Reinforcement Pre-Training

Qingxiu Dong[*†‡] **Li Dong**[*†]
Yao Tang[†] **Tianzhu Ye**[†§] **Yutao Sun**[†§] **Zhifang Sui**[‡] **Furu Wei**[†◇]
[†] Microsoft Research
[‡] Peking University
[§] Tsinghua University
https://aka.ms/GeneralAI

In this work, we introduce **Reinforcement Pre-Training** (RPT) as a new scaling paradigm for large language models and reinforcement learning (RL). Specifically, we reframe next-token prediction as a reasoning task trained using RL, where it receives verifiable rewards for correctly predicting the next token for a given context. RPT offers a scalable method to leverage vast amounts of text data for **general-purpose RL**, rather than relying on domain-specific annotated answers. By incentivizing the capability of next-token reasoning, RPT significantly improves the language modeling accuracy of predicting the next tokens. Moreover, RPT provides a strong pre-trained foundation for further reinforcement fine-tuning. The scaling curves show that increased training compute consistently improves the next-token prediction accuracy. The results position RPT as an effective and promising scaling paradigm to advance language model pre-training.

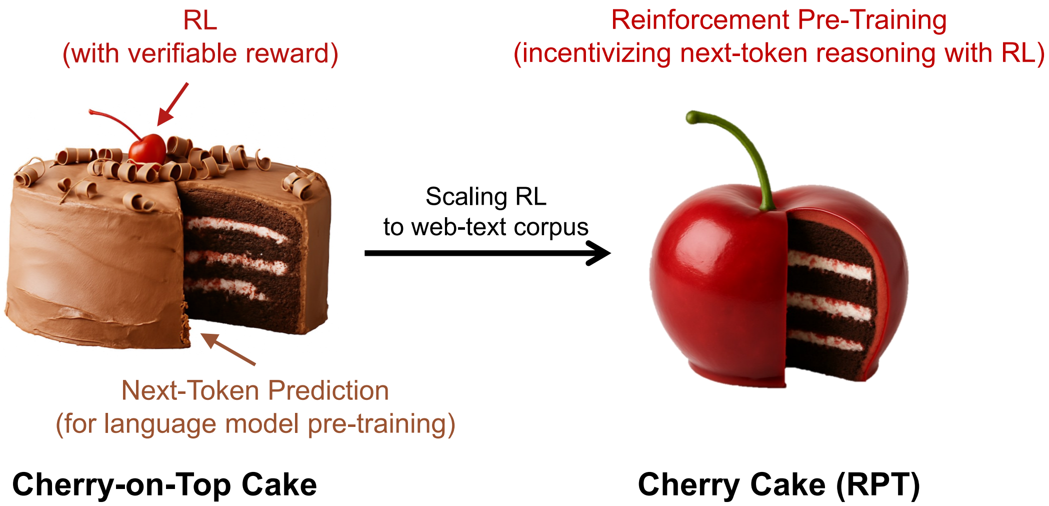


Figure 1: Reinforcement pre-training (RPT) reframes next-token prediction as a reasoning task, where the language model is incentivized via reinforcement learning (RL) to reason about and correctly predict the next token. The proposed approach allows RL to be scaled to the web-text corpus. The image of the cherry-on-top cake is taken from LeCun's slides [LeC16].

[*] Equal contribution. ◇ Contact person: fuwei@microsoft.com.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks, largely driven by the scalability of the next-token prediction objective on vast text corpora. This self-supervised paradigm has proven to be an effective general-purpose pre-training approach. Concurrently, reinforcement learning (RL) has emerged as a powerful technique for fine-tuning LLMs, aligning them with human preferences or enhancing specific skills such as complex reasoning [OWJ+22, JKL+24, GYZ+25].

However, current applications of RL in LLM training face scalability and generality challenges. Reinforcement learning from human feedback [OWJ+22], while effective for alignment, relies on costly human preference data, and its learned reward models can be susceptible to reward hacking, limiting scalability. Alternatively, reinforcement learning with verifiable rewards (RLVR) [LMP+25] utilizes objective, rule-based rewards, often from question-answer pairs. While this mitigates reward hacking, RLVR is typically constrained by the scarcity of annotated data with verifiable answers, restricting its application to domain-specific fine-tuning rather than general-purpose pre-training.

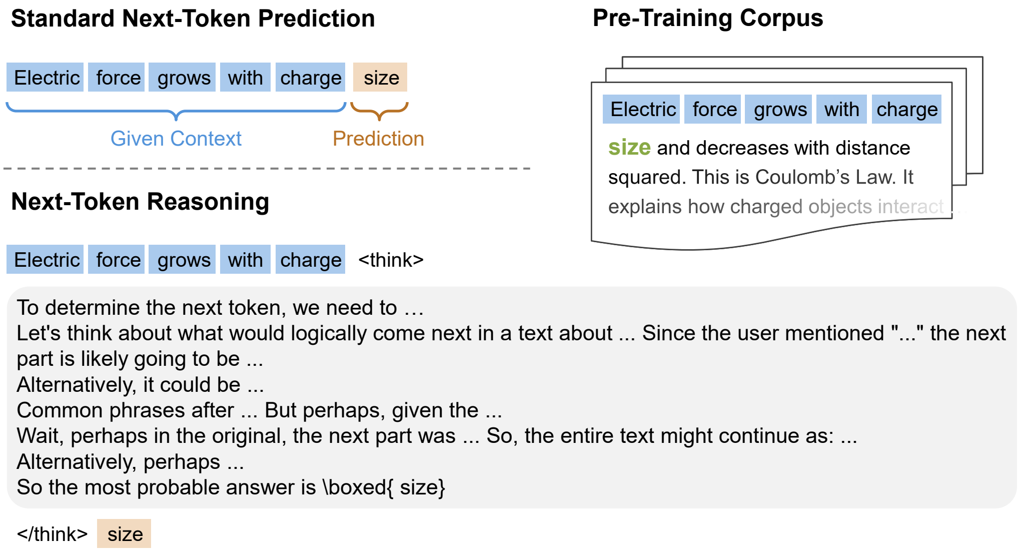
In this work, we introduce reinforcement pre-training (RPT), a novel paradigm that bridges the gap between scalable self-supervised pre-training and the power of reinforcement learning. RPT reframes the fundamental next-token prediction task as a next-token reasoning process. For any given context in a pre-training corpus, the model is incentivized to reason about the subsequent token before predicting it. It receives a verifiable, intrinsic reward based on the correctness of its prediction against the ground-truth next token from the corpus itself. This approach transforms the vast, unannotated text data typically used for next-token prediction into a massive dataset for general-purpose RL, without requiring external annotations or domain-specific reward functions.

This approach offers several crucial advantages. First, RPT is inherently scalable and general-purpose: it leverages the same vast, unannotated text data used for standard next-token prediction, transforming it into a massive dataset for general-purpose RL without requiring external annotations. Second, the use of direct, rule-based reward signals (i.e., the correctness of the predicted next token) inherently minimizes the risk of reward hacking often associated with complex, learned reward models. Third, by explicitly encouraging next-token reasoning patterns, RPT promotes deeper understanding and generalization instead of merely memorizing next tokens. The model learns to explore and validate hypotheses about why a certain token should follow, fostering more robust representations. Finally, the internal reasoning process during pre-training effectively allows the model to allocate more "thought" or computational effort to each prediction step, akin to a form of inference-time scaling applied at training time for each token, which directly contributes to improved next-token prediction accuracy.

Our experiments demonstrate that RPT significantly improves the accuracy of predicting next tokens. RPT also provides a more robust pre-trained foundation for subsequent reinforcement fine-tuning, leading to better final task performance. The scaling curves reveal that increased training compute under the RPT framework consistently improves next-token prediction accuracy, indicating its potential as a sustainable scaling strategy. These results position reinforcement pre-training as an effective and promising new paradigm to advance the pre-training of large language models.

Our contributions are summarized as follows:

- We introduce reinforcement pre-training (RPT), a new scaling paradigm that reframes next-token prediction as a reasoning task trained with reinforcement learning, utilizing intrinsic verifiable rewards derived directly from the pre-training corpus.
- RPT offers a scalable and general-purpose approach to RL pre-training, minimizing reward hacking through rule-based rewards and promoting generalization by encouraging next-token reasoning patterns over rote memorization.
- RPT significantly improves next-token prediction accuracy and exhibits favorable scaling properties, where performance consistently improves with increased training compute.
- RPT yields a stronger pre-trained foundation for subsequent reinforcement fine-tuning and enhances zero-shot performance on various downstream tasks.


Standard Next-Token Prediction
Electric force grows with charge size
Given Context
Prediction
Next-Token Reasoning
Electric force grows with charge <think>
</think> size
Pre-Training Corpus
Electric force grows with charge
size and decreases with distance squared. This is Coulomb's Law. It explains how charged objects interact

To determine the next token, we need to …
Let's think about what would logically come next in a text about ... Since the user mentioned "..." the next part is likely going to be ...
Alternatively, it could be ...
Common phrases after ... But perhaps, given the ...
Wait, perhaps in the original, the next part was ... So, the entire text might continue as: ...
Alternatively, perhaps ...
So the most probable answer is \boxed{ size}

Figure 2: Comparison of standard next-token prediction and next-token reasoning. Standard next-token prediction estimates the next token in the pre-training corpus directly, while next-token reasoning performs reasoning over multiple tokens before making the prediction.

2 Preliminary

Next-Token Prediction (NTP) Next-token prediction is the fundamental training objective for modern large language models [AAA+23]. Given an input sequence $x_0 \cdots x_T$ from the training corpus, the model is trained to maximize the following objective:

$$\mathcal{J}_{\text{NTP}}(\theta) = \sum_{t=1}^{T} \log P(x_t \mid x_0, x_1, \ldots, x_{t-1}; \theta), \tag{1}$$

where θ represents the parameters of the language model.

Reinforcement Learning with Verifiable Rewards (RLVR) RLVR employs a reinforcement learning objective to enhance specific skills with verifiable answers [LMP+25]. RLVR requires a labeled dataset of question-answer pairs $\mathcal{D} = \{(q, a)\}$. For a specific pair $(q, a) \in \mathcal{D}$, the LLM π_θ generates a response $o \sim \pi_\theta(\cdot \mid q)$. A deterministic verifier $\mathcal{V}$ calculates a verifiable reward $r = \mathcal{V}(o, a)$, and the model is trained to maximize the expected reward:

$$\mathcal{J}_{\text{RLVR}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\ o\sim\pi_\theta(\cdot|q)} \left[r(o, a) \right]. \tag{2}$$

3 Reinforcement Pre-Training

3.1 Pre-Training Task: Next-Token Reasoning

We propose the next-token reasoning task for language modeling. Given an input sequence $x_0 \cdots x_T$ from the training corpus, for each position $t \in \{1, \ldots, T\}$, the prefix $x_{<t}$ is treated as the context, and ground-truth next token is x_t. In the next-token reasoning task, the model π_θ is required to generate a chain-of-thought reasoning sequence, denoted by c_t, before generating a prediction y_t for the next token. The overall model response is $o_t = (c_t, y_t)$, $o_t \sim \pi_\theta(\cdot \mid x_{<t})$.

As illustrated in Figure 2, the long chain-of-thought process for next-token reasoning can involve various reasoning patterns such as brainstorming, self-critique and self-correction. The next-token reasoning task reconstructs the pre-training corpus into a vast set of reasoning problems, shifting pre-training beyond learning superficial token-level correlations to understanding the hidden knowledge behind them and making RL scaling possible.

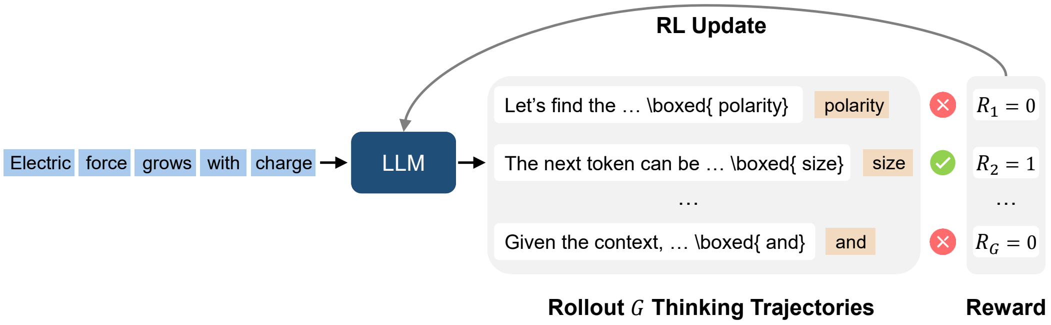

RL Update
Electric
force
grows
with
charge
LLM
Let's find the … \boxed{ polarity}
polarity
The next token can be … \boxed{ size}
size
…
Given the context, … \boxed{ and}
and
$R_1 = 0$
$R_2 = 1$
…
$R_G = 0$
Rollout G Thinking Trajectories
Reward

Figure 3: An illustration of reinforcement pre-training. Given a context with a missing continuation, the LLM performs on-policy rollouts to generate G different thinking trajectories. Each includes an intermediate reasoning step and a final prediction for the next token. A positive reward is assigned if the prediction matches the ground-truth token; otherwise, the reward is zero. This reward signal is used to update the LLM, encouraging trajectories that lead to accurate continuations.

3.2 Pre-Training with Reinforcement Learning

Reinforcement pre-training (RPT) trains LLMs to perform next-token reasoning via on-policy reinforcement learning, as illustrated in Figure 3. For the context $x_{<t}$, we prompt the language model π_θ to generate G responses (thinking trajectories), $\{o_t^i\}_{i=1}^G$. Each response $o_t^i = (c_t^i, y_t^i)$ consists of a chain-of-thought reasoning sequence c_t^i and a final prediction sequence y_t^i.

To verify the correctness of y_t^i, we introduce a **prefix matching reward**, which supports verifying predictions that span multiple tokens or involve out-of-vocabulary tokens.[2] Let $\overline{x}_{\geq t}$ and $\overline{y}_t^i$ denote the byte sequences of the ground-truth completion sequence $x_{\geq t}$ and the prediction y_t^i, respectively. Denote the byte length of $\overline{y}_t^i$ by l. We define the cumulative byte lengths of the tokens in the ground-truth completion sequence as valid boundaries, and denote this set by $\mathcal{L}_{gt}$. Formally, the reward r_t^i for the i-th output for $x_{<t}$ is defined as:

$$r_t^i = \begin{cases} 1 & \text{if } \overline{y}_t^i = \overline{x}_{\geq t}[1:l] \text{ and } l \in \mathcal{L}_{gt} \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where the reward is 1 if the byte sequence of the prediction is an exact prefix of the ground-truth completion sequence and its length l matches any valid token boundary.

Let $\mathcal{D}$ be the set of all $\{x_{<t}\}_{t=1}^T$, the model is trained to maximize the expected reward:

$$\mathcal{J}_{\text{RPT}}(\theta) = \mathbb{E}_{(x_{<t}, x_{\geq t}) \sim \mathcal{D},\ \{o_t^i\}_{i=1}^G \sim \pi_\theta(\cdot|x_{<t})} \left[r_t^i \right]. \tag{4}$$

3.3 Pre-Training Setup

We use the OmniMATH dataset [GSY+24] for reinforcement pre-training. OmniMATH contains 4,428 competition-level mathematical problems and solutions from official websites such as AoPS Wiki[3] and AoPS forum[4]. Since many tokens are easily predictable even without reasoning, we perform token-level data filtering before reinforcement pre-training. Particularly, we use Deepseek-R1-Distill-Qwen-1.5B as a small proxy model. For each token, we calculate the proxy model entropy on the top-16 next tokens. By applying an entropy threshold, we filter out low-entropy positions, prioritizing training on challenging tokens that require greater computational effort to predict.

In all experiments, we use Deepseek-R1-Distill-Qwen-14B [GYZ+25] as the base model. R1-Distill-Qwen-14B serves as a good starting point for reinforcement learning due to its basic reasoning capabilities. We implement our training framework with the verl library [SZY+24] and use vllm for inference. We employ the GRPO algorithm [GYZ+25], with specific hyperparameters detailed in

[2] Additional reward design choices for next-token reasoning are discussed in Appendix A.

[3] https://artofproblemsolving.com/wiki/index.php

[4] https://artofproblemsolving.com/community/c13_contests

Appendix B. During training, we adopt an 8k training length, a learning rate of 1×10^{-6}, zero KL penalty, and a batch size of 256 questions. For each question, G=8 responses are sampled, and for the rollout process, we use a temperature of 0.8. From each response, we directly extract the full sequence inside the last \boxed{} following the special token '</think>' as the model prediction for the next token. Starting from 500 steps, we utilize dynamic sampling to boost training efficiency [YZZ+25]. The total training steps for our main experiment is 1,000. The prompt template and its variants are discussed in Appendix D.

3.4 Evaluation of Pretrained Models

Once the model is pretrained, we can directly conduct next-token prediction and reinforcement fine-tuning on downstream tasks. We use the settings to show that reinforcement pre-training improves the language modeling capabilities and reasoning abilities of large language models.

Language Modeling Given the next-token reasoning objective, our models can be naturally used for language modeling. We report the next-token prediction accuracy to evaluate the language modeling performance and scaling properties of RPT.

Reinforcement Fine-Tuning on Downstream Tasks We conduct continual RL fine-tuning with RPT models in a pretrain-then-finetune manner. Since RPT aligns the pre-training process with RL, the objective gap between pre-training and RL during post-training is minimized. We evaluate whether the reinforcement pre-training process further enhances post-training on end tasks.

4 Experiments

4.1 Language Modeling

We evaluate the language modeling performance on a held-out validation set of 200 samples from OmniMATH. Following the entropy-based data filtering strategy described in our setup (Section 3.3), we categorize token positions in the validation set according to their difficulty. Specifically, we calculate the entropy at each token position using R1-Distill-Qwen-14B. We then designate positions as belonging to easy, medium, or hard splits if their entropy exceeds thresholds of 0.5, 1.0, and 1.5, respectively. For comparison, we report the performance of R1-Distill-Qwen-14B evaluated in two different ways: (1) Standard next-token prediction, selecting the token with the highest probability; and (2) Next-token reasoning, generating a chain-of-thought before the final prediction. We also include the results of Qwen2.5-14B, as it is the base model for R1-Distill-Qwen-14B.

	Easy	**Medium**	**Hard**
Standard next-token prediction			
Qwen2.5-14B	41.90	30.03	20.65
R1-Distill-Qwen-14B	41.60	29.46	20.43
Next-token reasoning			
R1-Distill-Qwen-14B	3.31	1.66	1.41
RPT-14B	**45.11**	**33.56**	**23.75**

Table 1: Next-token prediction accuracy across three test splits of varying difficulty. RPT outperforms both the standard next-token prediction baselines and the reasoning-based prediction baseline.

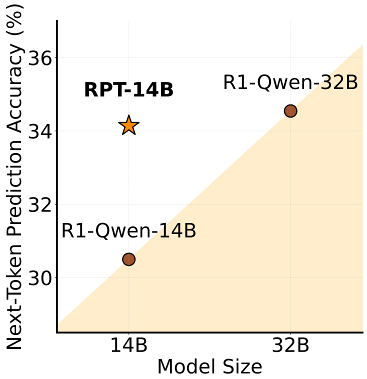


Figure 4: Average next-token prediction accuracy across data of different difficulty levels. R1-Qwen-14B/32B denote R1-Distill-Qwen-14B/32B, respectively.

As shown in Table 1, RPT-14B achieves consistently higher next-token prediction accuracy across all difficulty levels compared to R1-Distill-Qwen-14B. Remarkably, it matches the performance

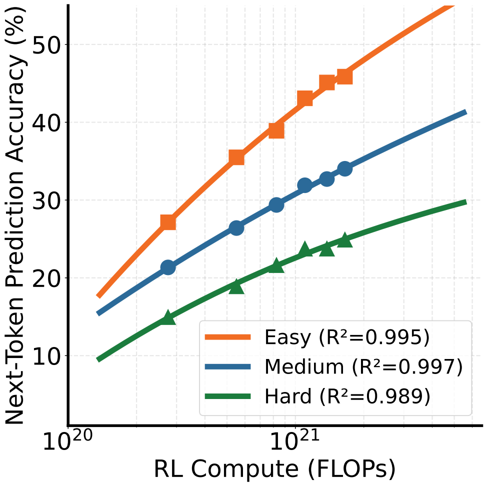


Figure 5: Next-token prediction accuracy of reinforcement pre-training improves consistently with increased training compute under all data difficulties. The fitted curves exhibit high coefficients of determination, indicating alignment between the predicted and observed values.

of a significantly larger model, i.e., R1-Distill-Qwen-32B (Figure 4). These results suggest that reinforcement pre-training is effective at capturing the complex reasoning signals underlying token generation, and holds strong potential for improving the language modeling capability of LLMs.

4.2 Scaling Properties of Reinforcement Pre-Training

In this section, we investigate the scaling properties of reinforcement pre-training. The loss achieved by next-token pre-training on natural language corpus empirically follows a power-law decay with respect to model size, number of training tokens, and training compute [HBM+22b, KMH+20]. Below, we analyze the scaling behavior of RPT specifically with respect to training compute C. We model this relationship using the following power-law form:

$$P(C) = \frac{A}{C^{\alpha}} + P^{*} \tag{5}$$

where $P(C)$ denotes the next-token prediction accuracy on the validation set. P^{*}, α, and A are parameters to be estimated.

We evaluate the next-token prediction accuracy of RPT at various training steps (100, 200, 400, 800, 1000, and 1200) and convert them into the corresponding training compute. To assess the impact of data difficulty, we consider validation splits filtered by entropy thresholds 0.5 (easy), 1.0 (medium), and 1.5 (hard). A higher threshold corresponds to more challenging inputs for the LLM. For each difficulty level, we fit the results according to Equation (5). We measure the goodness of fit using the coefficient of determination R^2, which quantifies how well the scaling curve fits the observed data.

As shown in Figure 5, the next-token prediction accuracy of RPT improves reliably as the training compute is scaled up. High R^2 values across all difficulty levels demonstrate that the fitted curves accurately capture performance trends.

4.3 Reinforcement Fine-Tuning with RPT

To investigate whether RPT models can be more effectively fine-tuned with RLVR, we randomly sample questions with verifiable answers from Skywork-OR1 [HLL+25] for further training. We use 256 examples for training and 200 for testing. Following the data filtering pipeline from Skywork-OR1 [HLL+25], we use R1-Distill-Qwen-32B to identify challenging instances for training. We set both the training batch size and the PPO mini-batch size to 64, and train the model for 15 epochs. During evaluation, the maximum number of tokens for validation is set to 32,000, with a temperature of 0.6.

As shown in Table 2, the reinforcement pre-trained model achieves a higher upper bound when further trained with RLVR. The reasoning ability of the model significantly declines when continually trained on the same data using a next-token prediction objective. Subsequent RLVR yields only slow performance improvements. These results indicate that with limited data, reinforcement pre-training can quickly transfer the strengthened reasoning patterns learned from next-token reasoning to end tasks.

	Before RL	**After RL**
R1-Distill-Qwen-14B	51.2	52.7
+ Continual NTP training	10.7	13.0
RPT-14B	**56.3**	**58.3**

Table 2: Reinforcement fine-tuning performance of different models. "Continual NTP training" means continual pre-training using standard next-token prediction objective on the same corpus as RPT-14B. RPT provides a stronger foundation for subsequent RL training.

4.4 Zero-Shot Performance on End Tasks

We evaluate the zero-shot performance of RPT-14B on end tasks. For comparison, we assess the next-token prediction performance of R1-Distill-Qwen-14B and R1-Distill-Qwen-32B, as well as the the reasoning performance of RPT-14B with R1-Distill-Qwen-14B.

Our evaluation involves two widely acknowledged benchmarks: MMLU-Pro [HBB$^+$20], a comprehensive multi-task understanding benchmark evaluating LLM capabilities across various domains; SuperGPQA [DYM$^+$25], a large-scale benchmark of graduate-level reasoning questions spanning 285 disciplines. Under the reasoning setting, we set the maximum number of tokens to 12,288 and the temperature to 0.8. Following previous works [MLJ$^+$25, ZLS$^+$25b], we use a multiple-choice question format for evaluation and report the accuracy.

	SuperGPQA	**MMLU-Pro**
Standard next-token prediction mode		
R1-Distill-Qwen-14B	32.0	48.4
R1-Distill-Qwen-32B	37.2	56.5
Reasoning mode		
R1-Distill-Qwen14B	36.1	68.9
RPT-14B	**39.0**	**71.1**

Table 3: Zero-shot performance on general-domain end tasks. RPT-14B in reasoning mode consistently outperforms 14B and 32B baselines.

As shown in Table 3, RPT-14B consistently outperforms R1-Distill-Qwen-14B (whether using standard next-token prediction or evaluated as a reasoning model) across all benchmarks. Notably, it also surpasses the significantly larger R1-Distill-Qwen-32B (under next-token prediction), with gains of 7 points on SuperGPQA and approximately 22 points on MMLU-Pro. Detailed per-subject results for each benchmark are provided in Appendix C.

4.5 Next-Token Reasoning Pattern Analysis

We analyze the differences in reasoning patterns between next-token reasoning and explicit problem solving. Following previous studies [WYZ$^+$25, GCD$^+$25], we statistically measure the proportion of model responses containing reasoning-indicative keywords (e.g., "*break down*", "*alternatively*").[5]

Our analysis compares the thought processes of two models on the OmniMATH datasets, i.e., R1-Distill-Qwen-14B for problem solving, and RPT-14B for next-token reasoning, based on 200 sampled responses from each model. We categorize reasoning patterns into six types: transition

[5] The keywords are listed in Appendix E.

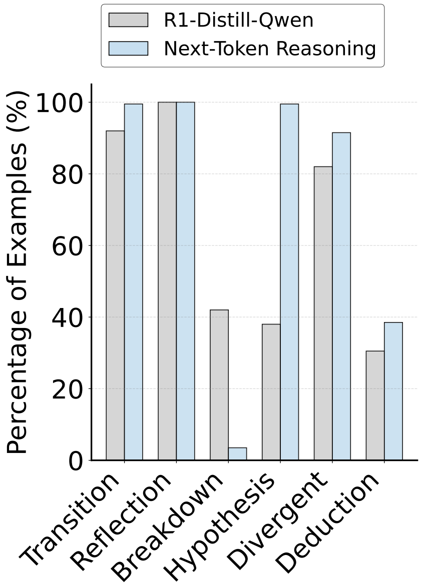


Figure 6: Reasoning pattern statistics of R1-Distill-Qwen-14B employed for problem solving and RPT-14B employed for next-token reasoning.

Thinking Process of RPT-14B
Okay, I need to figure out what the next token ... Let's think about what would logically come next in a text about calculating vector magnitude. The flow so far is introducing the topic, explaining the formulas for 2D and 3D vectors, and now moving on to how to actually perform the calculation. Since the user mentioned "go over some..." the next part is likely going to be ... Alternatively, it could be ... Common phrases after "we can go over some" could be "examples," "steps," "methods," etc. Looking at similar contexts, in educational materials, after explaining formulas, they often provide "steps" or "examples." But perhaps, given the user started a new section, it might be a new paragraph... Wait, perhaps in the original, the next part was an example or an explanation. Let's think about standard educational content. So, the entire text might continue as: ... Alternatively, perhaps the next line is a new section. Wait, another approach: ... However, the original text could have a space before the next word... Wait, perhaps looking at how the initial document is structured. It's using markdown with headers:...

Table 4: Example of reasoning patterns of RPT.

(switching strategies), reflection (self-checking), breakdown (decomposing the problem), hypothesis (proposing and verifying assumptions), divergent thinking (exploring possibilities), and deduction (logical inference).

As illustrated in Figure 6, RPT-14B's next-token reasoning process is markedly different from the problem-solving of R1-Distill-Qwen-14B, exhibiting a 161.8% greater use of the hypothesis pattern and a 26.2% greater use of the deduction pattern. In contrast, the problem-solving process relies more heavily on the breakdown pattern, highlighting that next-token reasoning elicits an inferential process qualitatively different from structured problem-solving.

We also provide an example of reasoning patterns in Table 4. The example reveals that the model engages in a deliberative process, not a simple pattern match. It analyzes the broader semantic context ("*calculating vector magnitude*"), identifies pivotal phrases ("*go over some...*"), and then brainstorms and weighs multiple plausible continuations. This involves hypothesis generation ("*the next part* *is likely going to be...*"), consideration of alternatives ("*Alternatively, it could be...*"), and reflection on structural cues ("*markdown with headers*") and even fine-grained token-level details ("*could have a space*"). This multi-faceted reasoning, encompassing both high-level semantic understanding and low-level textual features, demonstrates the model's effort to deduce the next token through a reasoned exploration, aligning with the goals of RPT to cultivate deeper understanding beyond superficial correlations. More examples are provided in Appendix F.

5 Related Work

Scaling Paradigms of Large Language Models The advancements of large language models have been driven by two primary scaling dimensions: training-time compute [KMH+20, HBM+22a] and test-time compute [ZLS+25a]. Training-time scaling substantially increases model parameters and training data, using next-token prediction as the pre-training task. Meanwhile, test-time scaling [JKL+24] trades extended inference compute to improve the reasoning capabilities of large language models. Going beyond existing scaling paradigms, RPT uniquely integrates the above principles, framing each next-token prediction as a reasoning task.

Reinforcement Learning for Large Language Models Reinforcement learning (RL) has played a crucial role in the post-training stage of large language models. Reinforcement learning from human feedback [OWJ+22] fine-tunes pre-trained language models on human preference data to improve alignment. Beyond alignment, large-scale RL has also been adopted to enhance the reasoning capabilities of language models [JKL+24, GYZ+25]. [ZHS+24] is the most relevant work, which encourages language models to generate helpful rationales for next-token prediction. The helpfulness-based reward tends to be hacked by repeating the target token in the generated rationale, where the shortcut potentially harms the model. In contrast, we use next-token prediction correctness as a rule-based reward signal to minimize reward hacking.

6 Conclusion and Future Work

We introduce reinforcement pre-training (RPT), a novel paradigm for pre-training large language models. By framing next-token prediction as a verifiable reasoning task and applying reinforcement learning with correctness-based rewards, RPT allows LLMs to leverage extended computation during pre-training to build stronger foundational reasoning capabilities. Our experiments demonstrate that RPT improves next-token prediction, enhances performance on mathematical and general reasoning benchmarks in zero-shot settings, and provides a better starting point for further RL fine-tuning. RPT offers a promising new direction for developing more capable and generally intelligent LLMs by fundamentally rethinking the pre-training objective itself.

While promising, this initial exploration of RPT has certain limitations. Our experiments are primarily conducted using a 14B parameter model. Although the RPT methodology is designed to be general, the current pre-training corpus predominantly consists of mathematical documents; future work will explore its efficacy on broader, general-domain text. Furthermore, RPT training is initialized from a reasoning model; investigating RPT training from a standard base language model would provide further insights into its foundational impact.

The work can be advanced from the following perspectives. We would like to scale up the training corpus, including data size, and domain coverage. Large-scale general Internet data can be utilized during reinforcement pre-training. We will also scale up training compute to push the frontier. Moreover, we can establish scaling laws for reinforcement pre-training to guide the scaling of large language models. Additionally, we are interested in integrating hybrid thinking [JWH+25] with RPT to enable fine-grained adaptive thinking by adaptively triggering next-token reasoning.

Acknowledgement

We extend our gratitude to Yuting Jiang for maintaining the GPU cluster. We also thank Zewen Chi and Yang Wang for technical support during the development of the RL infrastructure on the MI300 GPUs. We implement training based on verl [SZY+24].

References

[AAA^{+}23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[DYM^{+}25] Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.

[GCD^{+}25] Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Reward reasoning model, 2025.

[GSY^{+}24] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal Olympiad level mathematic benchmark for large language models. *ArXiv*, abs/2410.07985, 2024.

[GYZ^{+}25] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[HBB^{+}20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[HBM^{+}22a] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[HBM^{+}22b] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. Training compute-optimal large language models. *ArXiv*, abs/2203.15556, 2022.

[HDW^{+}25] Yaru Hao, Li Dong, Xun Wu, Shaohan Huang, Zewen Chi, and Furu Wei. On-policy RL with optimal reward baseline, 2025.

[HLL^{+}25] Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

[JKL^{+}24] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[JWH^{+}25] Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. Think only when you need with large hybrid-reasoning models, 2025.

[KMH^{+}20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[LeC16] Yann LeCun. Predictive learning. Advances in Neural Information Processing Systems, 2016.

[LMP+25] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025.

[MLJ+25] Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025.

[OWJ+22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[SZY+24] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[WYZ+25] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025.

[YZZ+25] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, et al. DAPO: An open-source LLM reinforcement learning system at scale. *ArXiv*, abs/2503.14476, 2025.

[ZHS+24] Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

[ZLS+25a] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

[ZLS+25b] Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. *arXiv preprint arXiv:2505.21493*, 2025.

A Design Choices of Reward

We have also investigated several alternative reward functions to assess their impact on reinforcement pre-training, in addition to the reward mechanism described in Section 3, i.e., prefix matching reward.

One variation is first-token matching. In this setup, the reward reflects only whether the first token of the model prediction y_t^i matches the ground-truth next token x_t, ignoring all tokens after the first in the prediction. Another alternative explores a 'dense reward' scheme. Here, correctly predicted next tokens (i.e., $y_t^i[0] = x_t$) receive a full reward (e.g., 1). For incorrect predictions ($y_t^i[0] \neq x_t$), the reward is a positive, smaller value, specifically the language model probability of generating that particular incorrect token, $P(y_t^i[0] \mid x_{<t}; \theta)$. This provides a denser feedback signal than binary rewards. A third design is a conditional application of this dense reward structure. The dense reward (1 for correct, $P(y_t^i \mid x_{<t}; \theta)$ for incorrect) is used as described above, but only for training instances (groups of rollouts for a given prefix $x_{<t}$) where at least one of the G sampled rollouts correctly predicted the next token. If all G rollouts in a group are incorrect, a different reward scheme (e.g., zero reward for all, or a uniform small penalty) will be applied.

Our experiments indicate that the alternative reward designs generally achieved performance comparable to the prefix matching reward. This suggests that the reinforcement pre-training framework is relatively robust to these particular modifications in the reward signal, and its core benefits may not be overly sensitive to these specific choices, at least within the scope of variations tested.

B Hyperparameters Used for Reinforcement Pre-Training

Table 5 presents the detailed hyperparameters for reinforcement pre-training in Section 4. We follow the setting of exact on-policy reinforcement learning [HDW+25] and set the entropy loss coefficient to 0.

Params	Values
Actor gradient clip	0.2
Batch size	256
PPO mini batch size	256
Rollout number	8
Learning rate	10^{-6}
Adam β	(0.9, 0.999)
Weight decay	0.01
Sampling temperature	0.8
Max prompt length	4096
Max response length	8192
Entropy loss coefficient	0

Table 5: Hyperparamters used for reinforcement pre-training in Section 4.

C Detailed Results on End Tasks

Table 6 and Table 7 present a detailed per-category performance across the general end task benchmarks. Notably, the performance of R1-Distill-Qwen-14B is evaluated in two different manner: standard next-token prediction and reasoning-based answer prediction (indicated as '+ think'). The RPT-14B model demonstrates superior performance compared to R1-Distill-Qwen-14B and R1-Distill-Qwen-32B.

D Impact of Prompt Templates

We explore the impact of various prompt templates on the initial next-token reasoning performance. Table 10 shows seven template variants. The templates use different phrasing of instructions and wrap the context in various formats.

	Agron.	Econ.	Educ.	Engin.	Hist.	Law	L.&A.	Manag.	Med.	Mil. Sci.	Phil.	Sci.	Sociol.	Overall
Standard next-token prediction mode														
R1-Distill-Qwen-14B	30.0	38.0	32.0	31.0	24.5	26.0	28.5	39.0	35.5	36.0	37.0	24.0	30.1	32.0
R1-Distill-Qwen-32B	32.5	39.5	43.0	34.0	29.5	31.0	28.5	41.5	43.5	49.0	44.5	29.5	38.5	37.2
Reasoning mode														
R1-Distill-Qwen-14B	31.0	41.0	32.0	34.5	29.0	31.0	29.5	39.5	38.5	39.5	44.0	41.5	39.2	36.1
RPT-14B	35.0	40.0	41.5	40.5	30.5	32.0	29.0	36.0	44.5	41.0	49.0	47.0	42.0	39.0

Table 6: Detailed zero-shot performance on SuperGPQA.

	Bio.	Bus.	Chem.	CS	Econ.	Engin.	Heal.	Hist.	Law	Math	Other	Phil.	Phys.	Psych.	Overall
Standard next-token prediction mode															
R1-Distill-Qwen-14B	72.5	42.5	34.0	46.5	58.0	44.0	57.5	54.0	37.0	36.5	50.0	48.5	34.5	62.0	48.4
R1-Distill-Qwen-32B	82.5	46.0	39.0	55.5	74.0	52.0	68.0	62.5	47.0	46.0	54.0	53.5	42.5	68.5	56.5
Reasoning mode															
R1-Distill-Qwen14B	85.0	65.5	74.5	75.0	81.5	52.0	70.0	61.5	42.0	86.0	65.0	62.5	80.0	64.5	68.9
RPT-14B	84.5	72.0	77.5	76.0	78.5	53.5	74.0	63.0	44.5	91.5	66.0	63.5	82.5	68.0	71.1

Table 7: Detailed zero-shot performance on MMLU-Pro.

Prompt Template	Random@1 (%)	Pass@8 (%)
v0	3.0	8.5
v1	5.7	11.0
v2	5.7	16.0
v3	5.3	11.0
v4	4.0	9.0
v5	4.4	12.5
v6	6.0	19.0

Table 8: Impact of prompt templates.

Pattern Group	**Keywords**
Transition	alternatively, think differently
Reflection	wait, initial answer, original answer, looking back, thought process
Breakdown	break down, break this down
Hypothesis	probably, something like
Divergent Thinking	etc., or something, either, sometimes it refers, otherwise, exploring, options
Deduction	summarize, conclusion, conclude, finally, logically, consequently

Table 9: Pattern groups and keywords applied in Section 4.5.

As presented in Table 8, clear prompts significantly improve the correctness of the initial performance. Notice that the reinforcement pre-training experiments in Section 4 used the 'v0' prompt template. We leave prompt engineering based on other template variants for future work, which tends to improve the final performance.

E Keywords for Reasoning Pattern Analysis

Table 9 presents the pattern groups and keywords applied in reasoning pattern analysis.

Version	Prompt Content
v0	Complete the given text under '### Context' by predicting the next token, and wrap it in '\boxed{}'. Please reason step by step to find the most probable next token as the final answer, and enclose it in \boxed{} (note: the token may begin with a space, e.g., \boxed{ para} or \boxed{ =}; do **not** use \text{}). ### Context {prompt_content}
v1	Complete the given text under ### Context by predicting the next token, and wrap it in \\boxed{}. Please reason step by step to find the most probable next token as the final prediction, and enclose it in \boxed{} (note: the token may begin with a space, e.g., \boxed{ para} or \boxed{ =}; do **not** use \text{}). ### Context ```{prompt_content}```.
v2	You are a helpful assistant, good at predicting the next token for a given context. Now, please complete the given text under ### Context by predicting the next token, and wrap it in \\boxed{}. Please reason step by step to find the most probable next token, and enclose it in \boxed{} (note: the token may begin with a space, e.g., \boxed{ para} or \boxed{ +=}; do **not** use \text{}). ### Context ```{prompt_content}```.
v3	Complete the given text under ### Context by predicting the next token, list multiple potential tokens and select the most probable one as the final answer. Wrap your final answer in \boxed{} (note: the token may begin with a space, e.g., \boxed{ para} or \boxed{ =}; do **not** use \text{}). ### Context ```{prompt_content}```
v4	Complete the given text under ### Context by predicting the next token, and wrap it in \boxed{}. Please reason step by step to find the most probable next token as the final answer, and enclose it in \boxed{}. Some examples: ### Context \n \n ```...(some omitted)...Matching calculations with 1990 valid combinations indicates the minimum value of \(b \) that fits all pre-requisites and restrictions for triangle formation and symmetry generates the efficient outcome: \n \n \[\n \boxed{1991^2} \n \] \n \nIn``` The next token is \boxed{ this} ### Context \n \n ```...Thus $2^{A}=\left(2^{a}\right)^{2}\left(2^{3}\right)=``` The next token is \boxed{9} ### Context \n \n ```..., numerical exploration shows``` The next token is \boxed{:\n} Now, the context is: ### Context \n \n ```{prompt_content}```.
v5	Complete the given text under ### Context by predicting the next token, and wrap it in \boxed{}. Please reason step by step to find the most probable next token as the final answer, and enclose it in \boxed{} (note: the token may begin with a space, e.g., \boxed{ para} or \boxed{ =}; do **not** use \text{}). ### Context ```{prompt_content}```.
v6	Complete the given text wrapped in ``` and ``` by predicting the next token, list multiple potential tokens and select the most probable one as the final prediction. Wrap your final prediction in \boxed{} (note: the token may begin with a space, e.g., \boxed{ para} or \boxed{ =}; do **not** use \text{}). The context is: ```{prompt_content}```, now please predict the next token.

Table 10: Seven prompt templates for the next-token reasoning task.

F Case Studies

To provide qualitative insights into the behavior of models trained with RPT, we present some cases of next-token reasoning in Table 11.

Case 1
Okay, I need to figure out what the next token is in the given text under the Context section. The user provided a partially completed text about integers, comparing and ordering them ... First, I’ll analyze the context provided ... Wait, the user wrote 6 x 4 = 24, 20 > 18 x, so the next token is probably the second fraction ... One way to find a common denominator is to multiply the two original denominators ... Alternatively, it could be ... Wait, but 18 x 24? That doesn’t make sense. Alternatively, perhaps it’s 18 x 4 = 72 or something... Another approach: perhaps the next token is ... Let me see: if the two fractions are 5/3 and 4/6. 5/3 is greater than 4/6 because 5/3 is 1.666..., and 4/6 is 0.666..., to compare them, find a common denominator... Alternatively, perhaps the next token is a backslash or something in LaTeX. Wait, but in the Markdown, the user presented it as: Writing Equivalent Fractions ... But wait, in the user’s provided context, it’s ... Maybe the next token is another multiplication symbol, but that seems unlikely. Alternatively, perhaps it’s just the end of the thought. But given the context, I think that the next token is either a space followed by a ’2’ or a period or something else ... Wait, maybe another example. If the fractions were 5/6 and 4/3 ... So, after ’x’ there’s a space, then ’4’.\n</think>\n\nSo, the next token is ’ 4’.\n\n\boxed{ 4}
Case 2
Okay, I need to figure out what the next token is in the given text under the Context section. Using the integral test, for a set ... Maybe the original answer was just a partial sentence. Let’s think again. The question is to prove convergence of the series, the answer uses the integral test, which is the standard approach ... Alternatively, maybe the original answer was longer, but got cut off, so I need to predict what comes next. Maybe the next token is ... So the next token is a space and then something. Alternatively, perhaps it’s a continuation of the same sentence ... Therefore, by the integral test, the series converges when $p > 1$. No, perhaps it’s better to see what the original answer was supposed to be. The correct full proof would be: Using the integral test, for a set ... But the user instruction is to provide the next token after the given text ... Alternatively, looking at the way the answer was written, perhaps ... I think the next token is ’$’.\n</think>\n\nSo, the next token is ’$’.\n\n\boxed{$}
Case 3
Okay, I need to figure out what the next token is in the given text under the Context section. The text so far is: Call the large circle’s center ... Scaling by a factor... Perhaps the next part is the ratio of the radii? Or maybe something else. Wait, the scaling factor is probably such that the smaller circle is scaled up to match the larger circle or something ... So, in coordinates, perhaps O1 is at (0,0), O2 is at (41,0). The small circle has radius 13, so the points where O1O2 intersects the small circle are at O2 plus or minus 13 units towards O1 ... So, point A is on the small circle, further away from O1 than O2, right. Because if you go from O2 towards O1, 13 units, that would be ... So, the next token after ’Scale the small circle by a factor of’ would probably be the number 30/13 ... Wait, I’m a bit confused. Alternatively, maybe it’s presented as a fraction ... Scale the small circle by a factor of. So, the actual next token is the next word or symbol ... Wait, but in the context given, the user just wrote ... Alternatively, perhaps in the text it’s written as ... So, in the given text, the next token after ’Scale the small circle by a factor of’ is space.\n</think>\n\nThe next token is ’ ’.\n\n\boxed{ }

Table 11: Three examples of reasoning patterns of RPT-14B.