Decision-Centric Reinforcement Learning over Promising Tokens for Large Language Model Reasoning

Anonymous ACL submission

Abstract

Reinforcement learning (RL) for large language models (LLMs) is commonly formulated as token-level policy optimization over the full vocabulary. This fixed action space is tens of thousands of tokens wide, although only a small state-dependent subset typically represents plausible next reasoning moves. Existing decoding strategies such as Top-k or nucleus sampling exploit this structure during rollout, but standard RL objectives still compute policy ratios and gradients with respect to the full-vocabulary policy, creating a mismatch between the space where trajectories are collected and the space where the policy is optimized. In this work, we study LLM generation as decision-making over a fixed global action space with a dynamic local decision set. We introduce Reinforcement Learning with Promising Tokens (RLPT), which uses the behavior policy's own distribution to construct a promising token set at each state and defines a masked policy consistently for both rollout and optimization. This simple policy reparameterization turns token-level RL from full-vocabulary fitting into action selection among plausible reasoning candidates. Across mathematical reasoning, code, and domain QA tasks, RLPT improves GRPO and DAPO, with multi-seed results on Math-17k, MATH500, AIME24, and AIME25 showing improved accuracy and reduced training variability. Controlled ablations further show that the gains come from aligning rollout and optimization supports, rather than from decoding-time truncation alone.

1 Introduction

Reinforcement learning (RL) has become a central tool for improving large language models (LLMs) on tasks with verifiable or preference-based rewards, including mathematical reasoning, code generation, and instruction following (Pang et al., 2024; Mu et al., 2024; Shao et al., 2024). In the standard formulation, an LLM is treated as a token-level policy: at each step, the policy chooses the next token from a fixed vocabulary. This makes LLM RL a sequential decision problem over a global discrete action space that often contains more than 50,000 actions.

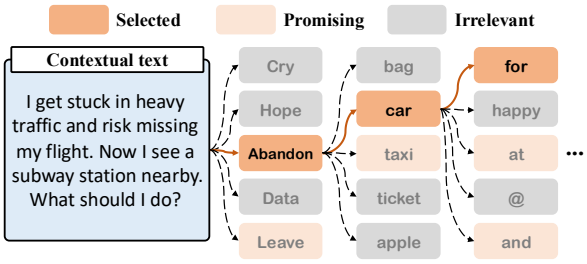


Figure 1: Decision-centric RL over promising tokens. The global action space is the fixed vocabulary, while each state exposes a compact set of plausible next-token actions. RLPT optimizes the policy inside this local decision set instead of treating the full vocabulary as equally relevant to every update.

This full-vocabulary view is formally simple but operationally inefficient. Most tokens are not meaningful actions for a given state: they are syntactically impossible, semantically unrelated, or irrelevant to the next reasoning move. Decoding methods such as Top-k sampling (Hopkins et al., 2023) and nucleus sampling (Holtzman et al., 2020) already exploit this fact during rollout by truncating the sampling distribution. However, standard RL training pipelines usually optimize policy ratios, KL terms, and gradients with respect to the original full-vocabulary policy (Sheng et al., 2024; Havrilla et al., 2023). As a result, the model collects trajectories in a restricted effective support but updates the policy as if every vocabulary item were an active action. This support mismatch weakens the decision-making interpretation of token-level RL and can waste optimization signal on tail tokens that were never available during rollout.

We argue that LLM RL could be formulated as decision-making over a fixed global action space with a dynamic local decision set. The pretrained

model already encodes strong linguistic and semantic priors: for most reasoning states, it assigns high probability to a small set of plausible continuations. Our empirical analysis shows that successful reasoning trajectories are highly concentrated in this high-probability subset. This suggests a different role for RL. *Instead of relearning basic next-token plausibility over the whole vocabulary, RL should learn to select the best reasoning move among the tokens that the base model already considers plausible*. Building on this insight, we propose Reinforcement Learning with Promising Tokens (RLPT) method. At each generation step, RLPT constructs a state-dependent promising token set from the behavior policy and defines a masked policy over this set. The same stored mask is used during rollout and policy optimization, so the behavior policy, current policy ratio, and gradient update are all computed on the same support. In this sense, RLPT is not merely a decoding heuristic: it lifts decoding-time truncation into an RL-consistent policy parameterization. The resulting update changes the learning problem from full-vocabulary token fitting to constrained action selection among plausible reasoning candidates.

Our contributions are summarized as follows. First, we formulate token-level LLM RL as decision-centric optimization over state-dependent promising action sets, explaining why full-vocabulary updates can waste learning signal on irrelevant tail tokens. Second, we propose RLPT, an algorithm-agnostic support-alignment framework that makes rollout and optimization operate on the same token subset, reducing support mismatch without adding an auxiliary model. Third, we show that successful reasoning trajectories concentrate in compact token sets and that RLPT improves GRPO and DAPO with better stability and controlled gains beyond decoding-time truncation (Shao et al., 2024; Yu et al., 2025).

2 Preliminary

RL for LLM optimization. We consider a standard language modeling task in which an LLM serves as a policy π_θ parameterized by θ. Given a context input (prompt or question) x, the model generates a response y composed of a sequence of tokens $y = (y_1, y_2, \ldots, y_T)$, where each token y_t belongs to a fixed vocabulary space $\mathcal{V}$. The probability of generating the entire sequence is given by the chain rule: $\pi_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t})$, where $y_{<t}$ denotes the partial sequence generated prior to step t.

To align the LLM with human preferences or specific logical rules, we formulate the generation process as a Markov Decision Process (MDP (Puterman, 1994)) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$:

- State space $\mathcal{S}$: The state s_t at time step t consists of the prompt and the history of generated tokens, i.e., $s_t = (x, y_1, \ldots, y_{t-1})$;
- Action space $\mathcal{A}$: the space of the tokens $\mathcal{V}$;
- Transition $\mathcal{T}$: $s_{t+1} = s_t \cup \{a_t\}$;
- Reward function $\mathcal{R}$: a score $r(s_t, a_t)$ that reflects the quality of the generated answer to the question, typically given at the end of the sequence (e.g., $r_T = R(x, y)$) based on correctness verification, with intermediate rewards being zero;
- Discount factor γ: We typically set $\gamma = 1$ for episodic generation tasks.

The goal of RL training is to find an optimal policy π_θ that maximizes the expected cumulative reward over the prompt distribution $\mathcal{D}$:

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(\cdot|x)}[R(x, y)],$$

where $R(x, y)$ represents the reward (e.g., answer correctness) received at the end of the episode.

Decision sets inside a fixed token action space. The global action space of the MDP is fixed as the vocabulary $\mathcal{A} = \mathcal{V}$. However, the effective decisions available at a particular state are highly state-dependent. We denote a local decision set by $\mathcal{P}_t \subset \mathcal{V}$, where $\mathcal{P}_t$ contains the next-token actions that are plausible under the behavior policy at state s_t. This distinction is central to our formulation: the vocabulary remains the fixed action space, while RL optimization is carried out on a compact state-dependent subset that preserves the actions available during data collection.

Given a behavior policy π_{old}, we define a masked policy $\tilde{\pi}$ supported on $\mathcal{P}_t$:

$$\tilde{\pi}(a_t|s_t, \mathcal{P}_t) = \frac{\pi(a_t|s_t)\mathbb{I}(a_t \in \mathcal{P}_t)}{\sum_{v \in \mathcal{P}_t} \pi(v|s_t)}. \quad (1)$$

The role of RLPT is to construct $\mathcal{P}_t$ from the behavior policy and use the corresponding masked policy consistently during both trajectory rollout and policy update.

3 Analyzing the Potential for Decision Making over Promising Tokens

Before introducing our method, we answer an important question: *Is the subspace of promising tokens sufficient for LLMs to solve the problem?* Intuitively, while the vocabulary of an LLM is vast ($|\mathcal{V}| > 50k$), the tokens required for a logically correct step are often concentrated within a small subset of high-probability candidates. This is consistent with the broad success of truncated decoding, but here we examine whether the same structure can support RL optimization rather than only inference-time sampling. In this section, we conduct an empirical analysis using the Qwen series models (Bai et al., 2023) to answer this question. We first formally define the concept of promising tokens.

Definition 3.1 (Promising Tokens). *Given the current state s_t and the policy distribution $\pi(\cdot|s_t)$, let $rank(v \mid s_t)$ denote the rank of a token $v \in \mathcal{V}$ when sorted in descending order of its probability. The set of promising tokens, denoted as $\mathcal{P}_t$, is defined as the top-K candidates:*

$$\mathcal{P}_t = \{v \in \mathcal{V} \mid rank(v \mid s_t) \leq K\}. \quad (2)$$

Consequently, $|\mathcal{P}_t| = K$ and $\mathcal{P}_t \subset \mathcal{V}$.

This definition constructs a focused action space, filtering out the long tail of contextually irrelevant tokens. To verify the coverage of $\mathcal{P}_t$, we analyze the token ranks of *successful trajectories* from two distinct perspectives: the ground truth answer and the model's own successful answer.

Setup. We employ `Qwen3-8B` and `Qwen3-32B` as base models. We use three diverse datasets: GSM8K (Cobbe et al., 2021) for mathematics, HumanEval (Chen, 2021) for coding, and AlpacaEval (Taori et al., 2023) for general instruction following. We further add Math-17k (Yu et al., 2025) to test whether the concentration pattern remains visible on a harder reasoning corpus used in our RL experiments. For each step t in a successful trajectory y, we compute the rank of the target token y_t within the base model's predicted distribution $\pi_{\text{base}}(\cdot|y_{<t})$. We define the *top-$K$ coverage rate* as the percentage of tokens in these trajectories that satisfy $y_t \in \mathcal{P}_t$.

3.1 Analysis of Ground-truth Solution

Observation 1: ***The labeled solution is effectively contained within the promising token subspace.***

Table 1: Top-K coverage rate (%) of successful trajectory tokens from labeled solutions. Math, Code, and General denote GSM8K, HumanEval, and AlpacaEval dataset, respectively.

Metric	Qwen3-8B			Qwen3-32B		
	Math	Code	General	Math	Code	General
Top-2	91.5	92.0	83.0	92.1	92.7	83.8
Top-4	95.3	95.0	91.0	96.1	95.3	91.7
Top-8	97.4	96.5	95.4	98.1	96.7	95.9
Top-16	98.6	97.2	97.6	99.0	97.5	98.0
Top-32	99.2	97.7	98.8	99.5	98.0	99.0

Table 2: Top-K coverage rate (%) of successful trajectory tokens on Math-17k using Qwen3-8B.

Task	Top-2	Top-4	Top-8	Top-16
Math-17k	95.32	98.15	99.34	99.77

We first investigate whether the ground truth answers (labeled solutions) fall within the promising tokens predicted by the base model. This measures the *feasibility* of finding the optimal policy within the promising token space $\mathcal{P}_t$. As shown in Tab. 1, the correct answer tokens are highly concentrated in the head of the distribution. For `Qwen3-32B`, over 99.5% of the ground truth tokens fall within the Top-32 candidates for Math tasks. Even for the smaller 8B model, the Top-32 coverage consistently exceeds 97% across all domains. Besides, qualitative inspection reveals that the few outliers (i.e., tokens outside $\mathcal{P}_t$) mainly appear at the very beginning of the answer generation or consist of infrequent symbols, **typically having negligible impact on the core reasoning logic.** This observation suggests that many solution tokens are already statistically prioritized by the pre-trained model. Therefore, pruning the vocabulary to $\mathcal{P}_t$ can preserve much of the useful solution space in the reasoning settings we study, although it does not provide a hard guarantee for every task or model.

3.2 Analysis of Model-generated Solution

Observation 2: ***The model's intrinsic answering capabilities do not rely much on the long tail of the vocabulary.***

While previous experiment confirms that the labeled solution is largely covered by promising tokens, one might worry that the model needs to explore creative low-probability tokens to find its own path to the solution. To address this, we sample multiple answers from the base model and analyze only those that reach the correct final answer.

Tab. 3 reports the rank of tokens in these *model-generated successful paths*. The results show that for Math and Code tasks in this analysis, the tokens in correct solutions are almost entirely located within the Top-8 candidates. This indicates that when the model successfully reasons, it relies mostly on high-likelihood tokens. The long-tail tokens contribute little to these observed correct reasoning chains. Consequently, the challenge of RL may not be expanding the search to the full vocabulary, but to learn to decide the correct move from the promising tokens.

Table 3: Top-K coverage rate (%) of successful trajectory tokens from model-generated solution.

Metric	Qwen3-8B			Qwen3-32B		
	Math	Code	General	Math	Code	General
Top-2	98.9	98.1	93.2	92.1	97.8	92.4
Top-4	99.9	99.6	98.0	100	100	100
Top-8	100	100	100	100	100	100

4 Method

In this section, we present RLPT, a decision-centric RL framework for LLM reasoning. The global action space remains the fixed vocabulary $\mathcal{V}$, but each state s_t induces a compact local decision set $\mathcal{P}_t$. RLPT first constructs $\mathcal{P}_t$ from the behavior policy's semantic prior, then defines a masked policy over this set for both sampling and optimization. This turns token-level RL from full-vocabulary fitting into action selection among plausible next reasoning moves.

4.1 Promising Token Construction

At each time step t, the behavior policy π_{old} computes a distribution over the vocabulary $\mathcal{V}$ given state $s_t = (x, y_{<t})$. We define the promising token set $\mathcal{P}_t$ as the Top-K tokens under $\pi_{\text{old}}(\cdot|s_t)$. The corresponding mask $M_t \in \{0,1\}^{|\mathcal{V}|}$ is

$$M_t[v] = \mathbb{I}(v \in \mathcal{P}_t) = \begin{cases} 1, & \text{if } v \in \mathcal{P}_t, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Unlike invalid-action masking with externally specified rules, this mask is induced by the LLM itself and is state-dependent. It should not be interpreted as a claim that all masked tokens are globally invalid. Instead, RLPT uses the behavior policy to identify the compact subset that is most relevant for the current decision.

4.2 Masked Policy for Rollout and Optimization

Given a mask M_t, RLPT defines a masked policy by renormalizing probability mass over $\mathcal{P}_t$:

$$\tilde{\pi}(a_t \mid s_t, M_t) = \frac{\pi(a_t \mid s_t) \cdot M_t[a_t]}{\sum_{v \in \mathcal{V}} \pi(v \mid s_t) \cdot M_t[v]}. \tag{4}$$

For rollout, actions are sampled from $\tilde{\pi}_{\text{old}}(\cdot|s_t, M_t)$. This is computationally equivalent to Top-K sampling, but RLPT stores the mask M_t together with each trajectory so that the same support is available during training.

For optimization, the current policy is evaluated under the stored mask rather than under the full-vocabulary distribution:

$$\tilde{\pi}_\theta(a_t \mid s_t, M_t) = \text{Softmax}\big(\text{Logits}_\theta(s_t) + (1 - M_t) \cdot (-\infty)\big)_{a_t}. \tag{5}$$

The mask is recomputed whenever a new rollout batch is collected, but it remains fixed while optimizing that batch. This avoids a support drift problem: if the mask were recomputed from the evolving current policy inside the update loop, a sampled action could fall outside the new support, making the policy ratio ill-defined. Storing the behavior-policy mask keeps data collection and policy improvement aligned.

4.3 Integration with RL Algorithms

RLPT is algorithm-agnostic because it only replaces the policy distribution used in ratios and regularization terms. We illustrate the integration with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Given trajectories sampled from the masked behavior policy, the RLPT-GRPO objective is

$$\mathcal{J}_{\text{RLPT}}(\theta) = \mathbb{E}_{Traj \sim \tilde{\pi}_{\text{old}}}\left[\frac{1}{T}\sum_{t=1}^{T} \min\left(\frac{\tilde{\pi}_\theta(a_t|s_t, M_t)}{\tilde{\pi}_{\text{old}}(a_t|s_t, M_t)} A_t, \text{clip}\left(\frac{\tilde{\pi}_\theta(a_t|s_t, M_t)}{\tilde{\pi}_{\text{old}}(a_t|s_t, M_t)}, 1-\epsilon, 1+\epsilon\right) A_t\right)\right], \tag{6}$$

where A_t is the advantage computed from rewards. The key distinction from standard GRPO is that the ratio is computed within the same promising-token support used for rollout. When KL or entropy regularization is used, we apply the same masked policy consistently to those terms as well.

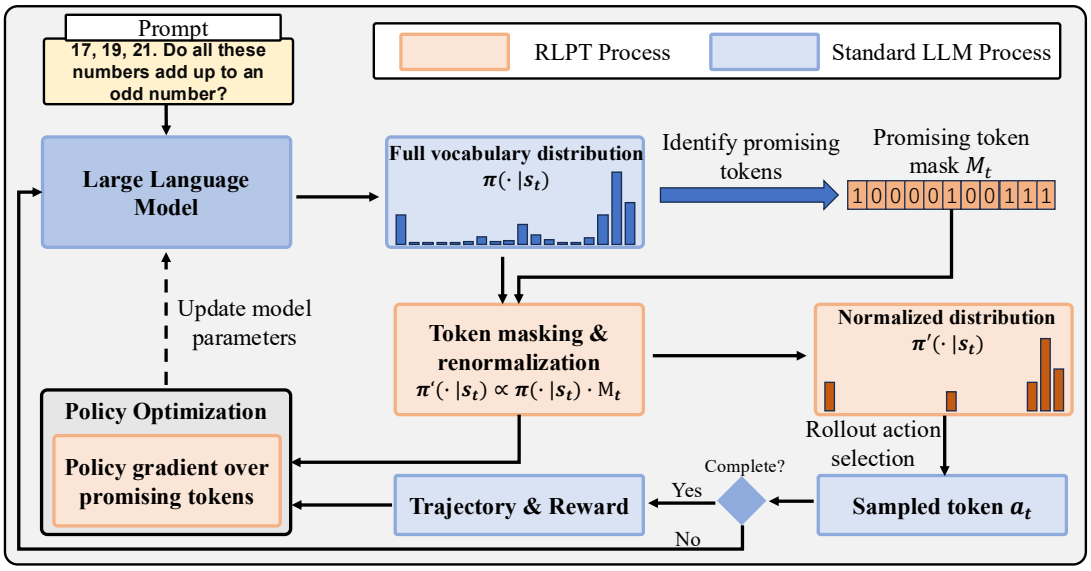


Figure 2: The overall framework of RLPT. For each rollout batch, the behavior policy constructs and stores a promising token set at every state, which is then used to compute rollout probabilities and optimization ratios.

4.4 Analytical View: Tail-Gradient Removal

We provide an analytical view of why constrained support can improve optimization stability. For a softmax policy over the full vocabulary, the gradient of the log-probability with respect to the logit vector z is $\nabla_z \log \pi(a|s) = \mathbf{e}_a - \pi$. Therefore, every update changes not only the sampled action logit but also all non-sampled token logits through the normalization term. In a large vocabulary, many of these components correspond to tail tokens that were not available under the rollout distribution.

Let $\mathcal{T}_t = \mathcal{V} \setminus \mathcal{P}_t$ be the masked tail. Under RLPT, the gradient components associated with $\mathcal{T}_t$ are deterministically zero because the policy is defined only over $\mathcal{P}_t$. This does not by itself prove that the total variance over the renormalized support is always lower without assumptions. Instead, it shows a concrete mechanism: RLPT removes gradient fluctuations for tail logits that are outside the collected trajectory support. The empirical gradient statistics in Sec. 6.3 validate that this support restriction is associated with smoother training in our reasoning experiments.

Proposition 4.1 (Tail-gradient removal)**.** *For any stored mask M_t and any token $v \in \mathcal{T}_t$, the gradient of $\log \tilde{\pi}_\theta(a_t|s_t, M_t)$ with respect to the masked tail logit z_v is zero. In contrast, the full-vocabulary softmax generally assigns a non-zero gradient component to z_v through the normalization term.*

We present the proof in Appendix F.1. This result formalizes the local effect of promising-token masking; the overall optimization benefit is then evaluated empirically through accuracy, ablations, and gradient-norm stability.

5 Related Work

5.1 RL for LLM Reasoning

Reinforcement learning is widely used to align LLM behavior and improve reasoning with outcome or preference rewards. RLHF methods commonly optimize PPO-style objectives (Ouyang et al., 2022; Schulman et al., 2017), while DPO and related preference-optimization methods remove the explicit online RL loop (Rafailov et al., 2023; Azar et al., 2024; Ethayarajh et al., 2024). For reasoning, GRPO (Shao et al., 2024), RFT (Trung et al., 2024), DAPO (Yu et al., 2025), and recent process-reward or credit-aware variants (Yao et al., 2026; Zhang, 2026) improve mathematical and code performance using verifier rewards, process signals, or group-normalized advantages. These methods mainly differ in reward design, advantage estimation, or clipping strategies, while typically evaluating the token policy over the full vocabulary during optimization. RLPT is complementary: it changes the action support on which policy ratios and gradients are computed, while leaving the reward and advantage estimators unchanged.

5.2 Decoding-time Truncation and Support Alignment

Top-k sampling (Hopkins et al., 2023), nucleus sampling (Holtzman et al., 2020), and related truncation rules such as Min-p sampling (Nguyen et al.,

2024) reduce the effective vocabulary during inference. These methods suppress low-probability tail tokens and are often essential for coherent generation. Recent analyses of entropy dynamics further show that token uncertainty is central to the exploration-exploitation trade-off in RL (Wang et al., 2026). However, decoding truncation is usually implemented only as a rollout heuristic, while the subsequent RL objective still evaluates the original full-vocabulary policy (Sheng et al., 2024; Havrilla et al., 2023). RLPT builds on truncated decoding but changes its role: the truncated support becomes part of the policy definition used for both data collection and optimization. This support-aligned view is the key distinction between RLPT and decoding-only Top-k or Top-p baselines.

5.3 Action Masking and Token-level Credit

Large discrete action spaces are a longstanding challenge in RL. Prior work studies action embeddings, action elimination, hierarchical decompositions, and LLM-specific action decomposition to make exploration and optimization more tractable (Zhong et al., 2018; Nachum et al., 2018; Pang et al., 2023; Wen et al., 2024). Invalid-action masking is effective when domain rules identify illegal actions in advance (Huang and Ontañón, 2022), and rule-based token filters have also been used in constrained language tasks such as recommendation (Wang et al., 2025a). In open-ended LLM reasoning, however, token validity is rarely binary and depends on context, semantics, and model competence. RLPT therefore constructs the mask from the behavior policy itself rather than from an external invalidity oracle or an auxiliary selector. Recent LLM optimization studies also show that not all tokens contribute equally to learning. High-entropy minority tokens can act as reasoning forks that drive much of the RLVR gain (Wang et al., 2025b), and follow-up work studies token-level credit through entropy-regulated objectives, polarity-entropy decomposition, attention entropy, and action-token bottlenecks (Yu et al., 2026; He et al., 2026b; Li et al., 2026; He et al., 2026a). These methods focus on which tokens should receive learning signal. RLPT instead intervenes on the feasible action support: the policy is explicitly renormalized over a compact decision set, so both exploration and optimization operate on the same candidate actions. Appendix B gives a more detailed comparison.

6 Experiment

Evaluation goals. We evaluate RLPT around two claims. First, support-aligned optimization should improve reasoning performance over standard RL objectives. Second, the gain should come from aligning rollout and optimization supports, rather than from decoding-time truncation alone. We therefore report multi-seed math results as the main evidence, then use cross-domain results, controlled ablations, and training diagnostics to explain when and why the method helps.

6.1 Experimental Setting

Datasets. We focus primarily on mathematical reasoning, where token-level decisions have long-horizon consequences. We evaluate on Math-17k (Yu et al., 2025), MATH500, AIME24, and AIME25 (Hendrycks et al., 2021). To test whether the method can be plugged into broader reasoning-oriented RL pipelines, we also include OpenR1-Code (Hugging Face, 2025) and two telecom QA datasets, Datacom and Wireless.

Baselines and Implementation. Our main baseline is GRPO (Shao et al., 2024); we also evaluate DAPO (Yu et al., 2025) to test algorithm compatibility. Unless otherwise specified, experiments use Qwen3-8B. We include Qwen3-4B and Llama-3.1-8B-Instruct as model-scale and model-family checks. Experiments are implemented using MindSpeed-RL (Feng et al., 2025). Reward signals are provided by rule-based verifiers for Math, execution sandboxes for Code, and model-based evaluation with Deepseek-V3 (DeepSeek-AI, 2024) for Telecom. Unless stated otherwise, the promising set size is $K = 4$ and results are reported at 200 training steps. For Table 4, we report mean and standard deviation over 5 independent runs. Full hyperparameters are listed in Appendix G.4.

6.2 Main Results

Results on mathematical reasoning. As shown in Table 4, RLPT improves GRPO on all four benchmarks about mathematical reasoning. On Math-17k, it raises mean accuracy from 35.78% to 37.78% and reduces standard deviation from 1.36 to 0.61. On MATH500, a standard MATH test benchmark, RLPT improves GRPO by 3.48 absolute points. AIME24 and AIME25 show the same direction, with gains of 1.66 and 2.20 points. These results support the central claim that reasoning RL benefits from optimizing inside the same compact

Table 4: Multi-seed mathematical reasoning results. We report accuracy (%) as mean ± standard deviation over 5 independent runs.

Method	Math-17k	MATH500	AIME24	AIME25
No Training	24.96 ± 0.59	55.48 ± 0.48	20.14 ± 1.68	16.53 ± 2.29
GRPO	35.78 ± 1.36	56.64 ± 0.88	21.57 ± 2.01	19.73 ± 1.35
GRPO+RLPT	**37.78 ± 0.61**	**60.12 ± 1.78**	**23.23 ± 1.95**	**21.93 ± 1.04**

Table 5: Compatibility with additional algorithms and domains. These are single-run results, included to show plug-in behavior beyond the multi-seed math table.

Method	Math			Domain		Code	Average
	Math-17k	AIME24	AIME25	Datacom	Wireless	OpenR1-Code	
No Training	25.2	16.7	16.7	55.65	50.87	40.89	34.34
GRPO	34.7	20.0	**19.3**	50.87	52.17	40.18	36.20
GRPO+RLPT	**38.3**	**23.3**	18.0	**51.30**	**55.65**	**40.92**	**37.91**
DAPO	36.4	**20.7**	17.3	54.43	49.57	**39.87**	36.38
DAPO+RLPT	**39.7**	19.3	**20.7**	**54.78**	**50.43**	39.01	**37.32**

decision set used for rollout. The benefit is not lower wall-clock cost, since identifying Top-K tokens still requires vocabulary-level scoring; rather, the benefit is optimization-side focus.

Algorithm and domain compatibility. Table 5 broadens the evaluation to DAPO, code, and telecom QA. The improvements are strongest and most consistent on math, which we treat as the primary setting. Code and telecom gains are smaller and occasionally mixed, but the average scores improve for both GRPO and DAPO variants. This supports the intended use of RLPT as a support-alignment layer that can be attached to different RL objectives, while also clarifying that the empirical claim is strongest for reasoning tasks where compact next-token decision sets are likely to be useful.

6.3 Analysis on Training Process

Optimization stability. The previous results evaluate final accuracy. We further inspect whether support alignment changes the training process. Figures 3a and 3b show faster early improvement on Math-17k and track gradient norms during GRPO and GRPO+RLPT training. The gradient norm standard deviation decreases from 0.0032 to 0.0029, a relative reduction of approximately 9.4%. We use this as diagnostic evidence rather than as a standalone proof: it is consistent with the analytical view that masked optimization removes tail-logit gradient fluctuations.

Model transfer and selector baseline. Table 6 shows that the effect is not limited to a single Qwen model size. The improvement on Llama-3.1-8B-Instruct indicates that RLPT is tied to the support-aligned policy formulation rather than to a Qwen-specific distributional artifact. We also compare against an explicit selector policy that predicts a candidate token from context and token embeddings. As shown in Fig. 3c, this auxiliary selector underperforms RLPT, suggesting that reusing the LLM's own behavior distribution is more reliable than training a separate selector from scratch.

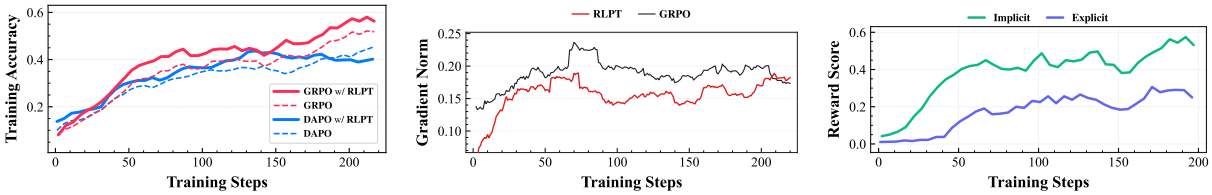
Table 6: Model-family and scale checks on Math-17k. Accuracy is reported in %.

Backbone	GRPO	GRPO+RLPT
Qwen3-4B	30.8	**32.1**
Qwen3-8B	34.7	**38.3**
Llama-3.1-8B-Instruct	33.2	**35.4**

Qualitative decision analysis. Finally, Table 7 gives a qualitative example of the decision-centric interpretation. The promising set contains plausible discourse and reasoning continuations while excluding most unrelated vocabulary items. RL still needs to choose the best continuation, but the choice is made inside a compact action set.

6.4 Ablation Study

Support alignment. Table 8 isolates the contribution of rollout masking and optimization masking. Truncated rollout alone improves over GRPO, con-



(a) Training reward (b) Gradient norm (c) Selector comparison

Figure 3: Training-process analysis and selector comparison on Math-17k.

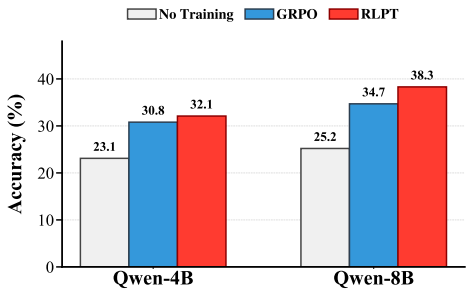


Figure 4: Performance on different model sizes.

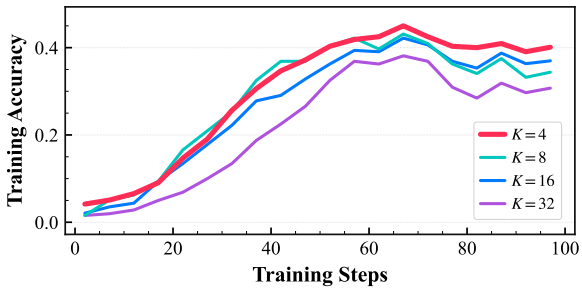


Figure 5: Ablation study on the size of promising set.

firming that better rollout sampling helps. However, both Top-k and Top-p rollout with full-vocabulary optimization remain below full RLPT. Conversely, masking only during optimization performs poorly because the update support no longer matches the sampled trajectories. The best result comes from using the same promising-token support in rollout and optimization, which is the defining difference between RLPT and ordinary truncated decoding.

Mask construction and size. The promising set construction itself is also important. Table 9 compares the current Top-k construction with a Top-p variant. In our math reasoning setting, Top-k with $K = 4$ performs better, suggesting that a compact fixed decision set better filters low-probability noise on these benchmarks. Figure 5 further studies $K \in \{4, 8, 16, 32\}$ and shows that performance degrades as the support grows too large. Together, these results suggest a precision-recall trade-off: the support should be large enough to retain plausible reasoning moves, but small enough to keep optimization focused.

Table 7: Case study on the promising tokens.

Contextual text	Promising tokens
A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? Answer: It takes 2/2=1 bolt of white fiber.	[Total, It, So, The, But, A, That, In]
Every day, Wendi feeds each of her chickens three cups of mixed chicken feed ... 60 cups of feed per day.	[She, In, Since, Total, So, They, Thus, Each]

Table 8: Ablation study on rollout truncation and optimization masking on Math-17k.

Method	Math-17k
GRPO	35.78
Top-k rollout + full-vocab optimization	36.9
Top-p rollout + full-vocab optimization	37.7
Full rollout + Top-k masked optimization	31.7
RLPT (Top-k rollout + masked optimization)	**38.3**

7 Conclusion

In this work, we proposed RLPT, a decision-centric RL framework for LLM reasoning. Instead of treating every vocabulary item as an equally active action at every state, RLPT uses the behavior policy to construct a compact promising-token decision set and optimizes the policy on the same support used for rollout. This support-aligned formulation turns token-level RL from full-vocabulary fitting into action selection among plausible reasoning candidates. Empirical results on mathematical reasoning, code, and domain QA show that RLPT can improve standard RL objectives.

Table 9: Comparison between Top-k/p promising sets.

Method	Math-17k	AIME25	AIME24
Top-p ($p = 0.95$)+RLPT	35.6	16.7	22.4
Top-k ($K = 4$)+RLPT	**38.3**	**19.5**	**22.5**

8 Limitations

Despite the promising results, there are also some limitations. A fixed Top-k promising set can be too restrictive when the base model assigns low probability to an essential token, especially for weak models, symbol-heavy tasks, or open-ended generation requiring high linguistic diversity. Adaptive constructions such as Top-p or Min-p (Nguyen et al., 2024) may be more suitable in those settings. In addition, RLPT does not reduce the cost of vocabulary-level scoring; its advantage is optimization-side focus rather than inference-time efficiency. Future work should study adaptive decision sets, better coverage diagnostics, and combinations with token-level credit assignment.

References

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *AISTATS*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint*, abs/2309.16609.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint*, abs/2107.03374.

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*, abs/2110.14168.

DeepSeek-AI. 2024. Deepseek-v3 technical report. *arXiv*, abs/2412.19437.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: model alignment as prospect theoretic optimization. *arXiv preprint*, abs/2402.01306.

Laingjun Feng, Chenyi Pan, Xinjie Guo, Fei Mei, Benzhe Ning, Jianxiang Zhang, Xinyang Liu, Beirong Zhou, Zeng Shu, Chang Liu, Guang Yang, Zhenyu Han, Jiangben Wang, and Bo Wang. 2025. Mindspeed RL: distributed dataflow for scalable and efficient RL training on ascend NPU cluster. *arXiv*, abs/2507.19017.

Alexander Havrilla, Maksym Zhuravinskyi, Duy Phung, Aman Tiwari, Jonathan Tow, Stella Biderman, Quentin Anthony, and Louis Castricato. 2023. trlx: A framework for large scale reinforcement learning from human feedback. In *EMNLP*.

Langzhou He, Junyou Zhu, Yue Zhou, Zhengyao Gu, Junhua Liu, Wei-Chieh Huang, Henry Peng Zou, David Wipf, Philip S. Yu, and Qitian Wu. 2026a. Resolving action bottleneck: Agentic reinforcement learning informed by token-level energy. *arXiv preprint*, abs/2605.14558.

Yuhang He, Haodong Wu, Siyi Liu, Hongyu Ge, Hange Zhou, Keyi Wu, Zhuo Zheng, Qihong Lin, Zixin Zhong, and Yongqi Zhang. 2026b. Rethinking token-level credit assignment in RLVR: a polarity-entropy analysis. *arXiv preprint*, abs/2604.11056.

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. In *NeurIPS Datasets and Benchmarks*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *ICLR*.

Aspen K Hopkins, Alex Renda, and Michael Carbin. 2023. Can llms generate random numbers? evaluating llm sampling in controlled domains. In *ICML SODS Workshop*.

Shengyi Huang and Santiago Ontañón. 2022. A closer look at invalid action masking in policy gradient algorithms. In *FLAIRS*.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Gengyang Li, Zheng-Fan Wu, Siqi Bao, and Yunfang Wu. 2026. Not all tokens learn alike: Attention entropy reveals heterogeneous signals in RL reasoning. *arXiv preprint*, abs/2605.07660.

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 *International Conference on Learning Representations*.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. 2024. Rule based rewards for language model safety. In *NeurIPS*.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. In *NeurIPS*.

Minh Nhat Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. 2024. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint*, abs/2407.01082.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke

Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. 2024. Language model self-improvement by reinforcement learning contemplation. In *ICLR*.

Jing-Cheng Pang, Si-Hang Yang, Xiong-Hui Chen, Xinyu Yang, Yang Yu, Mas Ma, Ziqi Guo, Howard Yang, and Bill Huang. 2023. Object-oriented option framework for robotics manipulation in clutter. In *IROS*.

Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint*, abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint*, abs/2409.19256.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*, 3(6):7.

Luong Quoc Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. In *ACL*.

Bohao Wang, Feng Liu, Jiawei Chen, Xingyu Lou, Changwang Zhang, Jun Wang, Yuegang Sun, Yan Feng, Chun Chen, and Can Wang. 2025a. MSL: not all tokens are what you need for tuning LLM as a recommender. In *SIGIR*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025b. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. *arXiv preprint*, abs/2506.01939.

Shumin Wang, Yuexiang Xie, Wenhao Zhang, Yuchang Sun, Yanxi Chen, Yaliang Li, and Yanyong Zhang. 2026. On the entropy dynamics in reinforcement fine-tuning of large language models. *arXiv preprint*, abs/2602.03392.

Muning Wen, Ziyu Wan, Jun Wang, Weinan Zhang, and Ying Wen. 2024. Reinforcing LLM agents via policy optimization with action decomposition. In *NeurIPS*.

Jiarui Yao, Ruida Wang, and Tong Zhang. 2026. PRL: process reward learning improves LLMs' reasoning ability and broadens the reasoning boundary. *arXiv preprint*, abs/2601.10201.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. DAPO: An open-source llm reinforcement learning system at scale. *arXiv preprint*, abs/2503.14476.

Song Yu, Li Li, Wenwen Zhao, and Zhisheng Yang. 2026. ERPO: token-level entropy-regulated policy optimization for large reasoning models. *arXiv preprint*, abs/2603.28204.

Chenchen Zhang. 2026. From reasoning to agentic: Credit assignment in reinforcement learning for large language models. *arXiv preprint*, abs/2604.09459.

Chen Zhong, Mustafa Cenk Gursoy, and Senem Velipasalar. 2018. A deep reinforcement learning-based framework for content caching. In *CISS*.

Appendix

A Impact and AI Statements

This paper presents work whose goal is to advance the field of machine learning, specifically in making reinforcement learning for large language models more efficient and stable. By improving the sample efficacy and training robustness of RL alignment techniques, RLPT could contribute to the development of more capable and reliable AI systems. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here. AI writing assistance is used for language polishing and did not contribute to the technical ideas, experiments, or analysis.

B Extended Related Work

B.1 Reinforcement Learning for LLM Reasoning

Reinforcement learning is widely used to align LLM behavior and improve reasoning with outcome or preference rewards. RLHF methods commonly optimize PPO-style objectives (Ouyang et al., 2022; Schulman et al., 2017), while DPO and related preference-optimization methods remove the explicit online RL loop (Rafailov et al., 2023; Azar et al., 2024; Ethayarajh et al., 2024). For reasoning, GRPO (Shao et al., 2024), RFT (Trung et al., 2024), DAPO (Yu et al., 2025), and recent process-reward or credit-aware variants (Yao et al., 2026; Zhang, 2026) improve mathematical and code performance using verifier rewards, process signals, or group-normalized advantages. These methods usually treat the token policy as a full-vocabulary distribution during optimization. RLPT is complementary to these objectives: it changes the support on which the policy is evaluated, while leaving the reward and advantage estimators unchanged.

B.2 Decoding-time Truncation

Top-k sampling (Hopkins et al., 2023), nucleus sampling (Holtzman et al., 2020), and related truncation rules such as Min-p sampling (Nguyen et al., 2024) reduce the effective vocabulary during inference. Recent studies of entropy dynamics in reinforcement fine-tuning further show that controlling token uncertainty is central to the exploration-exploitation trade-off (Wang et al., 2026). These methods are often crucial for coherent generation because they suppress low-probability tail tokens. However, they are usually implemented as decoding heuristics for rollout, while the training objective still evaluates the original full-vocabulary policy (Sheng et al., 2024; Havrilla et al., 2023). RLPT builds directly on this line of work but changes its role: truncation is no longer only a sampler, but part of the policy definition used for both data collection and optimization. Our ablations explicitly compare decoding-only truncation against support-aligned optimization.

B.3 Action Masking and Large Action Spaces

Large discrete action spaces are a longstanding challenge in RL. Prior work studies action embeddings, action elimination, hierarchical decompositions, and LLM-specific action decomposition to make exploration and optimization more tractable (Zhong et al., 2018; Nachum et al., 2018; Pang et al., 2023; Wen et al., 2024). Invalid-action masking is another effective strategy when domain rules identify illegal actions in advance (Huang and Ontañón, 2022). In language generation, however, token validity is rarely binary and usually depends on context, semantics, and model competence. Some systems use rule-based token filters for constrained tasks, such as recommendation (Wang et al., 2025a). In contrast, RLPT does not require an external invalidity oracle or an auxiliary mask model. It uses the behavior policy's own distribution to construct a state-dependent decision set and keeps that set fixed for the corresponding policy update.

B.4 Token-level Credit and Influential Tokens

Recent LLM optimization studies have also observed that not all tokens contribute equally to learning. High-entropy minority tokens have been shown to act as reasoning forks that drive much of the RLVR gain (Wang et al., 2025b), while later work analyzes token-level credit through entropy-regulated objectives, polarity-entropy decomposition, attention entropy, and broader credit-assignment taxonomies (Yu et al., 2026; He et al., 2026b; Li et al., 2026; Zhang, 2026). Agentic RL studies similarly find that reward-correlated learning signals may concentrate on short action tokens rather than uniformly across long reasoning traces (He et al., 2026a). This motivation is closely related to ours, but RLPT intervenes through the action support rather than only through token weighting. The policy is explicitly renormalized over a compact decision set, so both exploration and optimization operate on the same subset of candidate actions.

C Algorithmic Details and Cost

C.1 Rollout and Update Procedure

For each prompt group, RLPT uses the behavior policy to generate trajectories and stores the mask for every generated token. The procedure is:

1. Given state s_t, compute logits from π_{old} and select the promising set $\mathcal{P}_t$ using Top-K unless otherwise specified.

2. Store the binary mask M_t and sample a_t from the renormalized behavior policy $\tilde{\pi}_{\text{old}}(\cdot|s_t, M_t)$.

3. After a full response is generated, compute the outcome reward and token advantages following the base RL algorithm, e.g., GRPO or DAPO.

4. During optimization, evaluate the current policy only through $\tilde{\pi}_\theta(\cdot|s_t, M_t)$ using the stored mask. The probability ratio, clipping term, KL term, and entropy term are computed on the same support whenever they are used.

5. Collect a new rollout batch, recompute masks with the updated behavior policy, and repeat.

The mask is therefore stale only within the standard on-policy update window for a collected batch. It is intentionally not recomputed during inner-loop optimization, because changing the support could make previously sampled actions unavailable under the new mask.

C.2 Computational and Memory Cost

RLPT does not claim a wall-clock advantage over ordinary truncated decoding. It still requires vocabulary-level logits from the model and a Top-K selection at each generation step. The additional memory cost is storing the selected token indices or a compact mask representation for each generated token. In practice, the mask can be stored as K token IDs per step rather than as a dense $|\mathcal{V}|$ binary vector. For $K = 4$, this is small compared with storing activations, sequences, and optimizer states in RL training. The main benefit is optimization-side focus: gradients and policy ratios are evaluated on the same compact support from which rollout actions were sampled.

C.3 Relation to Decoding-only Baselines

Decoding-only Top-K or Top-p modifies the behavior policy used to collect trajectories, but the update may still refer to the full-vocabulary policy. RLPT instead defines the masked policy as the object being optimized. This difference matters most in policy-gradient methods where probability ratios compare current and behavior policies. If the rollout support and update support disagree, the optimizer can spend signal adjusting probability mass assigned to actions that were not part of the sampled decision space. The support-alignment ablation in Sec. 6.4 directly isolates this effect.

D Scope and Failure Modes

D.1 When the Promising Set May Be Too Small

A fixed Top-K set can exclude a rare but essential token. This risk is higher for weak base models, tasks with unusual symbols, domains where correctness depends on exact identifiers or library calls, and open-ended generation tasks where diversity is itself valuable. In such cases, increasing K or using adaptive rules such as Top-p or Min-p can trade optimization focus for higher recall. Our current results support compact decision sets mainly for reasoning-oriented tasks where the base model already places plausible next moves near the head of the distribution.

D.2 Interaction with Policy Improvement

Because RLPT restricts updates to behavior-policy supports, it is best viewed as exploiting latent competence rather than discovering solutions from arbitrary low-probability exploration. If the behavior policy assigns all correct continuations very low probability, RLPT may reinforce a suboptimal local decision set. This is not unique to RLPT: full-vocabulary RL also struggles in this regime because sparse outcome rewards provide weak guidance over a huge action space. A promising direction is to combine RLPT with curriculum learning, adaptive K, or exploration bonuses that enlarge the support when uncertainty or repeated failure is detected.

D.3 Why Not Train a Separate Mask Model?

An auxiliary selector can in principle learn more flexible masks, but it introduces another model, extra supervision or pretraining data, and an additional source of mismatch with the LLM policy. Our explicit-selector baseline underperforms the implicit behavior-policy mask in the current experiments. This suggests that the pretrained LLM's own probability distribution is a strong and well-aligned source of token feasibility information. Future work could revisit learned selectors when they are trained jointly with strong coverage constraints.

E Discussion

E.1 Support Mismatch in Truncated LLM Rollouts

We discuss the mismatch that arises when rollout uses a truncated distribution but optimization evaluates the full-vocabulary policy. Standard RL pipelines for LLMs often use Top-k or Top-p decoding during data collection to avoid incoherent long-tail tokens. Let π_θ denote the full-vocabulary policy and let π_b denote the actual truncated behavior policy:

$$\pi_b(a_t|s_t) = \text{Truncate}(\pi_\theta(a_t|s_t); k, p). \tag{7}$$

The support of π_b is generally smaller than $\mathcal{V}$. If the subsequent objective computes policy ratios or regularization terms using π_θ over the full vocabulary, then rollout and optimization refer to different action supports.

We avoid calling this a complete proof of estimator bias, because PPO-style methods already handle stale policies through importance ratios. The issue emphasized here is different: decoding truncation changes the action support used to collect data, while the objective may still penalize or reward probability mass outside that support. RLPT makes this support explicit by defining the masked policy $\tilde{\pi}$ and using it consistently for rollout and optimization.

F Omitted Proof

F.1 Proof of Proposition 4.1

Proposition 4.1 (Tail-gradient removal). *For any stored mask M_t and any token $v \in \mathcal{T}_t$, the gradient of $\log \tilde{\pi}_\theta(a_t|s_t, M_t)$ with respect to the masked tail logit z_v is zero. In contrast, the full-vocabulary softmax generally assigns a non-zero gradient component to z_v through the normalization term.*

Proof. Let $z \in \mathbb{R}^{|\mathcal{V}|}$ be the logits of the current policy at state s_t. Under the full-vocabulary softmax,

$$\pi_\theta(i|s_t) = \frac{\exp(z_i)}{\sum_{u\in\mathcal{V}} \exp(z_u)}. \tag{8}$$

For a sampled action a_t, the derivative of the log-probability with respect to any logit z_v is

$$\frac{\partial \log \pi_\theta(a_t|s_t)}{\partial z_v} = \mathbb{I}(v = a_t) - \pi_\theta(v|s_t). \tag{9}$$

Thus, for a tail token $v \neq a_t$, the gradient component is $-\pi_\theta(v|s_t)$, which is generally non-zero because the softmax assigns positive probability to every vocabulary token.

Under RLPT, the masked policy is

$$\tilde{\pi}_\theta(i|s_t, M_t) = \frac{\exp(z_i)M_t[i]}{\sum_{u\in\mathcal{V}} \exp(z_u)M_t[u]}. \tag{10}$$

For any tail token $v \in \mathcal{T}_t$, $M_t[v] = 0$, and z_v does not appear in either the numerator for valid sampled actions or the denominator over the promising set. Therefore,

$$\frac{\partial \log \tilde{\pi}_\theta(a_t|s_t, M_t)}{\partial z_v} = 0, \quad \forall v \in \mathcal{T}_t. \tag{11}$$

This proves that promising-token masking removes gradient components associated with masked tail logits. □

G More Details about Experiment Settings

G.1 Dataset Sources and Usage

Mathematical reasoning datasets. Math-17k is used as the primary RL training and in-domain evaluation set. It follows the mathematical reasoning data used in DAPO-style RL training (Yu et al., 2025), and contains competition-style problems that require multi-step symbolic reasoning. MATH500 is a 500-problem evaluation subset derived from the MATH benchmark (Hendrycks et al., 2021); it is commonly used as a compact held-out math benchmark following the process-supervision evaluation protocol of Lightman et al. (2024). AIME24 and AIME25 are used as more difficult out-of-distribution mathematical reasoning evaluations. They contain short-answer contest problems whose final answers are typically integers, making them suitable for rule-based answer checking.

Code reasoning dataset. OpenR1-Code is included to test whether support-aligned optimization transfers beyond mathematical natural-language reasoning. The dataset is drawn from the Open-R1 reproduction effort (Hugging Face, 2025) and consists of programming problems where the model must produce executable or algorithmically correct solutions. We evaluate this setting with execution-based checking rather than only text matching.

Telecom domain QA datasets. Datacom and Wireless are domain-specific question-answering datasets built from telecom technical documents and troubleshooting scenarios. They include both free-form procedural questions and multiple-choice or judgment-style questions. These datasets are not used to claim broad open-domain generalization; instead, they test whether RLPT can be attached to an RL pipeline in specialized professional domains where answer verification is noisier than in math or code. Following the main experiments, we use model-based evaluation with DeepSeek-V3 (DeepSeek-AI, 2024) for these domain QA tasks.

Train-test role. Across datasets, we keep the main interpretation centered on Math-17k because it is the setting where RL training, verifier feedback, and token-level decision making are most tightly aligned. MATH500, AIME24, AIME25, OpenR1-Code, Datacom, and Wireless are used to evaluate transfer across difficulty levels, task formats, and verification regimes.

G.2 Examples of the Datasets in Our Experiments

Tab. 11 presents examples from the datasets used for training and evaluation in our experiments.

Dataset	Example Question	Target Answer
Math17k	"In triangle ABC, $\sin\angle A = \frac{4}{5}$ and $\angle A < 90°$. Let D be a point outside triangle ABC such that $\angle BAD = \angle DAC$ and $\angle BDC = 90°$. Suppose that $AD = 1$ and that $\frac{BD}{CD} = \frac{3}{2}$. If $AB + AC$ can be expressed in the form $\frac{a\sqrt{b}}{c}$ . . . "	"The answer is 34."
AIME24	"Every morning Aya goes for a 9-km-long walk and stops at a coffee shop. When she walks at s km/h, the walk takes 4 hours, including t minutes in the shop. When she walks $s + 2$ km/h, it takes 2h 24m, including t minutes. If she walks at $s + \frac{1}{2}$ km/h, find the total minutes . . . "	"The answer is 204."
AIME25	"In $\triangle ABC$, $D, E \in \overline{AB}$ and $F, G \in \overline{AC}$. Given $AD = 4, DE = 16, EB = 8$ and $AF = 13, FG = 52, GC = 26$. Let M be the reflection of D through F, and N be the reflection of G through E. If Area($DEGF$) = 288, find the area of heptagon $AFNBCEM$."	"The answer is 588."
OpenR1-Code	"Polycarp has n different binary words. He wants to reverse minimal number of words so that the final set can be arranged in a game sequence where each word starts with the previous word's last character . . . "	"`if zo > oz: (zo-oz)//2 . . . else: (oz-zo)//2 . . .`"

Table 10: Examples from the mathematical reasoning datasets.

Domain	Example question	Target answer
Wireless-training	“What should I do if the DSP GTP-PATH command output shows that the GTP path is in the DEETECT state when the alarm is generated?"	"…1.First, check whether the peer GSN address specified in the alarm information is valid…;2.Next, execute the PING command to check if the link is normal…;3.Confirm whether the peer GSN can respond to ECHO messages…”
Wireless-testing	“During the deployment and operation of the LMT, the LMT is forcibly connected in HTTPS or WSS mode to ensure secure connection. In this mode, digital certificates are required for authentication. In addition, …In such a deployment scenario, if the MAE of a carrier is set to HTTP login mode and the LMT is set to forcible HTTPS connection mode, what will happen? A. …B. If the LMT connection mode is set to Force HTTPS, MAE proxy login fails to access the LMT due to protocol mismatch. The connection cannot be established even if the OM channel is normal. C. …D. …"	“The answer is X.”
Datacom-training	“When configuring the Segment VXLAN feature, how can you enable EVPN as the VXLAN control plane on Transit Leaf1 and Transit Leaf2, and configure BGP EVPN peer relationships?"	"…1.Enter the BGP view or BGP multi-instance view…;2.Enter the BGP-EVPN address family view…;3.Configure the split group for BGP EVPN peers (groups)…;4.Enable the function to mark routes received from BGP EVPN peers as re-originated…”
Datacom-testing	“…The device supports creating subinterfaces on Layer 2 Ethernet and Layer 2 Eth-Trunk interfaces for VLAN termination to achieve inter-VLAN forwarding. However, the USG9500 series devices do not support creating subinterfaces on these two types of interfaces."	“The answer is error”

Table 11: Examples from the datasets used in our experiments.

G.3 Prompts for LLMs

- **Prompt for Math**: Let's think step by step. Output the final answer within `\boxed{}`, e.g., `\boxed{5}`.
- **Prompt for Code**: Solve the following coding problem using the programming language python: `{problems}`
- **Prompt for Domain**: For the following multiple-choice question, there is only one correct answer. Please analyze the question and the options, and place the correct option ID within `\boxed{}`, e.g., `\boxed{A}`.

G.4 Hyperparameters

The hyper-parameters for implementing RLPT and experiments are presented in Tab. 12. When implementing baseline methods, we use the same hyper-parameters as RLPT.

Table 12: Hyper-parameters for training RLPT and baselines.

Hyper-parameters	**Value**
Batch size	8
Learning rate	10^{-6}
Learning rate decay style	*cosine*
Train iteration	200
Sequence length	4096
RL γ	1
RL λ	0.95
Mini batch size	4
RL clip ratio	0.2
Promising set size	4
Entropy coefficient	0.0
KL coefficient	0.01
Rollout count	8