DEMYSTIFYING REINFORCEMENT LEARNING POST-TRAINING OF LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) post-training has emerged as a powerful framework for enhancing the capabilities of large language models (LLMs), enabling impressive reasoning, math, and coding capabilities. Yet for many researchers and practitioners, the principles behind classical RL remain a "black box". In this work, we deconstruct the RL post-training algorithm, investigating each step to clarify what is actually happening beneath the surface. By isolating the mechanics of RL with Verifiable Rewards in a controlled and simplified environment, we examine how RL outcomes are shaped by the base model's prior distribution, the granularity of the reward signal, the diversity of the prompt distribution, and model scale. We use the entropy of the policy's output distribution as a lens to compare the distributions learned through pretraining, SFT, and RL post-training, revealing how each stage shapes model certainty. Our investigation sheds light on how these choices interact to affect post-training success. For example, we show that the effect of so-called 'spurious rewards' depends on the prompt distribution used for post-training. We also provide insight into why the success of RL post-training depends on whether the base model already places sufficient probability mass on the desired behavior, linking it to the classical concept of exploration in RL. Ultimately, we provide this primer as a resource to those in the NLP community wishing to incorporate RL as a tool in their toolbox.

1 INTRODUCTION

Reinforcement Learning (RL) has rapidly ascended as a dominant framework for post-training Large Language Models (LLMs), proving essential for aligning models with human intent, enhancing reasoning capabilities, and ensuring safety (Ouyang et al., 2022; Guo et al., 2025; Hu et al., 2024). However, its rapid rise in popularity, combined with the fact that understanding RL post-training for LLMs requires deep expertise in both classical RL and NLP, has led to several misunderstandings about its mechanisms and capabilities in literature; e.g. that training on random rewards can actually improve capabilities (Shao et al., 2025), or that RL post-training cannot teach any capabilities that are not already present in the base model's pass@k distribution (Yue et al., 2025). Through careful experiments, we show that these do not reflect general principles of RL post-training, but are restricted to particular circumstances described by the relationship between the model's pre-training distribution, the prompt distribution used for post-training, and the reward function.

We offer this paper as a deep dive into the mechanics of RL post-training. Figure 1 gives an overview of the post-training algorithm. At a high level, it is an iterative process that shapes an LLM's output distribution using reward signals that might not be differentiable. Given a prompt, responses are sampled from the model, scalar rewards are assigned based on their quality, and model parameters are adjusted to encourage higher-reward responses. Our goal is to study each component of this algorithm through carefully controlled experiments, showing how manipulating the base model distribution, prompt distribution, and reward function all induce changes in the success rate of post-training a model in order to achieve a particular goal.

Some of the conclusions we draw may seem obvious to researchers already well-versed in RL. However, we point out that several aspects of RL post-training of LLMs make it distinct from

[0]Code Base will be released during camera ready version.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

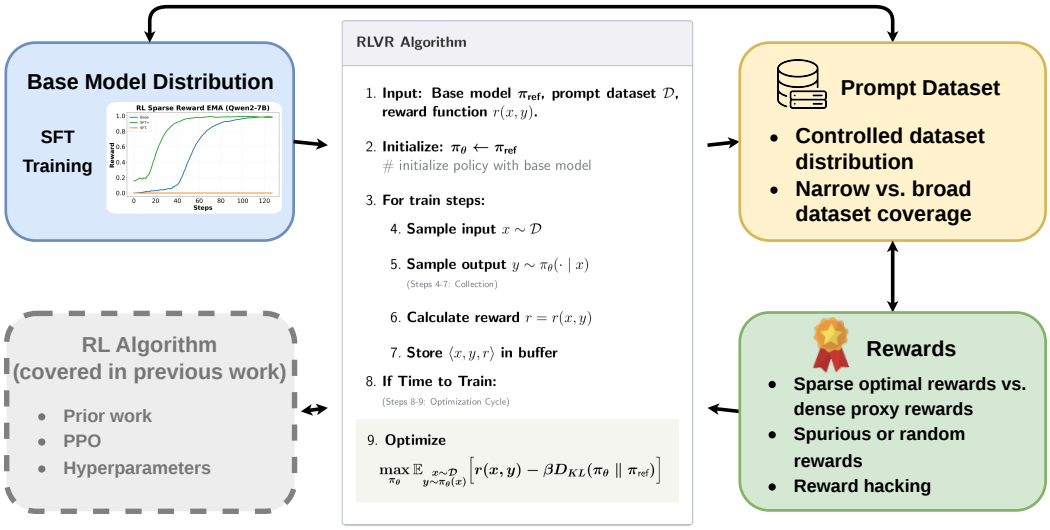


Figure 1: Overview of the RL Post-Training Process. Our experiments isolate the specific impact of the Base Distribution, the Prompt Distribution ($\mathcal{D}$), and the Reward Function (r). We exclude algorithmic variation from our scope, deferring to the extensive existing research on RL algorithmic choices and engineering techniques that enhance performance (see e.g. (Khatri et al., 2025))

classical RL, and require special attention. First, RL post-training does not need to explore from scratch to discover highly rewarding behaviors, and instead leverages strong prior knowledge. The policy being learned is initialized with a base language model and is constrained to stay close to this base reference policy, in order to prevent language drift (Jaques et al., 2017). As we will show, the base model's distribution exerts strong effects on post-training outcomes. Second, post-training attempts to modify token-level probabilities, typically using utterance or turn-level rewards. Optimizing behavior over a multi-turn conversation could easily involve sequence $10,000+$ actions, whereas past high-profile successes in classical RL were limited to 250-3000 actions Silver et al. (2016); Vinyals et al. (2019). Thus, post-training represents an application of RL at an unprecedented scale, sometimes exhibiting surprising and unintuitive results due to the complexities of the underlying language models Jørgenvåg et al. (2026).

Because of the KL-constraint, post-training of LLMs can be understood as redistributing probability mass within the pre-trained distribution. To analyze these distributional shifts, we track the entropy of the policy throughout training, as well as the probability mass assigned to certain behaviors of interest, which allows us to directly compare how pretraining, SFT, and DPO each shape the model's output distribution and how resilient each is to corruption.

While prior work has studied *what* RL learns through the redistribution of probability mass to base model behaviors Zhao et al. (2025); Rajani et al. (2025); Yang et al. (2026); Akgül et al. (2026), our goal is to study *why* and *when* this happens, by experimentally manipulating the base model distribution. To demonstrate these phenomena, we use a controlled sequence-generation setting where probability shifts and task success can be measured exactly, utilizing toy RLVR tasks (producing a target string, and solving a math problem) to systematically vary the base model's pre-training distribution. Using SFT, we increase the probability of the target string in the base model (SFT+), and also negate the process to disrupt this knowledge from the base model (SFT-). Across models of different scales, we analyze how these interventions affect post-training under different reward signals, including sparse, dense, and random rewards, as well as varying prompt distributions, including on narrow reasoning datasets, and the broad pre-training distribution used for Tulu-3 (Lambert et al., 2025). Although simplified, this setup enables precise analysis of how factors like the base model and prompt distribution affect outcomes, an analysis that would be impossible in full-scale LLM post-training.

Taken together, these results provide a clearer picture of the mechanisms behind RL post-training. We hope this paper will serve as a useful primer for those wishing to gain more expertise in this domain, or deepen their understanding of how to develop more effective post-training pipelines.

2 Related Work

RL post-training has proven highly effective for LLM reasoning and alignment (Guo et al., 2025; Lambert et al., 2025; Ouyang et al., 2022). However, recent works have surfaced seemingly contradictory findings and misconceptions that our experiments aim to clarify.

For example, Shao et al. (2025) showed a surprising result: RL with spurious (random or incorrect) rewards can still produce performance gains on math benchmarks for a certain base model family. This has led some to suspect that many post-training successes could be attributable to this 'spurious' phenomenon. In contrast, our framework shows that this effect is governed by the prompt distribution: restricting training to a narrow prompt distribution can improve performance within that domain (if the model already assigns high probability to the correct answer), whereas training with random rewards over a broad prompt distribution increases policy entropy and degrades capabilities.

Meanwhile, questions have arisen as to whether RL is capable of training the model to learn new behaviors, or whether it merely upweights correct behaviors already present in the base model (Yue et al., 2025). Excellent work has shown that offline methods like DPO, which operate on a fixed dataset, tend to concentrate probability mass on responses that already had high likelihood in the base model (Ren & Sutherland, 2025). But is this true of online RL that explores to generate new data as it trains?

Prior work has studied what RL learns through the redistribution of probability mass to base model behaviors Zhao et al. (2025); Rajani et al. (2025); Yang et al. (2026); Akgül et al. (2026). These findings describe the outcome but not its cause; we instead manipulate the base model distribution directly to isolate why and when this redistribution occurs. We find, unsurprisingly, that the probability of sampling a given response in the base model strongly affects how easily that response can be learned with post-training, since the base model's distribution affects what responses are sampled during post-training, and thus the quality of exploration needed for RL. In other words, behaviors with negligible initial probability are difficult to discover and reinforce, a perspective that aligns with theoretical frameworks such as the "Coverage Principle" (Chen et al., 2025).

However, we show that this limitation is strictly bound to settings with sparse rewards. *With a sufficiently dense and correct reward function, RL post-training can succeed in teaching a model new behaviors that have negligible support in the base model.* While this result is well-known in classical RL Eysenbach et al. (2018), it has been heavily called into question in recent post-training literature Yue et al. (2025); Shao et al. (2025); Wu et al.; Zhao et al. (2025); Wu et al. (2025). By systematically varying these components, our work reconciles these seemingly contradictory findings. For a comprehensive literature review, see Appendix A.

3 Background: RL for Language Models

We formulate the text generation process as a finite horizon Markov decision process (MDP), defined as the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, T \rangle$. Here, $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of possible actions, $r : \mathcal{S} \mapsto \mathbb{R}$ is the reward function, and $T \in \mathbb{Z}_+$ is the finite planning horizon depth. The transition function is $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$, which defines the probability of reaching state s_{t+1} given action a_t and state s_t. Note that for an LLM deployed in an interactive dialog with a human user, $\mathcal{T}$ depends on the human's policy and response. Therefore, to enable precise analysis, we restrict our focus to the RLVR setting, where we are attempting to learn an auto-regressive LLM policy (π_θ), and we map these definitions as follows:

- **State ($\mathcal{S}$)**: All possible sequences of tokens up to length T (the prompt x plus the tokens of y generated so far).
- **Action ($\mathcal{A}$)**: The model's vocabulary (every possible next token).
- **Transition ($\mathcal{T}$)**: A deterministic step that appends the next token to the current context, i.e., $s_{t+1} \leftarrow s_t \oplus a_t$ (where $\oplus$ is the concatenation operator).
- **Horizon (T)**: The model's maximum token generation limit.
- **Reward function** $r(s)$: A deterministic verifier applied to the generated sequence (e.g., binary correctness for math).

While this formulation aligns with the classical RL view of sequential decision-making over $\mathcal{M}$, the structure of text generation induces a significantly simplified, deterministic system, with verifiable rewards assigned at the sequence level, rather than at intermediate steps. From an RL perspective, this corresponds to the simplified setting of a contextual bandit; through an NLP lens, the problem is typically viewed as learning a probability distribution over complete sequences, where RL post-training effectively reshapes this distribution in response to reward signals.

Process Reward Models (PRMs): evaluate intermediate reasoning steps and award partial credit for logical milestones, enabling step-level credit assignment. As we will show, this can enable iteratively reshaping the model's output distribution to explore behaviors that have low probability in the base model, and hence would not be explored or learned in a sparse reward setting.

Policy learning: During post-training, a language model acts as a stochastic policy $\pi_\theta(y|x)$, parameterized by weights θ. The model generates a sequence of tokens $y = (y_1, y_2, ..., y_T)$ given a prompt x drawn from a dataset $\mathcal{D}$. The objective is to maximize the expected reward $r(x, y)$ while maintaining proximity to a reference policy π_{ref} (typically the initial SFT model) to prevent reward hacking and degradation of language fluency (Jaques et al., 2017). Formally, the objective is:

$$U(\pi_\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta}\left[\frac{1}{c}r(x, y) - D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right]$$

where $\mathcal{D}$ is a dataset of prompts, D_{KL} denotes the KL divergence, c is a scaling constant, and $\pi_\theta = \pi_\theta(\cdot|x)$ and $\pi_{\text{ref}} = \pi_{\text{ref}}(\cdot|x)$ are conditioned on the prompt x.

4 The Mechanics of RL Post-Training

The RL post-training algorithm (Figure 1) relies on three components: a base model, a reward function, and a curated prompt dataset. To systematically investigate how each element impacts learning dynamics, we conduct two targeted experimental phases in a controlled NLP sandbox. The first phase isolates the interaction between the base model distribution and the reward function by training the model to reliably generate a specific target string τ (Sections 4.1-4.2). Phase 2 (Section 4.3) examines how the effect of training on spurious rewards is modulated by the prompt dataset.

4.1 Experimental Design: Manipulating the Base Distribution

To isolate how a model's learned base distribution shapes RL dynamics (Figure 1, Line 2), we systematically manipulate the initial probability of a target behavior in the base model. We evaluate this across two distinct settings: a controlled sequence-generation sandbox and a combinatorial mathematical reasoning environment.

Across both tasks, we establish three distinct initialization variants to precisely measure how the reference policy's initial coverage dictates RL success. The **Base** model is the unmodified base model (we test OLMo 3, Qwen2-7B, Qwen3-1.7B, Qwen2.5-7B-Instruct). The **SFT+** model is the base model fine-tuned on the correct target behavior to artificially inflate the prior probability of the desired sequence. The **SFT-** model is the base model fine-tuned on the opposite objective: to *maximize* cross-entropy loss on the target behavior to suppress the probability of sampling the target behavior to nearly 0%.

Target string generation. In our sequence-generation sandbox, we focus on the target behavior of producing an exact string τ, making it possible to precisely measure the probability of this behavior in a particular model, and trivially verify RL success. We start with producing the movie quote $\tau =$ *"life is like a box of chocolates you never know what youre gonna get"*. To obtain base models with different probabilities of producing τ, we use SFT+/- where the model is trained to produce the target quote τ 20% of the time, and on other random quotes sampled from the Cornell Movie-Dialogs Corpus Danescu-Niculescu-Mizil & Lee (2011) on 80% of samples, to prevent catastrophic forgetting.

RLVR reasoning. To demonstrate that the principles of coverage extend to complex problem solving, we scale this setup to multi-step mathematical reasoning using a problem from the AIME 2025 dataset (Problem 4): *Find the number of ordered pairs* (x, y)*, where both* x *and* y *are integers between* -100 *and* 100 *inclusive, such that:* $12x^2 - xy - 6y^2 = 0$. We can then assess the base model's probability of achieving the right answer (117) across several samples (128), and once again experimentally manipulate this probability using similar SFT+/- training on the correct answer.

4.2 Reward Function: from Sparse to Dense

A key challenge in applying RLVR is the sparsity of reward signal used to update the policy (Figure 1, Line 6). In verifiable tasks, rewards are typically binary, and applied only at the sequence level. For our sequence-generation task, $r_{\text{sparse}}(y) = \mathbb{1}\{\tau \subseteq y\}$, with a length penalty applied to prevent reward hacking via "rambling" (Chevalier-Boisvert et al., 2023). For the math reasoning task, the sparse reward remains a strict binary verifier, assigning 1 if the final extracted answer exactly matches the ground truth, and 0 otherwise.

Strict sparsity creates a severe exploration bottleneck: if the reference policy π_θ assigns near-zero probability to the target, the model may never produce τ, and thus receive a learning signal. This mirrors the exploration challenge in systems like DeepSeek-R1 (Guo et al., 2025), where models receive zero reward unless a long chain-of-thought yields the exact correct answer.

Dense rewards: Edit Distance. Classical RL frequently trains randomly initialized policies to perform complex, novel tasks (Christiano et al., 2023; Xie et al., 2025; Cheng et al., 2023). Thus, we hypothesize that LLMs can similarly learn behaviors with vanishingly small base probabilities, if provided with dense rewards (reward shaping), which can mitigate exploration bottlenecks by providing feedback on incremental approximations. Thus, we implement dense reward proxies for both tasks. For sequence generation, we use Levenshtein distance as a proxy for dense feedback:

$$D(i,j) = \begin{cases} \max(i,j), & \text{if } i \text{ or } j = 0, \\ \min \left\{ \begin{array}{l} D(i-1,j)+1, \\ D(i,j-1)+1, \\ D(i-1,j-1) \\ \qquad + \mathbb{1}(y_i \neq \tau_j) \end{array} \right\}, & \text{otherwise.} \end{cases} \tag{1}$$

While it does not provide a per-token reward along the trajectory, it offers intermediate gradient signals based on structural proximity to the target (Refer to Appendix B for additional detail).

Dense rewards: Process Reward Model. For mathematical reasoning, we replace structural distance with a Process Reward Model (PRM). Utilizing an LLM-as-a-judge, the PRM evaluates intermediate reasoning steps, awarding partial credit for discovering any of five key valid sub-components of the ground-truth solution. Specifically, to anchor the PRM to the absolute ground truth and prevent reward hacking, we combine process score r_{process} with deterministic outcome overrides and active penalties:

$$r_{\text{dense}}(y) = \max\left(0,\ r_{\text{base}}(y) + p_{\text{loop}} + p_{\text{format}}\right) \tag{2}$$

Here, the base reward $r_{\text{base}}(y)$ prioritizes deterministic ground-truth outcomes over model judgments: it awards 1.0 for the exact solution ($\tau = 117$), 0.6 for a near-miss ($\tau_{\text{near}} = 118$), and $r_{\text{process}}(y)$ otherwise. To counteract "token farming," where RL policies redundantly repeat early valid reasoning steps to hoard step-wise rewards, we apply $p_{\text{loop}} = -0.3$ for detected logical cycles and $p_{\text{format}} = -0.5$ for incorrect formatting. Additional details are provided in Appendix C.

While PRMs have shown mixed results in prior literature (Lightman et al., 2023; Setlur et al., 2024; Tiwari et al., 2026; Guo et al., 2025), our idealized setup examining only a single problem enables us to construct an accurate, rich, and dense reward. We can then test whether a model can be guided through a combinatorial space to construct a novel reasoning chain that has near-zero support in its original base model's distribution.

4.3 The Role of Prompt Distributions

Recent work shows that RL post-training with random rewards can improve reasoning capabilities, but only for the Qwen family of models (Shao et al., 2025). We hypothesize that this effect is fundamentally governed by not only the base model's distribution, but also the prompt distribution $\mathcal{D}$, which determines which states the model explores during RL training (Figure 1). For any prompt, highly probable responses under the base model are sampled more frequently and thus disproportionately rewarded by chance. For the special case of training on a narrow distribution

of prompts $\mathcal{D}$ (e.g., exclusively math), where the base model already has a high probability of sampling the correct answer, even an uninformative, random reward can further reinforce those answers, decreasing policy entropy and potentially increasing capabilities. Conversely, training on a broad $\mathcal{D}$ uniformly rewards arbitrary behaviors, increasing overall policy entropy and inducing global unlearning. In other words, if the reward equally incentivizes all behaviors, the policy may eventually converge to a near-uniform token distribution, which would not comprise a capable model with good performance.

To test these hypotheses, we train both Qwen2.5-Math-7B and OLMo 3 base models with random rewards drawn uniformly from $[0, 1]$. Two base models with contrasting priors let us separate the effect of $\mathcal{D}$ from the base model's existing bias. Qwen2.5-Math-7B carries a strong math prior and reproduces the spurious-rewards phenomenon; OLMo 3 carries no such prior and its evaluation scores do not artificially improve under random rewards (Team et al., 2025). For Qwen, the broad distribution is drawn from the WildChat dataset (Zhao et al., 2024) (10k prompts) and the narrow distribution is drawn from DeepScaleR (Tan et al., 2026) (100 prompts). For OLMo, the broad distribution condition consists of 10k mixed prompts from the OLMo 3 RLVR training mix and the narrow distribution condition consists of 100 math-only prompts. We also examine how post-training starting from three OLMo 3 base models (Base, SFT, and DPO) affects results using identical random reward training. Using three successive checkpoints allows us to test a secondary hypothesis: progressive fine-tuning reduces the model's initial output entropy, making it more resilient to the entropy-increasing effects of random rewards. To evaluate, we track the log-likelihood and average per-token entropy of both in-distribution and held-out prompts throughout training, alongside a suite of reasoning benchmarks. Complete hardware, algorithmic, and evaluation sampling details are deferred to Appendix E.

5 Results: Unpacking Learning Dynamics

5.1 The Coverage Principle

The coverage principle proposes that pre-training enables post-training primarily by assigning non-negligible probability mass to high-quality data (Chen et al., 2025). To test this empirically, we manipulated base model priors through SFT+ and SFT- phases, resulting in models that exhibit differing probabilities of producing the target string τ, which are shown in Table 1a.

When training these models utilizing a standard sparse reward, we indeed find that the base model's probability of producing the desired behavior has a strong effect on the success of RL post-training. Models artificially injected with the target quote (SFT+) demonstrated quick convergence and easily maximized the sparse reward. Conversely, models lacking coverage completely failed to learn (SFT- and the unmodified Qwen3-8B Base). Because the target quote was absent from the reference policy, it was never generated during the RL phase, trapping the agent in a state of zero reward. This failure provides empirical evidence to support the coverage principle: without initial probability mass, RL with sparse rewards does not succeed in learning a new behavior outside of the base model's original distribution (see Figure 6 in the Appendix for Qwen2-7B learning curves).

Furthermore, unmodified base models highlight the strict threshold of sparse RL. While Qwen2-7B ($\approx 3.5\%$ prior) eventually converged after a 40-step exploration plateau, smaller models with lower priors, like Qwen3-1.7B, ($\approx 0.5\%$ prior) failed entirely. With rewards so sparse, hitting the correct sequence by chance was statistically improbable under sparse optimization (Figure 2a).

Model	SFT+	Base	SFT-
Qwen3-8B	42.8%	0.00%	0.00%
Qwen2-7B	28.75%	3.52%	0.00%
Qwen3-1.7B	14.04%	0.48%	0.00%
Qwen2-1.5B	7.96%	0.31%	0.00%

(a) Pre-RL probabilities.

Model	SFT+	Base	SFT-
Qwen3-8B	100.0%	0.00%	0.00%
Qwen2-7B	99.6%	99.9%	0.00%
Qwen3-1.7B	98.7%	10.0%	0.00%
Qwen2-1.5B	92.5%	0.71%	0.00%

(b) Post-RL probabilities (sparse rewards).

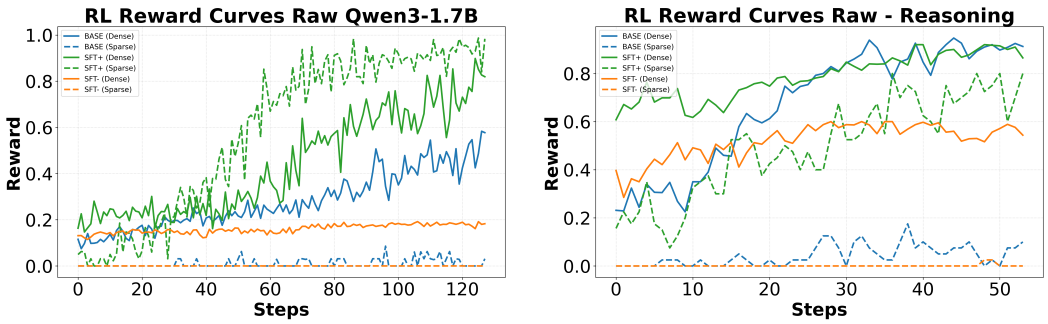
Table 1: Probability of producing the target quote before and after RL post-training, (eval = 10,000 samples).

We observe the same exploration bottleneck in our mathematical reasoning task (Table 2). Evaluating Qwen2.5-7B-Instruct on AIME Problem 4 with a sparse reward reveals similar behavior. Models injected with the correct derivation (SFT+) quickly converged (∼85% exact match), while the unmodified Base model stalled near 10%. Because the complete reasoning chain was unlikely under the base distribution, it was rarely sampled during exploration, leaving the model with almost no positive learning signal. This confirms that the Coverage Principle governs real-world RLVR: without sufficient initial probability mass on complete reasoning trajectories, sparse optimization stalls.

Configuration	SFT+	Base	SFT-
No Reward	26.6%	3.92%	0.00%
Sparse Reward	85.9%	10.2%	0.00%
Dense Reward	86.7%	92.2%	0.00%

Table 2: Probability of correctly solving AIME Problem 4 before and after RL post-training on Qwen2.5-7B-Instruct (eval = 128 samples).

5.2 Dense Reward Shaping: Beyond Pass@k



(a) Qwen3-1.7B, movie quote

(b) Qwen2.5-7B-Instruct, AIME

Figure 2: Overlay of sparse vs. dense learning dynamics. RL learning curves for **(a)** Qwen3-1.7B on the movie quote task; **(b)** Qwen2.5-7B-Instruct on the AIME task. Both figures show an example of how dense rewards can successfully guide the model to learn a behavior with very low probability in the base model, which it is not possible to learn with sparse rewards.

A prevailing assumption in recent alignment literature is that RL post-training cannot formulate new behaviors, but rather strictly upweights the existing "pass@k" capabilities of the base model, acting merely as a selector rather than a driver of novel behavior (Yue et al., 2025). Our next experiment challenges this assumption, by testing if providing the model with sufficiently dense yet accurate rewards can enable learning new behaviors.

Target string generation. We first experiment in our sequence-generation sandbox, by replacing the binary success metric with a dense Levenshtein distance reward to provide the model with intermediate gradient signals based on structural proximity to the target. Figure 2a shows learning curves for the Qwen3-1.7B model trained with both sparse and dense rewards. Despite the base model possessing a marginal $\approx 0.5\%$ base probability of producing τ, with post-training failing under standard sparse rewards, the dense signal successfully guided the model out of failure, pulling its final match rate to nearly 50%. However, as in classical RL we find that reward shaping has its limits; the SFT+ model trained with dense rewards underperforms the sparse variant in absolute string matching performance, likely because the Levenshtein distance does not sufficiently incentivize producing the *exact* target string over close approximations.

RLVR reasoning. We then assess if these results replicate for more complex reasoning tasks, testing whether dense rewards provided through a PRM can overcome base model exploration bottlenecks on an AIME mathematical problem. As Figure 2b shows, when optimizing with standard sparse rewards on this task, the base reasoning model failed to meaningfully improve, flat-lining at a match rate of approximately 10% (see Table 2).

By contrast, introducing the dense reward radically altered the learning dynamics. With exponentially increasing dense rewards for correctly reaching each step of the reasoning problem, the Base model's exact match rate jumped to over 90%. It even slightly outperformed the artificially inflated SFT+ model, which plateaued around 85% under both reward types. Qualitative analysis of the Base model's generated trajectories reveals that the dense reward successfully guided the model to combine disparate, partial reasoning steps into novel, valid solution paths that differed from the original SFT+ ground truth. This shows that RLVR can achieve complex generalization, provided the reward signal is granular enough to credit intermediate logic.

The SFT- reasoning experiments also highlight an interesting nuance about partial learning. While the heavily penalized SFT- model achieved a 0% exact match rate under both sparse and dense rewards, tracking its trajectory revealed that under the dense reward, its average reward climbed steadily to approximately 0.6 (Figure 2b). This indicates that even when a policy is actively suppressed from reaching the final correct answer, a dense reward allows it to re-learn through iteratively building on intermediate reasoning milestones.

Taken together, these results demonstrate that the limitations often attributed to RLVR—such as the inability to move beyond pass@k priors—are not inherent flaws of RL post-training, but rather artifacts of the exploration bottleneck caused by sparse, binary reward functions.

5.3 Prompt Distribution: Broad vs. Narrow

Qwen family. Figure 3 show the effects of training on spurious (random) rewards, with both the narrow prompt distribution of Shao et al. (2025), and the broad distribution. The narrow distribution replicates the original results, showing that the capabilities of Qwen models can actually increase with random rewards. However, when switching to a broad prompt distribution, we see minimal improvements on the MATH benchmark, and degraded performance on AMC. This is supported by tracking the entropy through training (Figure 4). Under the narrow distribution, entropy remains low through training, while under the broad distribution, it jumps at the first step and remains high.

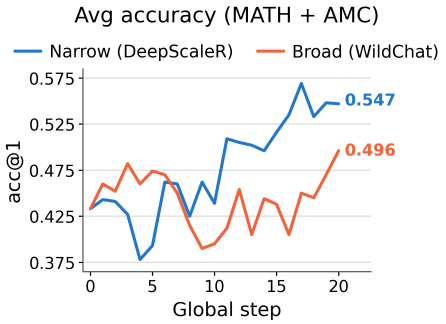
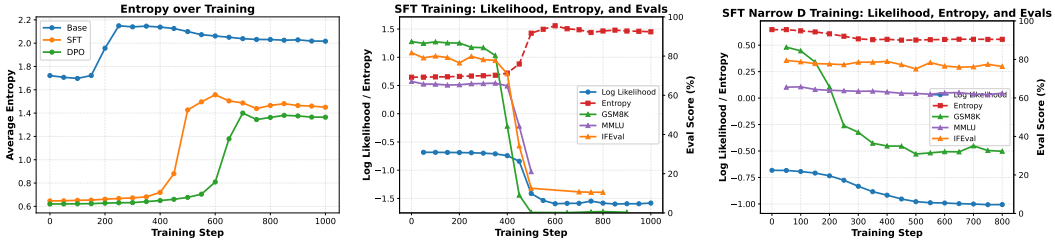


Figure 3: Acc@1 on the MATH and AMC datasets after training on narrow and broad $\mathcal{D}$; training on narrow $\mathcal{D}$ yields higher accuracy.

OLMo family. Figure 4 shows the effect of post-training the OLMo family of models (base, SFT, and DPO) on spurious rewards. Under a broad prompt distribution $\mathcal{D}$, we see that the entropy of the models increases (Fig. 4a), and that this increase at step $\sim$400 for the SFT model corresponds



(a) Broad distribution: entropy (b) Broad distribution: capabilities (c) Narrow distribution

Figure 4: Effect of training OLMo models on spurious (random) rewards). **(a)** Average entropy on broad $\mathcal{D}$ (10k prompts). Each training stage (Base $\rightarrow$ SFT $\rightarrow$ DPO) delays entropy increase onset, showing increasing resilience. **(b)** Broad $\mathcal{D}$ SFT model—entropy spike at step $\sim$400 coincides with collapse of all eval metrics (GSM8K, MMLU, IFEval), confirming global unlearning. **(c)** Narrow $\mathcal{D}$ SFT model (100 math prompts)—entropy *decreases*, MMLU and IFEval are preserved, but GSM8K collapses, showing targeted corruption of the training domain.

to a sharp decrease in capabilities, as measured with the GSM8K, MMLU, and IFEval benchmarks (Fig. 4b). Notably, log likelihood decreases and entropy increases uniformly across all three domains (math, code, instruction following), with no generalization gap. This indicates the unlearning is global: random rewards on a broad $\mathcal{D}$ effectively reward any response to any prompt, erasing the model's learned distribution.

Under a narrow $\mathcal{D}$ (100 math-only prompts), the dynamics differ strikingly. As shown in Figure 4c, entropy *decreases* slightly across all domains, indicating the model's distribution is tightening rather than dispersing. However, this increased certainty does not mean improved performance: GSM8K drops from 86% to $\sim$32% by step 400, while MMLU ($\sim$65%$\rightarrow$62%) and IFEval ($\sim$79%$\rightarrow$77%) remain largely intact. Damage is domain-specific: random rewards corrupt the training domain while preserving capabilities in unsampled domains.

The breadth of $\mathcal{D}$, as well as the base model's distribution, determine both the scope of degradation, and whether degradation occurs. A broad $\mathcal{D}$ causes widespread unlearning. A narrow $\mathcal{D}$ increases certainty about the responses the base model was already likely to produce, which can either lead to performance improvement or targeted corruption, depending on the base model's initial accuracy. Although a narrow $\mathcal{D}$ uses fewer examples with more repetition, this repetition is exactly what drives the policy to collapse toward its prior. Neither setting creates new capabilities. Qwen's gains under a narrow $\mathcal{D}$ fit this explanation: the same targeted sharpening that harms GSM8K performance in OLMo 3 instead reinforces the correct math reasoning paths already present in Qwen. This suggests that spurious rewards improve performance only when the prompt distribution is narrow and the base model is already strongly biased toward the target domain, a restricted setting that does not apply to most of RL post-training.

6 Conclusion

In this work, we deconstructed the mechanics of RL post-training for LLMs to explain how components of the post-training algorithm—the base model distribution, prompt distribution, and reward structure—affect optimization outcomes. By isolating the components of the RLVR pipeline within a controlled sequence-generation sandbox, we explain, empirically verify, or provide a counterpoint to existing post-training results in the literature. In particular, we show that standard sparse RL is unable to surpass the base model's initialization limits. When a target behavior lacks sufficient baseline coverage, the model cannot sample it, making optimization untenable. However, we show that this is not a hard and fast limitation of RL post-training as a technique; a sufficiently dense and accurate reward function can enable learning new behaviors with vanishing support in the base model, a finding that contradicts a popular position in the literature Yue et al. (2025); Shao et al. (2025); Wu et al.; Zhao et al. (2025); Wu et al. (2025). Furthermore, we show the effect of "spurious rewards" depends entirely on the post-training prompt distribution. Training with random rewards on a broad prompt distribution leads to an increase in model entropy and a corresponding catastrophic decrease in capabilities. By clarifying these interacting mechanics, we hope this work can serve as a resource to the NLP community and those looking to sharpen their understanding of the mechanics of RL post-training, and help move the field beyond treating RL as a "black box" toward a more controlled and interpretable optimization process.

Limitations

While our controlled framework introduces certain constraints, these simplifications are precisely what enable our core insights. To accurately measure probability shifts, we restrict our experiments to sequence-generation sandboxes and a single combinatorial math problem, bypassing the complexity of full-scale multi-turn dialogue. Our PRM is similarly idealized, avoiding the vulnerabilities typically encountered when scaling verifiers across broad domains (i.e. domain dependence, reward hacking, poor generalization, etc.). We also exclude algorithmic variation to strictly isolate the effects of the base distribution, prompt dataset, and reward function. Despite this restricted scope, removing confounding variables allows us to track optimization dynamics with a level of precision that would be impossible in full-scale LLM post-training. Ultimately, this sandbox approach demystifies the black-box nature of RL algorithms to help researchers, specifically those in the NLP community, design more effective and interpretable post-training pipelines.

AI USE STATEMENT

We have not used generative AI tools for generating synthetic data sets or help develop theoretical models or conceptual frameworks, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for aiding and polishing writing and identifying relevant literature. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we will make the complete post-training pipeline and evaluation codebase publicly available at the release of the camera ready version. Detailed descriptions of the open-weight base models and the public prompt datasets utilized across all experimental phases are outlined in Section 4 and C. Complete mathematical formulations and pseudo-code for our customized reward functions—including the sparse length-penalized reward, the dense Levenshtein distance proxy, and the Process Reward Model (PRM) logic—are provided in Appendices B and C. Finally, a full accounting of the computational specifications, algorithm configurations, and training hyperparameters required to replicate the experimental environments is documented in Appendices D, and E.

7 RESOURCE DETAILS

7.1 COMPUTATIONAL SPECIFICATIONS

All experiments were run on either NVIDIA A100 80GB GPU or NVIDIA L40s 48GB. For the Math reasoning task, an NVIDIA A100 80GB GPU was used for the main experiments, while an NVIDIA L40s 48GB was used for the PRM LLM-as-a-Judge.

REFERENCES

Ömer Faruk Akgül, Rajgopal Kannan, Willie Neiswanger, and Viktor Prasanna. Rethinking rl for llm reasoning: It's sparse policy selection, not capability learning. *arXiv preprint arXiv:2605.06241*, 2026.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Fan Chen, Audrey Huang, Noah Golowich, Sadhika Malladi, Adam Block, Jordan T. Ash, Akshay Krishnamurthy, and Dylan J. Foster. The coverage principle: How pre-training enables post-training, 2025. URL https://arxiv.org/abs/2510.15020.

Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots, 2023. URL https://arxiv.org/abs/2309.14341.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks, 2023. URL https://arxiv.org/abs/2306.13831.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control, 2017. URL https://arxiv.org/abs/1611.02796.

Magnus Jørgenvåg, David Kaczér, Lasse Ruttert, Marvin Gülhan, Lucie Flek, and Florian Mai. Reinforcement learning amplifies emergent misalignment from harmless rewards. *arXiv preprint arXiv:2605.31328*, 2026.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S. Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms, 2025. URL https://arxiv.org/abs/2510.13786.

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. URL https://arxiv.org/abs/2411.15124.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Neel Rajani, Aryo Pradipta Gema, Seraphina Goldfarb-Tarrant, and Ivan Titov. Scalpel vs. hammer: Grpo amplifies existing capabilities, sft replaces them. *arXiv preprint arXiv:2507.10616*, 2025.

Yi Ren and Danica J. Sutherland. Learning dynamics of llm finetuning, 2025. URL https://arxiv.org/abs/2407.10490.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning, 2024. URL https://arxiv.org/abs/2410.08146.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in rlvr, 2025. URL https://arxiv.org/abs/2506.10947.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL https://arxiv.org/abs/2009.01325.

Sijun Tan, Michael Luo, Justin Wong, Colin Cai, Xiaoxiang Shi, William Yuan Tang, Manan Roongta, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Effective RL scaling of reasoning models via iterative context lengthening, 2026. URL https://openreview.net/forum?id=I6GzDCne7U.

OLMo Team, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alex Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lj Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3. In *Technical Report, 2025*, 2025.

Rishabh Tiwari, Aditya Tomar, Udbhav Bamba, Monishwaran Maheswaran, Heng Yang, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Reward under attack: Analyzing the robustness and hackability of process reward models, 2026. URL https://arxiv.org/abs/2603.06621.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin, 2026. *URL https://arxiv.org/abs/2507.14843*.

Fang Wu, Aaron Tu, Weihao Xuan, Heli Qi, Xu Huang, Qingcheng Zeng, Shayan Talaei, Yijia Xiao, Peng Xia, Xiangru Tang, et al. Position: The hidden costs and measurement gaps of reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2509.21882*, 2025.

Weiji Xie, Jinrui Han, Jiakun Zheng, Huanyu Li, Xinzhe Liu, Jiyuan Shi, Weinan Zhang, Chenjia Bai, and Xuelong Li. Kungfubot: Physics-based humanoid whole-body control for learning highly-dynamic skills, 2025. URL https://arxiv.org/abs/2506.12851.

Wanli Yang, Hongyu Zang, Junwei Zhang, Wenjie Shi, Du Su, Jingang Wang, Xueqi Cheng, and Fei Sun. Beyond reasoning: Reinforcement learning unlocks parametric knowledge in llms. *arXiv preprint arXiv:2605.07153*, 2026.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild, 2024. URL https://arxiv.org/abs/2405.01470.

A Expanded Related Work

Post-training with RL has proven effective for a wide range of tasks, particularly for improving reasoning capabilities and aligning LLMs with human preferences. For instance, RL with Verifiable Rewards (RLVR) has shown that models can achieve state-of-the-art performance with smaller parameter counts, such as DeepSeek-R1 (Guo et al., 2025; Lambert et al., 2025), highlighting its role in developing efficient AI. Through Reinforcement Learning from Human Feedback (RLHF), reward signals are learned from preference data, leading to improvements in instruction following and response quality (Ouyang et al., 2022; Bai et al., 2022; Stiennon et al., 2022). Across both settings, RL post-training has shown to yield strong gains on reasoning and alignment tasks, making a mechanistic understanding of its success essential.

Several recent works have surfaced seemingly contradictory findings about RL post-training. Shao et al. (2025) demonstrate that RLVR can yield substantial gains in mathematical reasoning, even when reward signals are weak, random, or negatively correlated with the actual answer. Notably, they find that spurious rewards still trigger systematic behavioral shifts, such as increased code reasoning frequency in Qwen2.5-Math, suggesting the base model's pretraining dictates the form of emergent behavior, not the reward. Moreover, they observe that these gains do not generalize consistently across different model families; for example, Qwen2.5 and OLMo2 exhibit different behavioral trends to these spurious rewards. These findings appear to challenge a core idea in RL: that improvements arise from optimizing correctly specified reward signals. Our work addresses these questions by showing that the effect of random rewards depends critically on the prompt distribution. A broad prompt distribution causes global entropy increase and capability degradation, while a narrow distribution causes targeted corruption of the training domain while preserving other capabilities. The prompt distribution thus controls the *scope* of degradation, not whether degradation occurs.

Moreover, the original base model plays a central role in determining post-training success (Yue et al., 2025). Measuring pass@k (the probability of solving a problem in k attempts), the authors find that while RL-trained models excel with small values of k, base models catch up as k increases. This suggests RLVR primarily improves sampling efficiency, refining existing behaviors rather than creating novel ones. We build on this by comparing outcomes across models with different baseline capability distributions under controlled reward conditions, and show that a sufficiently shaped reward can overcome this problem, and teach new behaviors to models that previously did not generate them with high probability.

Chen et al. (2025) introduce the "Coverage Principle" and argue that existing metrics (e.g. cross-entropy loss) are often poor predictors of post-training success, instead stating the coverage (the total probability mass the pre-trained model assigns to the set of high-quality responses) is the critical link.

This motivates our choice to treat this notion of coverage as an independent variable; by selecting base models with varying coverage over target tasks, we can test how it interacts with the effect of reward density, allowing us to disentangle these two sources of variation.

Ren & Sutherland (2025) develop a framework to illustrate why post-training often redistributes probability mass rather than creating new capabilities by decomposing changes in log-probabilities after gradient updates. While their results are restricted to Direct Preference Optimization (DPO), this perspective accounts for observed behavioral shifts such as hallucinations and highlights the complex dynamics underlying alignment and performance gains. We adopt this framing for interpreting behavioral changes, as gains in the target task likely reflect probability mass redistribution rather than the emergence of new capabilities.

B Reward Function Implementations - Movie Quotes

In our sandbox experiments, we utilize two distinct reward functions to evaluate the impact of reward shaping on RL optimization: a sparse reward (equipped with an anti-rambling penalty) and a dense Levenshtein-based reward. Let y denote the model's generated sequence and τ denote the exact target string.

B.1 Sparse Reward with Length Penalty

The sparse reward acts primarily as a binary success metric. It assigns a base reward of 1 if the target string τ is successfully generated, and 0 if the model fails to produce the target entirely.

To prevent reward hacking by "rambling", we use a length penalty term p to decrease the reward for any extra tokens generated other than the target string. Let n be the maximum token generation limit and s be the number of extra tokens generated. Then, p is defined as

$$p = \begin{cases} 0 & \text{if } s = 0 \\ \frac{s}{n} & \text{if } s > 0 \end{cases} \tag{3}$$

and the sparse reward function is defined as

$$r_{\text{sparse}}(y, \tau) = \begin{cases} \max(0.5, 1 - p) & \text{if } \tau \in y \text{ and } |y| > |\tau| \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

(Note: p can be defined as a fixed scalar penalty or a function of the excess length ($|y| - |\tau|$)). This ensures the sparse optimization baseline actively discourages rambling and aligns only the precise target behavior.

B.2 Dense Reward (Levenshtein Distance)

To address the credit assignment problem for models lacking initial probability mass, we implement a dense reward that provides continuous, intermediate gradient signals based on structural proximity to the target.

We utilize the standard Levenshtein edit distance, which calculates the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform the generated sequence into the target sequence. Formally, let y and τ be strings of lengths m and n, and let $D(i, j)$ denote the Levenshtein distance between the first i characters of y and j characters of τ, where y_i and τ_j are the i-th and j-th characters. The Levenshtein distance is defined for boundary cases as $D(i, j) = \max(i, j)$ when $i = 0$ or $j = 0$. Otherwise, it is calculated as:

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + \mathbb{1}(y_i \neq \tau_j) \end{cases} \tag{5}$$

Let $n = \max(\text{len}(y), \text{len}(\tau))$ and D be the Levenshtein distance between y and τ. Then, reward function is then defined as

$$r_{dense} = \max(0, 1 - \frac{D}{n}) \tag{6}$$

C Exponential Process Reward Model (PRM) Implementation

For our math reasoning experiment, in order to scale our analysis from structural sequence-matching (Levenshtein distance) to combinatorial mathematical reasoning, we replaced the string-based dense reward with a Process Reward Model (PRM). Mathematical reasoning requires logical verification across diverse, valid solution paths (e.g., direct factoring, quadratic formula, de-homogenization). To achieve this, we deployed a 32-billion parameter instruction-tuned model (Qwen2.5-32B-Instruct) functioning as an LLM-as-a-Judge.

The PRM evaluates the generated trajectory y at the end of each rollout, classifying the strategy used and evaluating the presence of specific, sequential logical milestones.

C.1 Logical Milestones and Reward Curve

The AIME problem utilized in our experiments ($12x^2 - xy - 6y^2 = 0$) requires a multi-step algebraic derivation. We instruct the PRM to extract a boolean vector $M(y) \in \{0, 1\}^5$, representing the successful completion of the following five milestones:

- m_1: Initiated a correct algebraic path configuration (e.g., setting up factorization).
- m_2: Derived accurate base root relationships between x and y.
- m_3: Evaluated the integer boundary constraints ($-100 \leq x, y \leq 100$) correctly.
- m_4: Calculated the correct solution count for at least one case branch.
- m_5: Identified and correctly subtracted the intersection overlap at the origin $(0, 0)$.

A critical challenge in applying linear partial credit (e.g., $+0.15$ per step) is the introduction of local minima. An optimizing policy may discover that completing three steps and halting yields a "good enough" reward, dampening the advantage gradient required to risk generating further tokens to reach the final answer. To prevent this advantage compression, we map the milestones to a reward curve. The step weights are defined as $w = [0.05, 0.05, 0.10, 0.15, 0.25]$. The base process reward is calculated as:

$$r_{\text{process}}(y) = \sum_{i=1}^{5} w_i \cdot m_i \tag{7}$$

This exponential scaling strictly enforces increasing marginal value, ensuring that advanced reasoning steps provide a stronger gradient pull than early exploratory steps.

C.2 Outcome Overrides and Defensive Penalties

To anchor the PRM to the absolute ground truth and prevent reward hacking, the final reward incorporates deterministic outcome overrides and active penalties.

Let $\tau = 117$ be the exact correct answer, and $\tau_{near} = 118$ represent the specific, known near-miss where the model successfully completes the calculus but fails to account for the origin overlap. Let extract(y) be a deterministic parsing function that extracts the integer from the final `\boxed{}` command. The unpenalized reward r_{base} is defined as:

$$r_{\text{base}}(y) = \begin{cases} 1.0 & \text{if extract}(y) = \tau \\ 0.6 & \text{if extract}(y) = \tau_{\text{near}} \\ r_{\text{process}}(y) & \text{otherwise} \end{cases} \tag{8}$$

Finally, LLM policies optimizing against dense rewards frequently attempt to "farm" tokens by redundantly repeating early logical steps that are known to yield partial credit. To counteract this, the PRM is instructed to flag cyclic reasoning with a boolean penalty marker m_{loop}. If the model loops without reaching the final exact answer, a penalty $p_{loop} = -0.3$ is applied. Furthermore, a strict formatting penalty $p_{format} = -0.5$ is applied if the trajectory fails to conclude with the `\boxed{}` delimiter.

The final dense reward is bounded at 0 and defined as:

$$r_{\text{dense}}(y) = \max\left(0,\ r_{\text{base}}(y) + p_{\text{loop}} + p_{\text{format}}\right) \tag{9}$$

By heavily penalizing redundancies and exponentially rewarding progress, this PRM structure successfully bridges the exploration gap for models with near-zero initial priors.

D RLVR ALGORITHM

Algorithm 1 RLVR Algorithm

Input: base model π_{ref}, prompt dataset D, reward function $r(x, y)$
Initialize: $\pi_\theta \leftarrow \pi_{ref}$
for train steps **do**
  Sample input: $x \sim \mathcal{D}$
  Sample output: $y \sim \pi_\theta(y \mid x)$
  Get reward: $r \leftarrow r(x, y)$
  Store $\langle x, y, r \rangle$ in buffer
  if Time to train **then**
    Update model with an RL algorithm of choice to maximize: $\mathbb{E}_{\text{buffer}}[\frac{1}{c} r(x, y) - D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})]$
  end if
end for
return π_θ

E PROMPT DISTRIBUTION EXPERIMENTAL DETAILS

Training Details: For the broad-distribution condition detailed in Section 4.3, we use a production-scale dataset comprising 10,000 prompts sampled from the OLMo 3 RLVR training mix, evenly split between mathematics, instruction following, and code. We utilize the standard OlmoRL codebase with verifiable rewards replaced by random scalars drawn uniformly from $[0, 1]$. To ensure fully on-policy training, async steps are set to 0. All experiments are distributed across three 8xH100 nodes, sampling 8 responses per prompt. The narrow-distribution condition utilizes the identical hardware and algorithmic setup, but is restricted to 100 math-only prompts fine-tuned directly from the SFT checkpoint.

Evaluation Tracking: To precisely measure entropy and unlearning dynamics, we extract 200 evaluation prompts per domain (100 that appeared during training, and 100 held-out out-of-distribution prompts). Prior to RL training, we sample a fixed response for each prompt from the base model. Throughout the random reward fine-tuning process, we evaluate the model every 50 training steps, measuring the log-likelihood and average per-token entropy assigned to these fixed baseline responses.

F MODEL SIZES

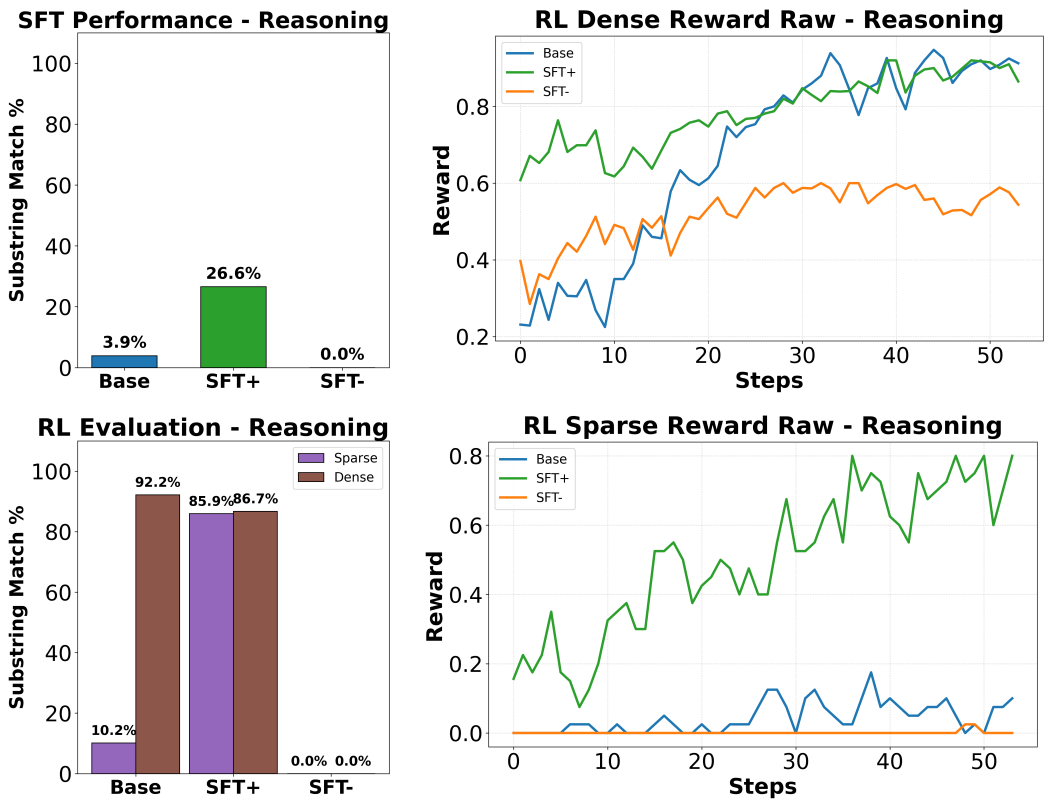


Figure 5: **RL post-training dynamics and evaluation metrics for Qwen2.5-7B-Instruct - Math Reasoning (Top Left)** Prior to RL, the SFT+ model shows a strong initial prior (26.6%), and the Base model exhibits a small but non-zero starting mass (3.9%). **(Bottom Left)** Final RL evaluation reveals that Dense rewards on both the SFT+ and Base models successfully and acquire the target behavior ($> 85\%$ match rates) however under sparse rewards the Base model was unable to acquire the target behavior while the SFT+ model acquires the target behavior. SFT- unable to gain momentum with either reward structure. **(Right)** The raw reward curves highlight a difference in the RL Training between Base, SFT+ and SFT- models. Experiments show that even through complex reasoning tasks, the priors of the models impact the role of RL post training.

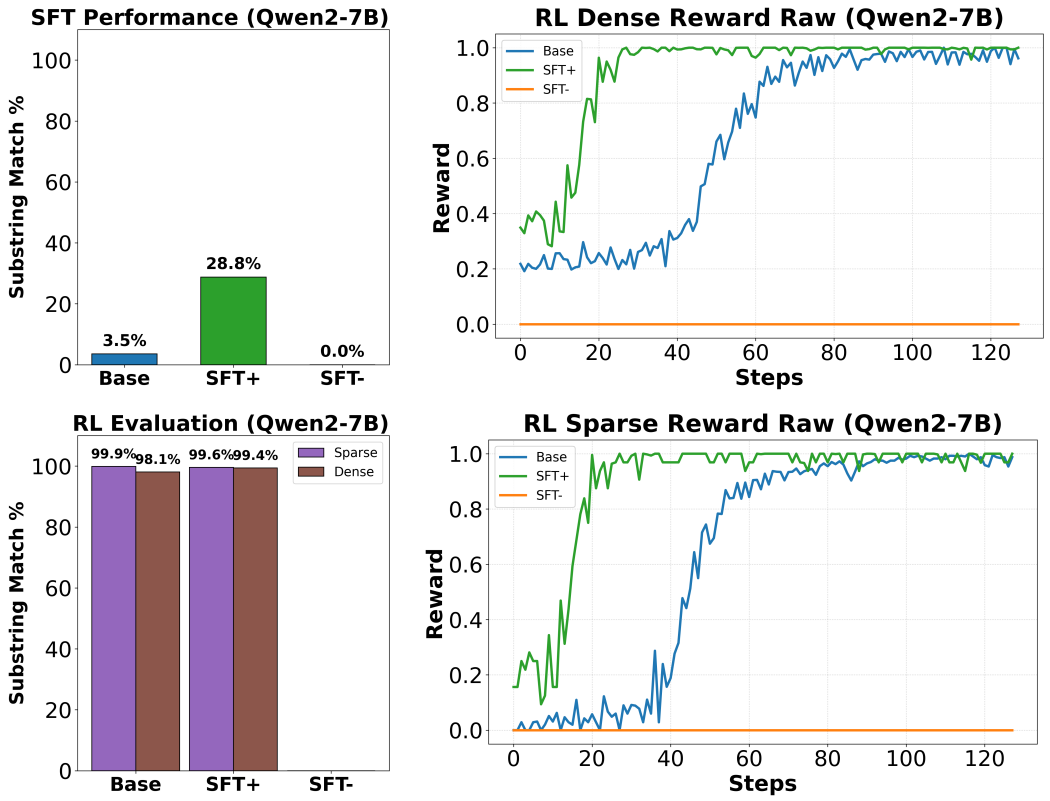


Figure 6: **RL post-training dynamics and evaluation metrics for Qwen2-7B. (Top Left)** Prior to RL, the SFT+ model shows a strong initial prior (28.8%), and the Base model exhibits a small but non-zero starting mass (3.5%). **(Bottom Left)** Final RL evaluation reveals that both the SFT+ and Base models successfully and perfectly acquire the target behavior ($> 98\%$ match rates) under both dense and sparse rewards, while SFT- completely fails. **(Right)** The raw reward curves highlight a delayed but successful optimization for the Base model. While SFT+ climbs immediately, the Base model requires roughly 40 steps to escape its initial plateau before rapidly converging to a 1.0 reward. The SFT- model remains strictly suppressed.

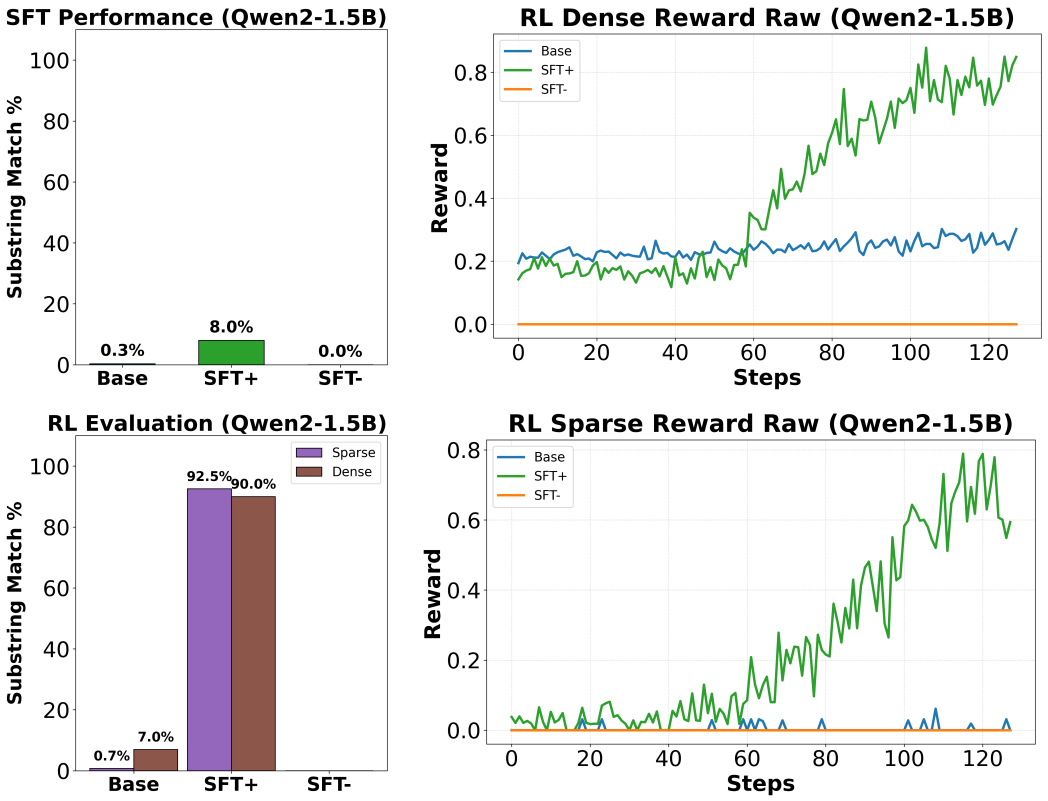


Figure 7: **RL post-training dynamics and evaluation metrics for Qwen2-1.5B. (Top Left)** Initial substring match performance prior to RL shows the SFT+ model possesses only marginal probability mass for the target sequence (8.0%), while Base (0.3%) and SFT- (0.0%) exhibit near-zero coverage. **(Bottom Left)** Final RL evaluation demonstrates that only the SFT+ model successfully acquires the target behavior (achieving $> 90\%$ match rates under both reward regimes). The Base model gains a slight benefit from dense rewards (7.0%) but largely fails, alongside SFT-. **(Right)** Raw reward curves confirm these dynamics: the SFT+ model successfully optimizes both signals, while the Base and SFT- variants flatline, failing to escape their initial exploration plateaus.

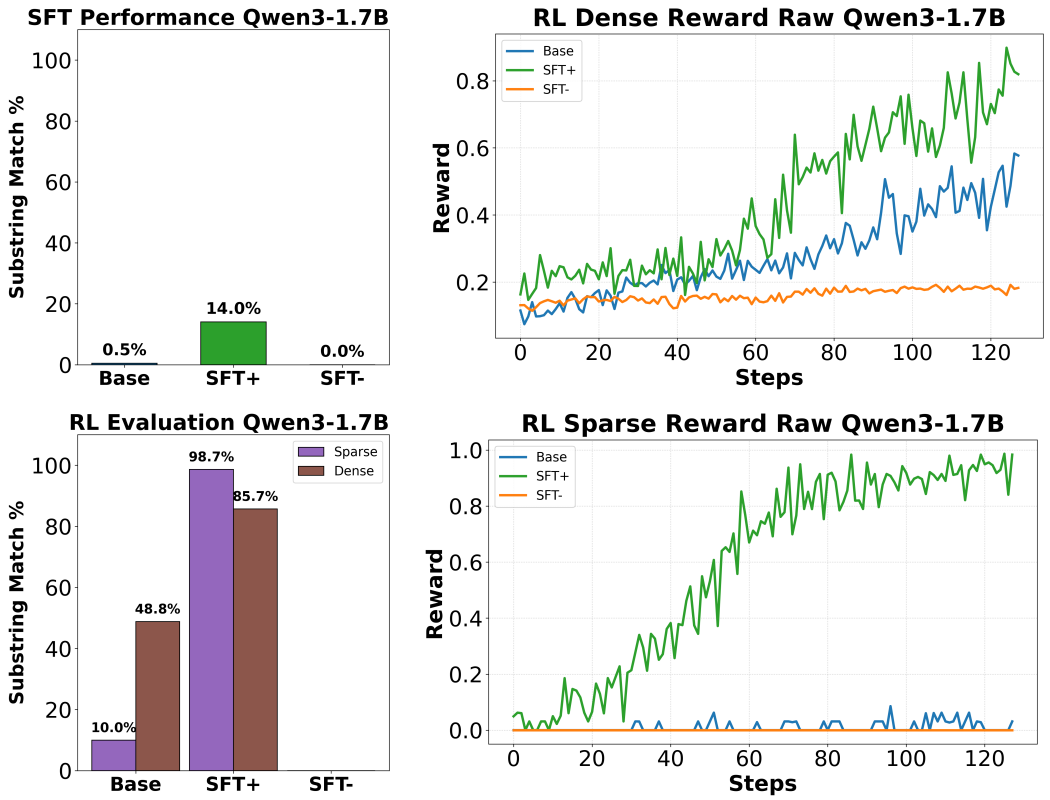


Figure 8: **RL post-training dynamics and evaluation metrics for Qwen3-1.7B. (Top Left)** Initial coverage is limited, with SFT+ at 14.0% and Base/SFT- near zero. **(Bottom Left)** Final evaluation demonstrates the specific power of dense rewards. While the Base model only achieves a 10.0% match rate under sparse rewards, the granular gradient of the dense reward allows it to reach 48.8%. The SFT+ model succeeds broadly across both signals. **(Right)** The reward curves explicitly show this dynamic: under dense rewards, the Base model successfully begins climbing (reaching $\sim$0.5), whereas under sparse rewards, it remains completely flat at 0. SFT- fails entirely across all regimes.

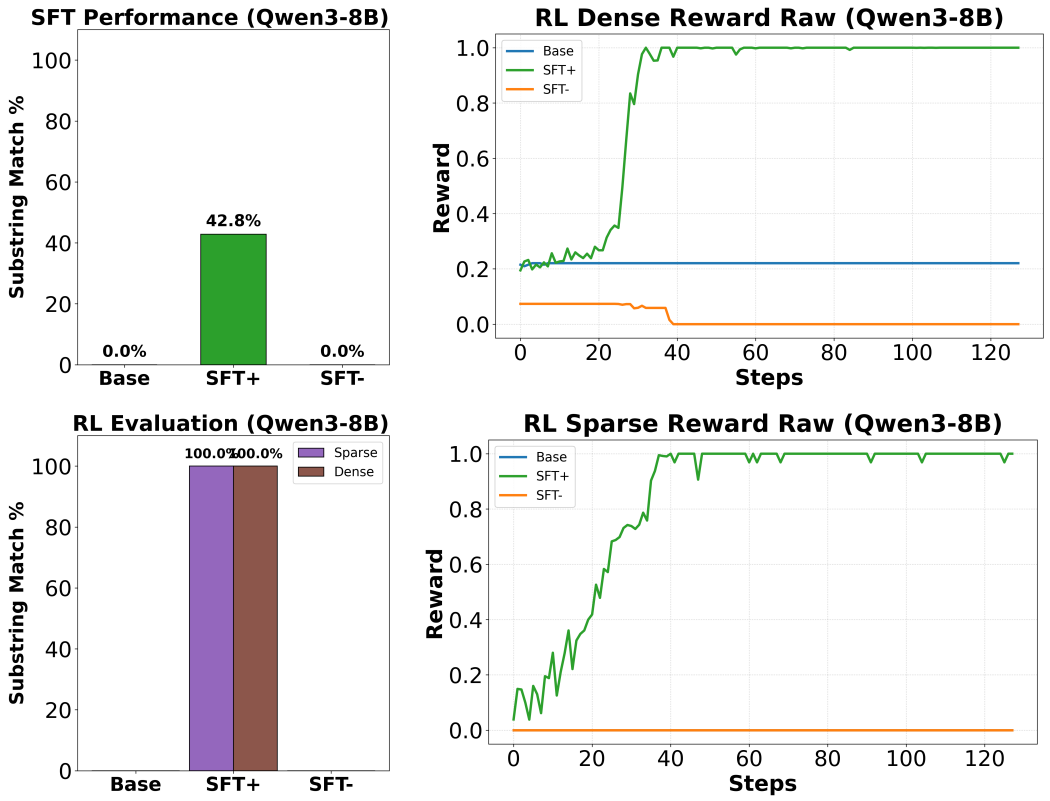


Figure 9: **RL post-training dynamics and evaluation metrics for Qwen3-8B. (Top Left)** The SFT+ model starts with a highly accessible prior (42.8%), while the Base and SFT- models start with exactly 0.0% probability mass. **(Bottom Left)** Final evaluation shows a stark binary outcome: the SFT+ model perfectly converges to a 100.0% match rate under both sparse and dense rewards, while the Base and SFT- models achieve absolute 0%. **(Right)** The raw reward curves demonstrate rapid, smooth convergence for SFT+ under both regimes. In contrast, the Base and SFT- models remain completely flat, unable to generate the target sequence a single time during exploration, confirming the strict bottleneck imposed by a 0% base prior.