}(\text{Template}(q)) \tag{1}$$

where the Template$(\cdot)$ represents the default `Chat Template` of Φ_{θ^0}, used to wrap q as an instruction prompt.

After performing a single sampling on all questions, we obtain a set of data pairs $P = \{(q_i, a_i)\}_{i=1}^{M}$ with size M. Next, we proceed with prefix fine-tuning using these data pairs. Specifically, for each generated answer a_i containing n_i tokens, we only use the first k tokens (where $k \ll n_i$) along with the template T shown in Figure 2 for fine-tuning.

Formally, for each answer a_i, we extract the first k tokens:

$$a_i^{1:k} = \{a_i^1, a_i^2, \ldots, a_i^k\} \tag{2}$$

The optimization objective is to minimize this loss function:

$$\mathcal{L}(q_i, a_i) = -\log P(a_i^{1:k} | q_i; \theta^0) \tag{3}$$

Algorithm 1 The algorithm of U-RFT

Input: Φ_{θ^0}, Q, p, k
Output: Φ_{θ^1}

// Sampling
$\mathrm{T}_0 \leftarrow$ //Default `Chat Template` of Φ_{θ^0}
$\mathrm{T} \leftarrow$ //`Chat Template` of initial step training
$\mathrm{P} \leftarrow \phi$//Initialize an empty data set
$\Phi_{\theta^1} \leftarrow \Phi_{\theta^0}$//Initialize the SFT model using Φ_{θ^0}
for question q in $\mathcal{Q}$ **do**
  $\mathrm{a} \leftarrow \Phi_{\theta^0}(\mathrm{T}_0(\mathrm{q}))$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathrm{q}, \mathrm{a})\}$
end for
$\mathrm{M} \leftarrow$ length//Get the data size of Q
threshold $\leftarrow \mathrm{M} \times \mathrm{p}$//Initialize an empty data set
// Unsupervised Rejection-Free Sampling Fine-Tuning
for $i, (q, a)$ in enumerate$(\mathcal{P})$ **do**
  $\mathrm{n} \leftarrow$ //Get the length of a
  if $i >$ threshold **then**
    $\ell_1 \leftarrow -\log P(a^{1:k} | T(q); \theta^1)$
    $\theta^1 \leftarrow$ UpdateParameters(ℓ_1, θ^1)
  else
    $\ell_1 \leftarrow -\log P(a^{1:n} | T_0(q); \theta^1)$
    $\theta^1 \leftarrow$ UpdateParameters(ℓ_1, θ^1)
  end if
end for
return Φ_{θ^0}

3.2 Reasoning Structure Tuning

To prevent catastrophic forgetting of reasoning structures and length generalization (Muennighoff et al., 2025) during the prefix learning process, we have designed a simple yet effective multi-task learning method to address this issue. Specifically, we randomly select a subset $P_s = \{(q_j, a_j)\}_{j=1}^{L}$

The prompt for initial step learning

```
[question] Please provide the
initial step towards resolving the
question. This step may serve
as a foundation but might not
encompass the entire solution.
```

Figure 2: The prompt for initial step learning. [question] represents the question that needs to be answered.

of the training data for reasoning structure tuning, where L is the length of P_s and $L/M = p$. During the reasoning structure tuning stage, all tokens in a and q will be used to align the model's reasoning format. Formally, given a set of data pairs $P_s = \{(q_j, a_j)\}$, the optimization objective of reasoning structure tuning is defined as:

$$\mathcal{L}(q_j, a_j) = -\log P(a_j^{1:n} | q_j; \theta^0) \quad (4)$$

In summary, the entire algorithm flow of U-RFT is shown in Algorithm 1.

4 Experiment

In this section, we will introduce the systematic experiments conducted in this paper. We compare U-RFT to traditional supervised-based methods like SFT and RFT under the supervised sampling and unsupervised sampling settings, assess its scalability with different self-training corpora and different backbone models. We also analyze the impact of reasoning structure tuning and prefix length on performance, and present a detailed case study and rollout performance to provide a deeper understanding of U-RFT.

4.1 Experiments Setup

Backbone Model. For backbone model, we use three open-source models on huggingface[1].

- Llama-3.1-8B-Instruct (Dubey et al., 2024): an general instruction-tuned language model based on the Llama 3 architecture.
- Qwen2.5-Math-7B-Instruct (Yang et al., 2024): a specialized model in the Tongyi Qwen2.5 series that has been optimized for mathematical tasks.
- DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025): an open source 7B checkpoints based on Qwen2.5-Math-7B and distilled from DeepSeek-R1.

[1]https://huggingface.co/

Self-Training Dataset	Questions
PRM-12K (Lightman et al., 2024)	12K
OpenMathInstruct2 (Toshniwal et al., 2024)	600K
U-Hard	100K
LIMO (Ye et al., 2025)	819

Table 1: Comparison of various self-training datasets.

Self-Training Corpora Construction We employ four dataset as our training dataset to generate self-training data to perform U-RFT and other baselines. The statistic details are illustrated in Table 1.

- PRM-12K (Lightman et al., 2024): it contains 12K math questions, including data from 4.5K MATH (Hendrycks et al., 2021) test problems in the PRM800K training set.
- OpenMathInstruct2-600K (OMI2) (Toshniwal et al., 2024): OpenMathInstruct2 is a math instruction tuning dataset consisting of 14 million problem-solution pairs generated using the Llama3.1-405B-Instruct model. By deduplicating the questions, we obtained a set of 600,000 unique questions. We refer to this dataset as OMI2-600K in the following sections for convenience.
- LIMO (Ye et al., 2025): LIMO demonstrates that models can achieve superior performance with significantly less but higher quality training data, comprising a total of 817 questions. We introduce LIMO because of its high difficulty level, making the cost of filtering to obtain the correct paths through large-scale rejection sampling very high. Therefore, it is naturally well-suited for the unsupervised setting studied in this paper.
- U-Hard: To further explore the potential of U-RFT in an unsupervised setting, we introduce a question set with real-world practical significance called U-Hard-100K. U-Hard is a dataset containing 100,000 questions. We began by extensively collecting a large number of questions from publicly available online sources. Following the Omni-math (Gao et al., 2024) approach, we labeled the difficulty of the data. By filtering out questions

with lower difficulty scores, we ultimately curated a dataset consisting only of challenging questions. We refer to this dataset as U-Hard-100K in subsequent sections.

For each data source, we used the corresponding base model to sample questions. We considered two experimental settings: **unsupervised sampling** and **supervised sampling**. In the unsupervised sampling setting, we sampled each question only once without posterior filtering; supervised sampling follows the RFT setting, where each question is sampled multiple times and then the standard answers are used to filter multiple samples to obtain the final question set.

Implementation Details It is worth noting that when the best performance is achieved under the vanilla SFT setting, we fix the current set of hyperparameters for the training of U-RFT for a fair comparison. Except for LIMO, the default value of p is set to 0.1 for all self-training datasets. Due to the limited amount of LIMO data, we increase its proportion for reasoning format tuning, setting p to 0.3. We set k to 8, 32, and 128 for Llama-3.1-8B-Instruct, Qwen2.5-Math-7B-Instruct, and Deepseek-R1-Distill-Qwen-7B, respectively. Implementation details is shown in Appendix C.

Benchmarks. We evaluate the reasoning performance of model on four downstream reasoning datasets: 1) **GSM8K** (Cobbe et al., 2021): a dataset of high-quality, linguistically diverse grade school math word problems created by human problem writers; 2) **MATH500** (Hendrycks et al., 2021): a rigorous dataset comprising problems from high school math competitions; 3) **AIME24**: the 2024 American Invitational Mathematics Examination; 4) **GPQA** (Rein et al., 2023): a challenging dataset designed to evaluate the capabilities of LLMs and scalable oversight mechanisms. For the evaluation of all benchmarks, we use accuracy as the evaluation metric.

Baselines. Following RFT (Yuan et al., 2023) and V-STaR (Hosseini et al., 2024), we select the following methods into our comparison:

- 1) In-Context Learning (ICL) (Wei et al., 2022b): We use ICL to perform reasoning.
- 2) Vanilla Supervised Fine-Tunign (Vanilla SFT) (Ouyang et al., 2022): Applying fine-tuning on training data without any self-improvement or test-time verification.
- 3) RFT (Yuan et al., 2023): After running 16 times sampling and using label filter to these candidate answers, then we train the model only 1 iteration.
- 4) V-STaR [1 Iter] (Hosseini et al., 2024): Bootstrapping a generator and training a verifier for 1 iteration with 16 answers from V-STaR's generator.

4.2 Main Results

We conducted extensive experiments on the four datasets mentioned in Section 4.1. The experimental results under unsupervised sampling are shown in Table 2. In a single-sample unsupervised setting, the performance of vanilla SFT decreases on both OMI2-600K and U-Hard-100K. In contrast, U-RFT surpasses the performance of vanilla SFT across all model and training corpus settings. In n the single-sample unsupervised setting, the performance of vanilla SFT decreases on both OMI2-600K and U-Hard-100K datasets. Specifically, on the OMI2-600K dataset, U-RFT achieves an average accuracy of 52.9, which represents an improvement of 3.2 points over the vanilla SFT's average accuracy of 49.7, and an improvement of 1.5 points over the ICL's average accuracy of 51.4.

For comparison purposes, we also conduct experiments under a supervised sampling setup. The experimental results are shown in Table 3. Similarly, on the U-Hard-100K dataset, U-RFT achieves an average accuracy of 61.6. This marks a significant improvement of points over the vanilla SFT's average accuracy of 56.4, and an improvement of 4.4 points over the ICL's average accuracy of 57.2. These results demonstrate the superior performance of U-RFT across different models and training corpus settings. We perform extensive sampling on the PRM12K dataset to collect trajectories leading to the correct answer for each question. The results demonstrate that U-RFT exhibits exceptional data efficiency and SOTA performance across various reasoning benchmarks. For both Llama-3.1-8B-Instruct and Qwen2.5-Math-7B-Instruct models, U-RFT$_{t=16}$ + label filter achieves the highest average accuracy. Moreover, even with a single sampling (t=1), U-RFT achieves performance competitive with RFT$_{t=16}$ + label filter and V-STaR, indicating that U-RFT can effectively enhance model performance with minimal sampling.

Dataset	Model	GSM8K	MATH500	AIME2024	GPQA	Avg Acc.
OMI2-600K	**Qwen2.5-Math-7B-Instruct**					
	ICL	95.2	84.0	16.7	9.6	51.4
	Vanilla SFT$_{t=1}$	95.4	83.4	13.3	6.6	49.7
	U-RFT$_{t=1}$	95.4	86.4	20.0	9.6	**52.9**
U-Hard-100K	**Qwen2.5-Math-7B-Instruct**					
	ICL	95.2	84.0	16.7	9.6	51.4
	Vanilla SFT$_{t=1}$	95.5	83.4	16.7	9.6	51.3
	U-RFT$_{t=1}$	96.0	85.6	26.6	9.6	**54.5**
	DeepSeek-R1-Distill-Qwen-7B					
	ICL	88.6	87.0	40.0	13.1	57.2
	Vanilla SFT$_{t=1}$	89.7	87.0	36.7	12.1	56.4
	U-RFT$_{t=1}$	91.4	89.2	50.0	15.7	**61.6**
LIMO-800	**Qwen2.5-Math-7B-Instruct**					
	ICL	95.2	84.0	16.7	9.6	51.4
	Vanilla SFT$_{t=1}$	95.8	84.2	20.0	7.6	51.9
	U-RFT$_{t=1}$	95.6	85.8	20.0	8.6	**52.5**
	DeepSeek-R1-Distill-Qwen-7B					
	ICL	88.6	87.0	40.0	13.1	57.2
	Vanilla SFT$_{t=1}$	89.7	87.0	40.0	12.1	57.2
	U-RFT$_{t=1}$	92.0	89.4	43.3	17.7	**60.6**

Table 2: Comparison of model performance (Acc.) across various reasoning benchmarks under the unsupervised sampling setting. The backbone models include state-of-the-art LLMs such as Qwen2.5-Math-7B-Instruct and DeepSeek-R1-Distill-Qwen-7B. $t = 1$ means only a single sampling is performed, without filtering the data based on the correct extract answer.

4.3 Ablation Study

The ablation results shown in Table 4 clearly show that the integration of U-RFT with RST results in the best performance across the benchmarks. The significant drop in accuracy for U-RFT w/o RST underscores the necessity of RST in achieving optimal performance. For a detailed experimental analysis of the RST data proportion p and prefix length k, please refer to the subsequent chapters.

4.4 Impact of Prefix Length

Table 5 and Table 6 show our comprehensive comparison experiments regarding the prefix length. The highest average accuracy of 68.7 is achieved at $k = 8$. As k increases beyond 8, the average performance tends to decrease slightly. The optimal prefix length for the LIMO dataset appears to be $k = 128$, achieving the highest average accuracy of 66.4. The optimal k values for the two models are different. Possible reasons are: 1) Stronger models have reasoning chains with longer common prefix patterns; 2) Difficult questions often lead models to generate longer outputs, resulting in longer similar substrings. Therefore, to further explore the underlying mechanisms of U-RFT and the values of k, we conducted a case study and comprehensive rollout experiments in Section section.

4.5 Impact of Reasoning Structure Tuning

To investigate the impact of reasoning structure tuning, we conducted comprehensive experiments by varying the proportion of this part of the data. Specifically, we used llama-3.1-8B-Instruct in a supervised setting on PRM12K. To mitigate the potential interference of incorrect paths in the experiment, we adopted a supervised setting. The results are shown in Table 7. Without mixing reasoning structure tuning data for fine-tuning, the model's performance on GSM8K and MATH500 was 2.0 and 10.2, respectively. This indicates that without full-token fine-tuning, the model performs extremely poorly on reasoning tasks and cannot effectively handle complex reasoning. Further observation of the model's reasoning results shows that this disrupts the model's instruction-following capability for reasoning structures, affecting its generalization in the mathematical domain as well as its generalization for reasoning length.

When the data ratio for full-token fine-tuning was 50% and 100%, the model's performance on GSM8K and MATH500 was 86.3 and 52.0 (50% ratio) and 86.0 and 52.0 (100% ratio), respectively. This indicates that increasing the full-token fine-tuning data ratio beyond 10% provides very limited performance improvement.

Dataset	Model	GSM8K	MATH500	AIME2024	GPQA	Avg Acc.
PRM-12K	**Llama-3.1-8B-Instruct**					
	ICL	82.0	51.0	3.3	8.6	36.2
	Vanilla SFT$_{t=1}$	83.8	48.4	3.3	8.6	36.0
	RFT$_{t=16}$ + label filter	86.0	52.0	6.7	9.1	38.5
	V-STaR$_{t=16}$	85.4	52.6	6.7	8.6	38.3
	U-RFT$_{t=16}$ + label filter	85.8	53.4	6.7	9.1	**38.8**
	U-RFT$_{t=1}$	85.4	52.0	6.7	9.1	38.3
	Qwen2.5-Math-7B-Instruct					
	ICL	95.2	84.0	16.7	9.6	51.4
	Vanilla SFT$_{t=1}$	95.8	83.4	13.3	9.1	50.4
	RFT$_{t=16}$ + label filter	95.7	85.2	20.0	9.6	52.6
	V-STaR$_{t=16}$	96.0	85.4	20.0	10.1	**52.9**
	U-RFT$_{t=16}$ + label filter	96.0	85.6	20.0	10.1	**52.9**
	U-RFT$_{t=1}$	95.5	85.6	20.0	9.6	52.6

Table 3: Comparison of model performance (Acc.) across various reasoning benchmarks under the two sampling settings. The backbone models include state-of-the-art LLMs such as Qwen2.5-Math-7B-Instruct and DeepSeek-R1-Distill-Qwen-7B. $t = 1$ means only a single sampling is performed, without filtering the data based on the correct extract answer.

Methods	GSM8K	MATH500	Avg.
U-RFT w/o RST	2.0	10.2	6.1
vanilla SFT	83.8	48.8	66.1
RST only	84.3	48.4	66.4
U-RFT	85.4	52.0	68.7

Table 4: Ablation study of U-RFT based on Llama-3.1-8B-Instruct trained on PRM12-12K.RST stands for reasoning structure tuning, using a subset of the total data for format fine-tuning.

k	GSM8K	MATH500	Avg.
1	84.9	51.8	68.4
2	84.2	51.8	68.0
4	84.8	51.8	68.3
8	**85.4**	**52.0**	**68.7**
16	85.4	50.2	67.8
32	84.8	49.4	67.1
64	84.8	48.2	66.5

Table 5: Effect of the prefix length for math domain generalization.Experiments are conducted on the PRM12K dataset, using Llama-3.1-8B-Instruct as the backbone model.

k	GSM8K	MATH500	GPQA	Avg.
8	91.7	87.2	15.7	64.9
32	91.5	87.0	16.6	65.0
64	91.6	89.2	13.1	64.6
128	**92.0**	89.4	**17.7**	**66.4**
256	90.5	**90.4**	**17.7**	66.2

Table 6: Effect of the prefix length for math domain generalization.Experiments are conducted on the LIMO dataset, using DeepSeek-R1-Distill-Qwen-7B as the backbone model.

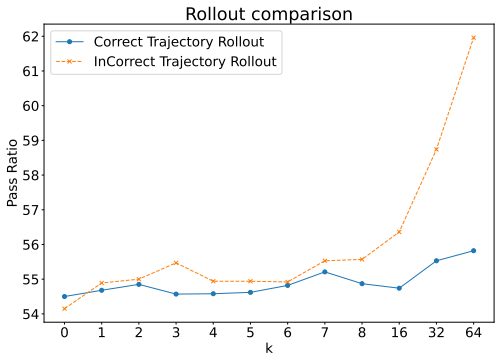

Figure 3: Performance comparison of rollout based on correct trajectory and incorrect trajectory on PRM-12K dataset using Llama-3.1-8B-Instruct.

4.6 Discussion about Prefix Self-Consistency

Figure 4 presents a multi-sampling case using Qwen2.5-Math-7B-Instruct. By examining the first 32 words of these responses, we can see that the first 32 tokens of different responses are very similar. This indicates that during multiple reasoning processes, the model exhibits a high degree of consistency in the initial steps. This consistency reflects the model's stability in the preliminary reasoning stage, such as understanding the problem and formulating the overall strategy. Further observation reveals that the divergent information between different samples often occurs in the later stages. Multiple divergent later stages often share the same prefix. Based on this insight, we designed a rollout experiment at different positions. Specifically, we randomly sampled 500 questions from prm12K, each containing one trajectory lead-

Case Study

Question: Melinda has three empty boxes and 12 textbooks, three of which are mathematics textbooks. One box will hold any three of her textbooks, one will hold any four of her textbooks, and one will hold any five of her textbooks. If Melinda packs her textbooks into these boxes in random order, the probability that all three mathematics textbooks end up in the same box can be written as $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $m+n$.

The first 32 words (prefix substring) of all answers:

A1: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A2: To determine the probability that all three mathematics textbooks end up in the same box, we need to follow these steps: 1. **Calculate the total number of ways to pack the textbooks

A3: To determine the probability that all three mathematics textbooks end up in the same box, we start by calculating the total number of ways to distribute the 12 textbooks into the three

A4: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

Figure 4: With the temperature set to 0.7, we sample 16 times based on Qwen2.5-Math-Instruct for the given question, where A1-A4 represent the corresponding eight output results. Due to space limitations, please refer to Figure 5 for the complete 16 sampling results.

p	GSM8K	MATH500	Avg.
0%	2.0	10.2	6.1
1%	57.5	27.8	42.7
5%	84.5	50.0	67.3
10%	85.8	**53.4**	**69.6**
50%	**86.3**	52.0	69.2
100%	86.0	52.0	69.0

Table 7: Effect of data ratio for reasoning structure tuning.Here we conduct experiments based on PRM12K.Note that when the RST data is 100%, the corresponding term is equivalent to vanilla full-token fine-tuning.

ing to the correct answer and one trajectory leading to the wrong answer. We refer to these as supervised rollout and unsupervised rollout settings. During reasoning, the model is given the first k tokens of the trajectory to perform 32 rollout samples. The result is shown in Figure 3. At smaller k values (such as 1 to 8), the differences between the metrics for posterior and non-posterior are minimal, indicating that posterior information does not significantly enhance performance in these cases. Starting from k=8, the accuracy for posterior begin to significantly exceed those for non-posterior, and this difference becomes more pronounced at larger k values (such as 64). This suggests that at larger k values, posterior information significantly improves the model's performance. This phenomenon might indicate that as the k value increases, the model has more information for reasoning and adjustment, thus allowing the posterior information to play a greater role, significantly enhancing the performance metrics. The detailed rollout results can be found in Table 8 and Table 9.

5 Conclusion

In this work, we presented an unsupervised fine-tuning method that enhances the reasoning capabilities of large language models using only prefix substrings as minimal guidance. Our approach leverages the inherent reasoning structures within pretrained models, exploiting the phenomenon of Prefix Self-Consistency where different reasoning trajectories share common prefixes. Extensive experiments demonstrated that our method outperforms traditional full-token fine-tuning and achieves performance comparable to supervised approaches like RFT, with significantly reduced training and inference times. This work highlights the potential of minimal unsupervised fine-tuning in improving the reasoning abilities of LLMs without relying on external supervision or extensive computational resources. Future work will explore the application of this method to other challenging tasks and investigate the theoretical underpinnings of Prefix Self-Consistency in more depth.

Limitations

- **Implementation within the R1 framework:** In the RL framework, online search by the model is costly and often requires numerous searches to find the correct posterior trajectory. Although our method has achieved excellent performance with Unsupervised Fine-tuning, how to extend this method to the currently popular R1-Zero framework has yet to be explored. Future research should focus on how to combine our method with RL to achieve data-efficient RL.
- **Validation with larger-scale models:** In our experiments, the choice of backbone was primarily based on the mathematical capabilities of the model. Our goal is to further enhance the model's reasoning ability at a low cost. In future research within the RL framework, we will consider validating our approach on larger-scale models, such as QwQ-32B-Preview and DeepSeek-R1-Distill-Qwen-32B.
- **Study on sample-independent prefix selection strategy:** Despite achieving satisfactory performance, this work is based on a heuristic prefix selection strategy. In future research, we will continue to design an adaptive sample-independent prefix selection for U-RFT.

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, and 1 others. 2021. Pre-trained models: Past, present and future. *AI Open*, 2:225–250.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Ke Ji, Junying Chen, Anningzhe Gao, Wenya Xie, Xiang Wan, and Benyou Wang. 2024. Llms could autonomously learn without external supervision. *arXiv preprint arXiv:2406.00606*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and 1 others. 2023. The flan collection: Designing data and methods for effective instruction tuning.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, and 22 others. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *Preprint*, arXiv:2211.14275.

Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. 2023a. Making large language models better reasoners with alignment. *arXiv preprint arXiv:2309.02144*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022a. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Hai Ye, Qingyu Tan, Ruidan He, Juntao Li, Hwee Tou Ng, and Lidong Bing. 2020. Feature adaptation of pre-trained language models across languages and domains with robust self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7386–7399.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston.

2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

A Ethics Statement

The datasets used in this study are all sourced from publicly available resources on the internet and are freely accessible. Additionally, the backbone models we employ are also publicly available. Therefore, there are no ethical concerns related to this study.

k	Pass@2	Pass@4	Pass@8	Pass@16
0	68.01	78.53	85.98	91.10
1	68.34	78.15	84.76	89.20
2	68.46	78.44	85.38	90.25
3	68.97	78.75	85.33	89.71
4	68.56	78.43	84.99	89.49
5	68.44	78.38	85.25	90.10
6	68.60	78.67	85.49	90.10
7	69.08	79.01	85.89	90.59
8	69.08	78.94	85.65	90.16
16	69.76	79.27	85.75	90.36
32	71.08	79.65	85.44	89.52
64	73.60	81.52	87.03	91.06

Table 8: Correct solution rollout based on PRM-12K dataset using Llama-3.1-8B-Instruct.

k	Pass@2	Pass@4	Pass@8	Pass@16
0	68.39	78.91	86.24	91.22
1	68.43	78.84	86.07	91.19
2	68.52	78.94	86.31	91.33
3	68.30	78.71	85.98	91.13
4	68.21	78.57	85.88	90.97
5	68.24	78.7	86.21	91.53
6	68.41	78.83	86.41	91.99
7	68.77	79.04	86.12	90.99
8	68.43	78.69	85.98	91.16
16	67.72	77.69	84.91	90.27
32	67.68	77.23	84.35	89.56
64	65.83	74.05	80.67	85.93

Table 9: Incorrect solution rollout based on PRM-12K dataset using Llama-3.1-8B-Instruct.

Hyperparam.	Math-12K	OMI2-60K	LIMO	U-Hard
Optimizer	AdamW			
Warmup Ratio	0.03			
Learning Rate	1e-6		2e-6	1e-6
LR Schedule	constant_with_warmup			
Batch Size	1			
Gradient Step	8			
Max Length	4096		16384	
# Epoch	2	1	3	1

Table 10: The hyperparameters used for our method on all training corpora.

B Detailed Rollout Results

Table 8 shows the correct solution rollout where increasing k generally leads to improved performance across various pass metrics, indicating that longer prefixes contribute to better modeling of context. Notably, performance peaks at $k = 64$ for Pass@2, achieving 73.60%, and stabilizes across higher k values for other metrics, suggesting a saturation point beyond which additional prefix length offers diminishing returns. In contrast, Table 9, which details incorrect solution rollouts, follows a similar trend but with slightly lower pass rates. However, beyond $k = 8$, the performance starts to decline. For example, Pass@2 decreases to 67.68% at $k = 32$ and further to 67.53% at $k = 64$. A similar trend is observed for Pass@4, Pass@8, and Pass@16, where the performance drops after $k = 8$.

C Hyperparameters and Implementation

The training hyperparameters of U-RFT on different datasets are reported in Table 10. To ensure a fair comparison, please note that all hyperparameter settings are based on the vanilla SFT performance configuration. During the inference stage, we adopt a prompted zero-shot setup and use standard greedy decoding, wherein models are directed to answer each question using natural language instructions without any accompanying contextual demonstrations.

Case Study

Question: Melinda has three empty boxes and 12 textbooks, three of which are mathematics textbooks. One box will hold any three of her textbooks, one will hold any four of her textbooks, and one will hold any five of her textbooks. If Melinda packs her textbooks into these boxes in random order, the probability that all three mathematics textbooks end up in the same box can be written as $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $m+n$.

The first 32 words (prefix substring) of all answers:

A1: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A2: To determine the probability that all three mathematics textbooks end up in the same box, we need to follow these steps: 1. **Calculate the total number of ways to pack the textbooks

A3: To determine the probability that all three mathematics textbooks end up in the same box, we start by calculating the total number of ways to distribute the 12 textbooks into the three

A4: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A5: To determine the probability that all three mathematics textbooks end up in the same box, we start by calculating the total number of ways to distribute the 12 textbooks into the three

A6: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A7: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A8: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A9: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A10: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A11: To determine the probability that all three mathematics textbooks end up in the same box, we start by calculating the total number of ways to distribute the 12 textbooks into the three

A12: To determine the probability that all three mathematics textbooks end up in the same box, we need to follow these steps: 1. **Calculate the total number of ways to distribute the 12

A13: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A14: To determine the probability that all three mathematics textbooks end up in the same box, we need to follow these steps: 1. **Calculate the total number of ways to pack the textbooks

A15: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

A16: To determine the probability that all three mathematics textbooks end up in the same box, we need to consider the total number of ways to distribute the 12 textbooks into the three

Figure 5: With the temperature set to 0.7, we sample 16 times based on Qwen2.5-Math-Instruct for the given question, where A1-A16 represent the corresponding eight output results.