FROM VERIFIABLE DOT TO REWARD CHAIN: HARNESSING VERIFIABLE REFERENCE-BASED REWARDS FOR REINFORCEMENT LEARNING OF OPEN-ENDED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) succeeds in reasoning tasks (e.g., math and code) by checking the final verifiable answer (i.e., a verifiable *dot* signal). However, extending this paradigm to open-ended generation is challenging because there is no unambiguous ground truth. Relying on single-dot supervision often leads to inefficiency and reward hacking. To address these issues, we propose reinforcement learning with verifiable reference-based rewards (**RLVRR**). Instead of checking the final answer, RLVRR extracts an ordered linguistic signal from high-quality references (i.e, reward chain). Specifically, RLVRR decomposes rewards into two dimensions: *content*, which preserves deterministic core concepts (e.g., keywords), and *style*, which evaluates adherence to stylistic properties through LLM-based verification. In this way, RLVRR combines the exploratory strength of RL with the efficiency and reliability of supervised fine-tuning (SFT). Extensive experiments on more than 10 benchmarks with Qwen and Llama models confirm the advantages of our approach. RLVRR (1) substantially outperforms SFT trained with ten times more data and advanced reward models, (2) unifies the training of structured reasoning and open-ended generation, and (3) generalizes more effectively while preserving output diversity. These results establish RLVRR as a principled and efficient path toward verifiable reinforcement learning for general-purpose LLM alignment.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024; Yu et al., 2025a; Team, 2025) has emerged as a promising paradigm for enhancing large language models (LLMs) in reasoning tasks such as mathematics and code generation. At its core, RLVR sidesteps the complicated Chain-of-Thought (CoT) supervision and only checks the correctness of the final reasoning result (i.e., the verifiable *dot*) within the reasoning solution. The presence of unambiguous ground truth makes such a verifiable dot a reliable signal, guiding exploration toward correct CoTs while preventing drift into spurious reasoning paths.

While RLVR is simple yet effective for reasoning tasks (e.g., math and code generation), it fails in open-ended generation tasks, where no unambiguous ground truth exists and reliable verification cannot be reduced to a single dot. In many cases, high-quality responses in open-ended generation should satisfy a list of content requirements simultaneously; for instance, a safe-response policy answer should explain the risk, refuse the harmful request, cite the relevant rule, and offer a safer alternative. In practice, researchers often resort to reinforcement learning from human feedback (RLHF) (Christiano et al., 2017; Bai et al., 2022; Ouyang et al., 2022) using preference-based reward models (Liu et al., 2024; 2025a) or generative reward models (Jia et al., 2025; Gunjal et al., 2025). Despite their widespread adoption, reward models suffer from two major limitations: (1) they are prone to reward hacking, often overfitting superficial artifacts and spurious correlations (Chen et al., 2024); (2) they require large-scale pairwise annotations, making training costly and brittle during RL optimization. This motivates a critical research question: *how can we extend RL optimization to open-ended generation by moving beyond single-dot supervision?*

To this end, we introduce **RLVRR** (reinforcement learning with verifiable reference-based rewards), a framework that extends RLVR to open-ended generation. Instead of relying on a single verifiable *dot*, RLVRR extracts an ordered sequence of verifiable linguistic signals from high-quality references, transforming the *dot* supervision into a *reward chain*, akin to how mathematical reasoning derives rules from ground truth. A *reference* is a high-quality exemplar for the same prompt, which can be drawn from synthetic instruction-following corpora (e.g., OpenHermes, Magpie, WebR) (Teknium, 2023; Xu et al., 2025; Jiang et al., 2025) at scale and low cost. Mechanistically, RLVRR mirrors the single-dot principle: the reward chain anchors exploration to a standardized, verifiable checklist derived from the reference. To make supervision both reliable and efficient, RLVRR decomposes rewards into two complementary dimensions: content and style. The **content** reward uses reference-derived key points (e.g., key entities or keywords) to score a rollout by whether those deterministic core concepts are present, which remains flexibility in phrasing and expression; The **style** reward runs a small set of LLM-generated, verifiable Python checks on the rollout to confirm adherence to reference-specific stylistic properties (e.g., length, format). By integrating these complementary signals, RLVRR retains RL's exploratory dynamics but injects SFT-like token-level guidance, yielding lightweight reward, stable learning, and better generalization.

Comprehensive experiments across over 10 benchmarks show that: (1) RLVRR substantially outperforms SFT with $10\times$ more data, advanced reward models, and confidence-based rewards; (2) RLVRR can be effectively integrated into RLVR, unifying the training of structured reasoning and open-ended generation; (3) RLVRR eliminates loading reward models during RL training, incurring merely 0.71% computational overhead compared to random rewards. Moreover, our in-depth analyses reveal why RLVRR generalizes more effectively and confirm that it preserves output diversity despite relying on rule-based verifiers, underscoring its practical potential.

2 Related Work

Reinforcement learning with verifiable rewards. RLVR has demonstrated strong capabilities on reasoning tasks such as math and code (Shao et al., 2024; Yu et al., 2025a; Team, 2025). By leveraging deterministic verifiers like Math-Verify (Kydlíček, 2024) and SandboxFusion (Bytedance-Seed-Foundation-Code-Team et al., 2025), RLVR enables direct correctness evaluation. Building on this paradigm, recent work has extended RLVR to broader reasoning domains. For instance, (Su et al., 2025; Ma et al., 2025) train specialized LLMs as verifier models to assess whether generated responses are equivalent to reference answers. VeriFree (Zhou et al., 2025) and RLPR (Yu et al., 2025b) bypass answer verification by leveraging policy likelihood for reference answer as a reward signal. However, these methods merely conduct experiments on datasets comprising short-form answers (nearly 10 words), overlooking the challenges of open-ended generation.

Reinforcement learning for open-ended generation. A pivotal advancement in applying reinforcement learning (RL) to open-ended generation is RLHF, which leverages human preference data to train a reward model that guides policy optimization. While effective, RLHF introduces several drawbacks including high training costs and susceptibility to reward hacking (Gao et al., 2023). These challenges have spurred the development of offline methods such as Direct Preference Optimization (DPO) (Rafailov et al., 2023), which optimizes policies directly from preference data without an explicit reward model. More recently, (Chang et al., 2025) directly uses BLEU (Papineni et al., 2002) between the reference and the rollout as a reward signal for open-ended tasks. Despite its simplicity, n-gram precision metrics such as BLEU fail to capture key content aligned with human preferences, resulting in misaligned and noisy reward signals during training.

3 Methodology

We propose reinforcement learning with verifiable reference-based rewards (RLVRR), a framework designed to provide reliable, low-cost rewards for open-ended generation by leveraging **Reward Chain** extracted from reference responses. RLVRR decomposes the reward signal into **content** and **style** dimensions, each computed through rule-based verification rather than subjective model-based scoring, as illustrated in Figure 1. We first introduce the problem formalization of RLVRR in §3.1, followed by illustrating the details of content and style rewards in §3.2 and §3.3.

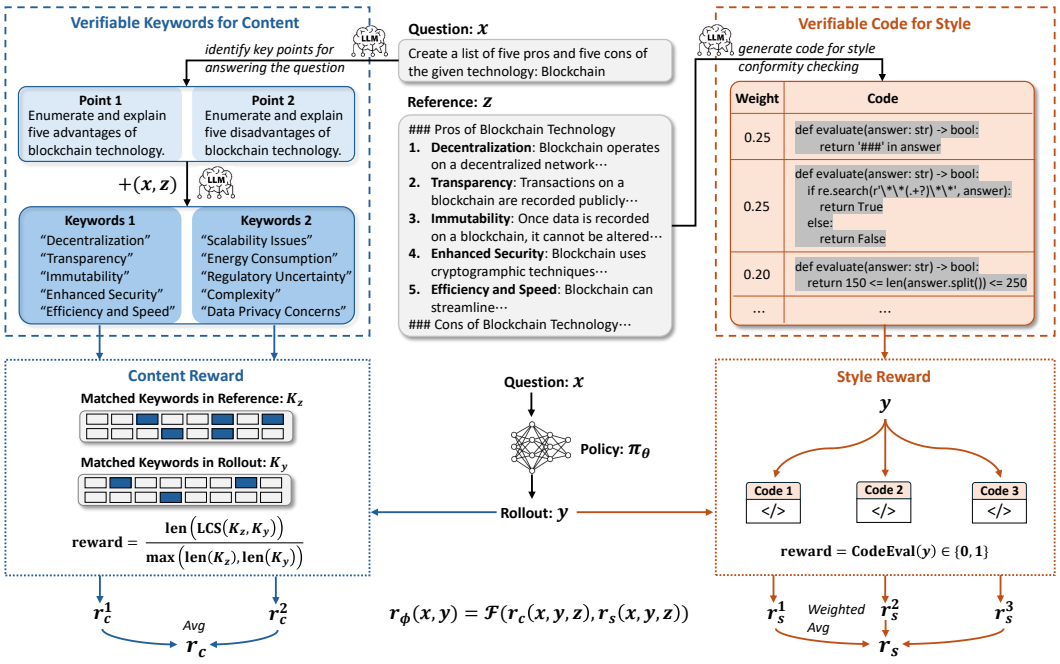

Figure 1: Overview of our proposed RLVRR framework. (1) **Upper (data construction)**: given a Question x and a Reference z, we use an off-the-shelf LLM to generate verifiable components in terms of content and style for open-ended generation. (2) **Lower (RL training)**: these verifiable components are leveraged to calculate the rule-based reward of the Rollout y.

3.1 Problem Formulation

Let x denote an open-ended instruction sampled from the training data $\mathcal{D}$. Our goal is to train a policy π_θ that generates a response y maximizing the RL objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[r_\phi(x,y)] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,\|\, \pi_{\mathrm{ref}}(y \mid x)\right], \tag{1}$$

where r_ϕ is the reward function, $\mathbb{D}_{\mathrm{KL}}$ is the KL divergence, and π_{ref} denotes the reference model. Unlike conventional RLHF methods that rely on learned reward models to instantiate r_ϕ, RLVRR derives rewards directly from **verifiable linguistic signals** based on a reference answer z:

$$r_\phi(x,y) = \mathcal{F}\left(r_c(x,y,z), r_s(x,y,z)\right), \tag{2}$$

where r_c and r_s quantify content fidelity and stylistic conformity, respectively, and $\mathcal{F}$ denotes the aggregation function (simple averaging in our experiments). Since both r_c and r_s are computed via reference-grounded rule-based reward, **RLVRR greatly mitigates the reward hacking and the inefficiency of reward models**, enabling robust and scalable RL training.

3.2 Content Reward of RLVRR

Verifiable keywords for content. Numerous studies have shown that active learning, engaging with core concepts and rephrasing information, is more effective than passive memorization (Miller, 1956; Newport, 1990). Building on this insight, we propose a novel approach to verifiable reward design: our method extracts critical keywords (or phrases) from reference responses and optimizes the policy to maximize their inclusion during reinforcement learning. Rather than directly selecting keywords that loosely capture the semantics of the reference, we propose a novel **two-level hierarchical extraction** method: (1) an LLM first identifies a set of essential *key points* $\{p^m\}_{m=1}^M$ that the AI assistant must address when answering the question (See prompt in Figure 4); (2) for each key point p^m, the LLM extracts a set of *keywords* K^m (each fewer than three words) from the reference answer that encode the core facts, concepts, or entities required to assess the correctness and relevance of the response (See prompt in Figure 5). This strategy enables broader and more systematic keyword coverage while decomposing the content reward into fine-grained, verifiable units.

As shown in Table 3, this separation significantly improves performance by 0.9 points. On average, the extracted keywords constitute approximately **15%** of the reference response, striking a balance between coverage and conciseness.

Content reward calculation. To assess content fidelity during RL training, we propose a reward function based on keyword alignment between a generated rollout y and reference text(s) z. For each key point p^m (where $m \in [1, M]$), we extract the matched keyword sequences K_y^m from y and K_z^m from z using regular expression matching. Crucially, K_y^m and K_z^m **preserve both the frequency and sequential order** of matched keywords, ensuring fine-grained alignment evaluation. For each key point p^m, we compute the semantic coherence between y and z using the longest common subsequence (LCS) metric (Wagner & Fischer, 1974). LCS is chosen because it inherently captures keyword ordering and repetition, making it well-suited for evaluating the structural and semantic fidelity of generated text. The alignment score for p^m is given by the normalized LCS length, while the overall content reward $r_c(x, y, z)$ is defined as the mean alignment score across all key points:

$$r_c(x, y, z) = \frac{1}{M} \sum_{m=1}^{M} \frac{\text{len}\left(\text{LCS}\left(K_z^m, K_y^m\right)\right)}{\max\left(\text{len}\left(K_z^m\right), \text{len}(K_y^m)\right)}. \tag{3}$$

To improve robustness and accommodate multiple references $\{z_i\}_{i=1}^I$, we extend Eq. (3) by selecting the highest alignment score per key point across all references. This ensures tolerance to variations in reference phrasing while maintaining rigorous content fidelity assessment:

$$r_c(x, y, \{z_i\}_{i=1}^I) = \frac{1}{M} \sum_{m=1}^{M} \max_i \left[\frac{\text{len}\left(\text{LCS}\left(K_{z_i}^m, K_{y_i}^m\right)\right)}{\max\left(\text{len}\left(K_{z_i}^m\right), \text{len}\left(K_{y_i}^m\right)\right)} \right]. \tag{4}$$

In RLVRR, we set $I = 3$ and show in Table 3 that multiple references consistently improve policy performance, suggesting diversified references enhance robustness.

3.3 Style Reward of RLVRR

Verifiable code for style. Unlike reasoning tasks, stylistic quality significantly influences model performance in open-ended generation tasks. To quantify stylistic alignment, we employ an LLM to generate a set of verifiable Python functions $\{\text{CodeEval}_n(\cdot)\}_{n=1}^N$, each assessing whether the rollout y adheres to stylistic properties of a reference z. These properties include answer length, markdown formatting, and other measurable features (See prompt in Figure 6). Additionally, the LLM assigns a weight w_n to each $\text{CodeEval}_n(\cdot)$, reflecting its relative importance—an approach validated empirically in our ablation study (Table 3). While our current implementation focuses on verifiable stylistic elements, semantic aspects such as tone are implicitly captured through the content reward.

Style reward calculation. During reinforcement learning, we compute the style reward $r_s(x, y, z)$ by evaluating y against each $\text{CodeEval}_n(\cdot)$ and aggregating the results as a weighted sum:

$$r_s(x, y, z) = \sum_{n=1}^{N} w_n \cdot \text{CodeEval}_n(y). \tag{5}$$

4 Experiments

4.1 Experimental Setup

Models and training data. We conduct experiments using the Qwen2.5 (Qwen Team, 2024) and Llama3.1 (Dubey et al., 2024) model series to ensure fair comparisons with prior work and enable comprehensive evaluation. For training data, we adopt the dataset released by (Jiang et al., 2025), comprising 100K open-ended instruction-response pairs curated from diverse high-quality instruction-tuning datasets. All responses are regenerated by GPT-4o-mini to maintain consistency in response quality. During the data construction of RLVRR, we also leverage GPT-4o-mini as the off-the-shelf LLM to generate verifiable components. Besides, we **cross-validate the quality of**

the verifiable components using the reference, filtering out cases where both content and style rewards of the reference fall below 0.7. Finally, we randomly sample 10K data for RL training, where GRPO (Shao et al., 2024) is applied as the optimization algorithm to ensure that all other settings are consistent with our approach.

Evaluation benchmarks. We assess our models using five of the most popular open-ended instruction-following benchmarks: AlpacaEval 2 (Li et al., 2023), Arena-Hard (Li et al., 2024), MT-Bench (Zheng et al., 2023), IFEval (Zhou et al., 2023), and FollowBench (Jiang et al., 2024). For AlpacaEval 2, we report the length-controlled win rate (LC), which ensures robustness against verbosity. For Arena-Hard, we report the win rate (WR) against the baseline model. For MT-Bench, we provide the average score, using GPT-4.1-mini as the evaluation judge. For IFEval and FollowBench, we report the prompt-level strict accuracy and the hard satisfaction rate, respectively. Besides, we evaluate the impact of diverse methods on tasks across multiple domains: (1) **Knowledge**: MMLU (Hendrycks et al., 2021a); (2) **Reasoning**: ARC (Clark et al., 2018); (3) **Math**: MATH (Hendrycks et al., 2021b); (4) **Code**: HumanEval (Chen et al., 2021). More evaluation details are listed in Appendix B.

Baselines. We compare RLVRR with seven established and contemporaneous methods, categorized into SFT, reward strategies, and DPO. (1) **SFT**: Standard supervised fine-tuning (Wei et al., 2022; Mishra et al., 2022) on (i) 10K data which shares identical prompts with RL, or (ii) 100K data. (2) **Random**: We examine whether random rewards $\sim$ Uniform$(0, 1)$ can benefit open-ended generation. (3) **BLEU**: (Chang et al., 2025) directly uses BLEU (Papineni et al., 2002) between the reference and the rollout as a reward signal for RL-based alignment. (4) **RM**: We use Skywork-Reward-V2-Llama-3.1-8B[1] (Liu et al., 2025a) trained on well-curated preference data as the reward model to score output in GRPO. (5) **GRM**: Following Rubrics as Rewards (Gunjal et al., 2025), we use GPT-4o-mini as the generative reward model to judge whether the rollout satisfies checklist-style rubrics. (6) **RLPR** (Yu et al., 2025b): RLPR is a verifier-free framework that uses the LLM's own token probability scores of reference answers as the reward signal. (7) **DPO** (Rafailov et al., 2023): We generate the preference dataset following (Meng et al., 2024). For each question x, we first generate 5 responses using the INSTRUCT model and then use GPT-4o-mini to select the best one as *win* and the worst one as *lose*. All implementation details are illustrated in Appendix A.1.

4.2 MAIN RESULTS

Table 1 summarizes the performance of various methods across five open-ended benchmarks and four additional tasks, revealing several key findings. (1) **Superiority over SFT**: Remarkably, RLVRR outperforms SFT by a significant margin on open-ended tasks, even when SFT is trained with *10× more data*. (2) **Advantages over alternative reward strategies**: RLVRR consistently surpasses other reward strategies, including random reward, BLEU, reward model (RM), generative reward model (GRM), and RLPR. Notably, it improves over the RM-based approach—which requires loading an auxiliary reward model during training—by **+2.3** and **+2.7** points on Qwen2.5-3B-Base and Instruct, respectively. (3) **Improved over DPO**: RLVRR exhibits stronger performance than DPO, a widely adopted alignment method, further validating its effectiveness. (4) **Robustness across scales and initializations**: The benefits of RLVRR persist across varying model sizes and training starting points, demonstrating its general applicability. (5) **Generalization to diverse tasks**: Beyond open-ended generation, RLVRR achieves state-of-the-art results on knowledge-intensive, reasoning, mathematical, and coding tasks, highlighting its superior generalization capability.

4.3 INTEGRATION WITH MATHEMATICAL REASONING

To examine the compatibility of our method in jointly optimizing for both closed-form reasoning and open-ended generation within RLVR, we focus on the mathematical domain as a representative setting. The reasoning template is shown in Appendix A.3. Following SimpleRL-Zoo (Zeng et al., 2025), we stratify the MATH dataset (Lewkowycz et al., 2022) into five difficulty levels and randomly sample 10K examples from levels 2–5 as the base for math-focused RL training. To explore integration, we construct a mixed training set by combining 5k math-focused samples (*using rule-based reward*) with 5k open-ended instances (*using RLVRR-based reward*). We evaluate the result-

[1]This model ranks first on RewardBench (Lambert et al., 2025) as of September 24th, 2025.

Table 1: Evaluation results across five open-ended benchmarks and four other tasks. The results of Llama3.1, which indicate consistent findings, are shown in Appendix C.

Method	#Data	Alpaca Eval 2	Arena Hard	MT Bench	IF Eval	Follow Bench	Avg.	MMLU	ARC	MATH	Human Eval	Avg.
Qwen2.5-3B Models												
Base	-	0.8	6.5	6.4	22.0	12.4	9.6	66.8	75.6	54.0	**66.5**	65.7
↪ SFT	10K	22.0	27.3	7.5	31.8	45.0	26.7	66.1	83.7	59.6	65.9	68.8
↪ SFT	100K	**25.1**	32.9	7.5	35.9	**51.3**	30.5	60.4	81.4	58.7	65.9	66.6
↪ GRPO (Random)	10K	3.7	3.6	6.1	25.7	16.1	11.0	66.9	73.7	59.7	61.6	65.5
↪ GRPO (BLEU)	10K	14.4	26.6	6.9	29.2	41.8	23.8	67.2	82.0	59.9	62.8	68.0
↪ GRPO (RM)	10K	22.4	33.7	7.3	32.8	47.6	28.8	67.1	84.2	59.6	65.1	69.0
↪ GRPO (GRM)	10K	21.1	30.5	7.4	35.4	47.3	28.3	65.5	81.2	58.2	63.9	67.2
↪ GRPO (RLPR)	10K	21.8	28.6	7.4	32.6	47.2	27.5	65.7	82.8	58.7	65.3	68.1
↪ GRPO (RLVRR)	10K	23.7	**35.3**	**7.6**	**37.7**	51.2	**31.1**	**67.9**	**85.7**	**60.6**	66.0	**70.0**
Instruct	-	17.0	19.3	7.8	54.9	47.5	29.3	67.3	84.8	63.2	71.3	71.6
↪ DPO	10K	18.0	31.1	7.6	59.3	49.6	33.1	67.1	84.8	**63.7**	69.2	71.2
↪ GRPO (RM)	10K	22.3	34.1	7.6	55.3	49.3	33.7	67.5	85.3	63.2	70.7	71.7
↪ GRPO (RLVRR)	10K	**24.3**	**36.5**	**7.9**	**61.3**	**51.9**	**36.4**	**67.8**	**85.4**	63.6	**71.9**	**72.2**
Qwen2.5-7B Models												
Base	-	2.1	8.9	7.3	24.7	14.9	11.6	74.2	79.8	69.4	76.0	74.9
↪ SFT	10K	30.0	53.2	8.3	42.3	47.5	36.3	75.2	**89.8**	67.5	77.4	77.5
↪ SFT	100K	32.3	52.0	8.3	43.5	**56.3**	38.5	70.9	87.5	67.6	76.7	75.7
↪ GRPO (Random)	10K	4.5	7.8	7.4	28.3	15.0	12.6	74.3	78.6	68.3	76.6	74.4
↪ GRPO (BLEU)	10K	19.9	44.1	7.8	39.5	46.8	31.6	74.6	83.5	68.0	77.1	75.8
↪ GRPO (RM)	10K	32.8	53.5	8.2	43.2	49.0	37.3	74.8	88.3	68.8	76.5	77.1
↪ GRPO (GRM)	10K	31.6	52.7	8.1	43.9	49.5	37.2	73.6	86.4	68.8	76.2	76.3
↪ GRPO (RLPR)	10K	31.9	51.7	8.2	42.6	49.2	36.7	72.4	87.0	67.1	76.1	75.7
↪ GRPO (RLVRR)	10K	**33.6**	**54.9**	**8.3**	**47.8**	54.6	**39.8**	**75.7**	89.6	**70.1**	**77.5**	**78.2**
Instruct	-	35.6	37.1	8.7	69.7	53.8	41.0	74.9	90.2	80.6	83.8	82.4
↪ DPO	10K	36.7	52.4	8.2	69.3	53.3	44.0	74.3	89.9	**80.9**	82.6	81.9
↪ GRPO (RM)	10K	37.6	53.6	8.4	69.1	53.9	44.5	75.1	89.5	80.2	82.8	81.9
↪ GRPO (RLVRR)	10K	**41.4**	**55.8**	**8.8**	**70.3**	**55.7**	**46.4**	**75.6**	**90.3**	80.6	**84.1**	**82.6**

Table 2: Performance comparison of math tasks based on Qwen2.5-3B-Base.

Method	GSM8K	MATH 500	Minerva Math	GaoKao 2023 En	Olympiad Bench	College Math	Avg.	Open-ended Avg.
Base	81.8	61.2	21.0	48.1	25.0	39.5	46.1	9.6
↪ GRPO - 10K math (RLVR)	86.2	**68.0**	26.1	**57.4**	**30.7**	43.2	51.9	22.6
↪ GRPO - 10K open-ended (RLVRR)	85.0	64.0	25.4	54.3	26.8	43.0	49.8	**31.1**
↪ GRPO - 5k math (RLVR) + 5k open-ended (RLVRR)	86.0	67.8	**29.4**	57.3	28.0	42.7	**51.9**	30.7
↪ GRPO - 5k math (RLVR) + 5k open-ended (RM)	84.6	67.0	24.7	56.5	27.3	42.3	50.4	28.2
Instruct	**87.0**	64.8	27.6	56.6	27.3	**45.1**	51.4	29.3

ing models on six standard mathematical reasoning benchmarks, including GSM8K (Cobbe et al., 2021), MATH 500 (Hendrycks et al., 2021b), Minerva Math (Lewkowycz et al., 2022), GaoKao 2023 En (Liao et al., 2024), Olympiad Bench (He et al., 2024), and College Math (Tang et al., 2024). We report the performance of CoT reasoning with greedy decoding.

As shown in Table 2, RLVR trained solely on mathematical data significantly boosts performance on math benchmarks but generalizes poorly to open-ended tasks (Avg. 22.6). In contrast, RLVRR trained only on open-ended data achieves strong performance in open-ended tasks and also improves mathematical reasoning (Avg. 49.8), indicating positive transfer. Unified training on mixed data provides the best balance, reaching 51.9 on math benchmarks and 30.7 on open-ended tasks. Remarkably, this setting even surpasses the INSTRUCT model trained on millions of samples, despite using only 10K RL training instances. Furthermore, RLVRR demonstrates better compatibility with reasoning tasks compared to RM. These results demonstrate that **our method seamlessly integrates with RLVR, unifying the training of structured reasoning and open-ended generation**.

5 ANALYSIS

5.1 ABLATION STUDY

To systematically evaluate method components and offer a comprehensive understanding of RLVRR, we conduct ablation studies based on Qwen2.5-3B-Base in Table 3.

Table 3: Ablation study based on Qwen2.5-3B-Base.

Method	AlpacaEval 2	Arena-Hard	MT-Bench	IF-Eval	FollowBench	Avg.
GRPO (RLVRR)	**23.7**	**35.3**	**7.6**	**37.7**	51.2	**31.1**
Effect of content reward and style reward						
-w/o content reward	9.6	10.2	6.7	28.6	35.5	18.1
-w/o multiple references	23.6	35.1	7.6	36.1	50.9	30.7
- repl. LCS with direct matching	10.3	8.5	6.5	29.0	34.7	17.8
-w/o style reward	19.6	28.5	6.6	36.6	50.1	28.3
-w/o weight in style	22.2	35.0	7.6	36.2	48.6	29.9
Effect of keyword extraction						
-w/o two-level extraction	22.6	35.3	7.5	34.3	**51.3**	30.2
- extract 15% keywords randomly	19.0	32.4	7.1	32.9	43.7	27.0
- extract 15% keywords by TF-IDF	19.6	31.3	7.4	33.3	45.5	27.4
- extract 30% keywords by TF-IDF	19.4	30.5	7.3	32.7	45.9	27.2

Table 4: Impact of various reference LLMs based on Qwen2.5-3B-Base.

Method	LLM of Ref	#Data	Open-ended Avg.
SFT	GPT-4o-mini	10K	26.7
SFT	GPT-4o-mini	100K	30.5
GRPO (RLVRR)	GPT-4o-mini	10K	**31.1**
SFT	Llama3-70B-Inst	10K	24.8
SFT	Llama3-70B-Inst	100K	28.3
GRPO (RLVRR)	Llama3-70B-Inst	10K	**28.9**

Table 5: Results of SFT via self-data distillation based on Qwen2.5-3B-Base.

Method	#Data	Open-ended Avg.
Base	-	9.6
↪ SFT	10K	26.7
↪ SFT	100K	**30.5**
↪ SFT-distilled SFT	10K	25.0
↪ RLVRR-distilled SFT	10K	29.2

Effect of content reward. Our ablation study reveals that removing the content reward results in a severe performance degradation, with the average score dropping by 13.0 points compared to the full method. This underscores the critical role of content alignment in response generation. Interestingly, when using only a single reference (instead of multiple references) for content reward computation, performance remains robust, declining marginally from 31.1 to 30.7, demonstrating the method's resilience to reference variability. Finally, we attempt to replace LCS with a naïve "direct matching" approach, which calculates the percentage of keywords appearing in the rollout as the content reward. This approach leads to catastrophic failure as it (1) disregards keyword ordering and (2) incentivizes reward hacking (Skalse et al., 2022), where the model generates excessively verbose outputs to artificially inflate keyword coverage.

Effect of style reward. The absence of style reward reduces performance by 2.8 points, confirming that learning presentation, structure, and formatting from references is essential for high-quality responses. Moreover, when style reward components are aggregated without LLM-generated importance weights, performance drops by 1.2 points, validating that LLM-derived weighting effectively captures stylistic nuances.

Effect of keywords extraction. We analyze the impact of keyword extraction strategies on alignment performance. First, we ablate the two-level hierarchical extraction process in favor of a single-step approach where keywords are directly extracted from the full response. This leads to a 0.9-point drop in average score, confirming that hierarchical extraction improves keyword precision and coverage. Next, we compare LLM-based extraction with rule-based alternatives: (1) **random selection** after stopword filtering and (2) **TF-IDF-based selection** (Sparck Jones, 1972), both extracting 15% of words for fairness. As shown in Table 3, LLM extraction outperforms both variants by 3.7–4.1 points, demonstrating its superiority in identifying semantically critical keywords. Notably, increasing the TF-IDF keyword ratio to 30% further degrades performance, suggesting that **quality matters more than quantity**—a sparse set of high-value keywords suffices for effective learning.

Effect of reference LLMs. Table 4 examines the impact of diverse reference LLMs, where the LLM is used for (1) reference generation and (2) verifiable component generation of RLVRR. Re-

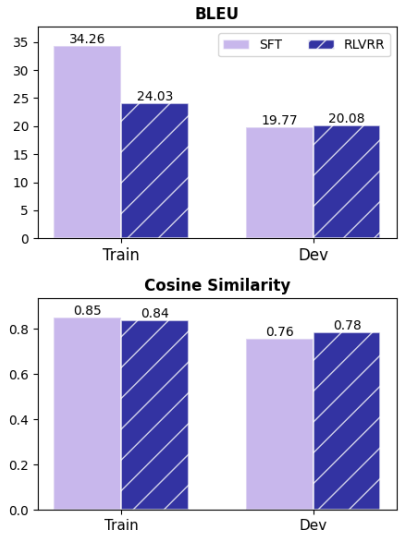

Figure 2: Average BLEU scores and semantic similarities between references and generated responses for SFT and RLVRR.

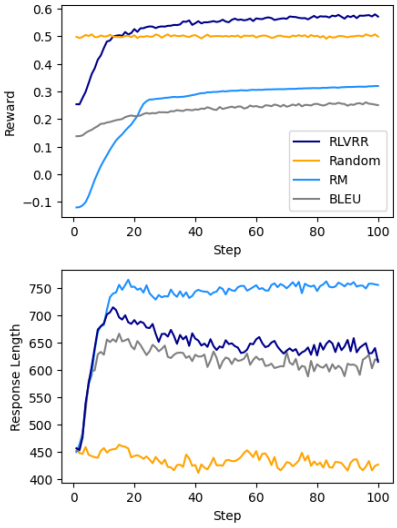

Figure 3: Curves of reward and response length during RL training with different methods.

markably, substituting the GPT-4o-mini model with a less powerful yet open-source alternative, such as Llama3-70B-Instruct (Dubey et al., 2024), yields consistent results— **RLVRR continues to outperform SFT even when SFT is trained with 10× more data**. This demonstrates the robustness of our approach across varying levels of LLM sophistication and highlights its potential to reduce reliance on proprietary commercial models without compromising downstream performance.

5.2 Learning What Matters: Why RLVRR Generalizes Better Than SFT

In this section, we investigate why RLVRR, which reinforces quality signals (keywords or phrases), outperforms SFT, which models the entire reference sequence token-by-token. To compare their generalization behaviors, we conduct a controlled study on 1,000 randomly sampled prompts, each from the training and development sets. For each prompt, we generate responses using two models trained separately with SFT and RLVRR, and evaluate their quality with BLEU and cosine semantic similarity against references. Semantic similarity is computed using embeddings from all-mpnet-base-v2 (Reimers & Gurevych, 2019). Results show that while SFT achieves higher BLEU on the training set, this advantage vanishes and even reverses on the development set, indicating strong memorization but poor generalization (Chu et al., 2025). The limitation stems from SFT's *imitation learning* objective, which minimizes token-level prediction error under teacher forcing: $\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{|z|} \log \pi_\theta(z_t|x, z_{<t})$. This training paradigm enforces exact mimicry but suffers from *exposure bias* (Zhang et al., 2019; Schmidt, 2019), as the model never recovers from its own mistakes. RLVRR, in contrast, rewards the preservation of key semantic elements while allowing flexible phrasing, leading to stable performance across both training and development sets. Specifically, RLVRR maintains consistent BLEU and higher semantic similarity (0.84 vs. 0.85 on training; 0.78 vs. 0.76 on development, compared to SFT). These results suggest that **RLVRR better captures the semantic essence of references and generalizes more effectively to unseen inputs**.

5.3 Self-Data Distilled RLVRR Outperforms Standard SFT

Recent work such as DeepSeek-R1 (DeepSeek-AI, 2025) demonstrates that fine-tuning on trajectories sampled from the same model post-RL training, an approach we refer to as *self-data distillation*, can yield better performance than standard SFT on reasoning tasks. In this section, we extend this idea to open-ended generation and examine whether a similar benefit holds. As shown in Table 5, self-data distillation using RLVRR significantly improves performance over standard SFT (2.5-point gain in average performance) when both are trained on the same 10K dataset. While it does not

Table 6: Average runtime per step for different reward strategies in RL training, based on Qwen2.5-3B-Base.

Method	Time (s)	Δ Random (%)
Random	121.56	0.00%
BLEU	122.38	0.67%
RM	131.62	8.28%
GRM	128.92	6.05%
RLPR	129.38	6.43%
RLVRR	122.42	0.71%

Table 7: Average best@5 performance and Self-BLEU cross five open-ended benchmarks, based on Qwen2.5-3B-Base.

Method	Open-ended Avg. Best@5 (↑)	Self-BLEU (↓)
Base	11.1	27.1
↪ SFT	29.8	24.2
↪ GRPO (BLEU)	25.2	26.6
↪ GRPO (RM)	30.8	23.9
↪ GRPO (RLPR)	29.6	24.5
↪ GRPO (RLVRR)	**33.2**	24.0
Instruct	31.0	**23.7**

match the performance of SFT trained on the full 100K data, it notably narrows the gap using only 10% of the data. These results highlight the superior quality of supervision signals produced by RLVRR. Moreover, since the distilled data remains close in distribution to the base model's outputs, the resulting student model benefits from both strong alignment and distributional consistency.

5.4 Training Dynamics Analysis & Cost Analysis

Training dynamics analysis. Figure 3 visualizes the curves of reward and response length during RL training with different methods (Random, BLEU, RM, and RLVRR). We observe that RLVRR achieves a more stable and substantial increase in reward compared to the other methods, highlighting its effectiveness in providing consistent and high-quality learning signals. This trend is further validated by the content/style reward curves in Figure 8. Notably, RLVRR's response length surges initially, reflecting exploratory behavior for informative outputs, then declines and stabilizes as the model learns conciseness. This demonstrates **RLVRR's robustness against reward hacking**, as it avoids exploiting length for reward gains. In contrast, RM persistently favors longer responses, likely due to over-reliance on superficial heuristics rather than true quality.

Cost analysis. We present a detailed cost breakdown of RLVRR in Appendix D, covering both the data construction and RL training phases. Key findings include: (1) the total cost of API calls during data construction is $21.36, which is highly economical given the scale of the task; (2) in the RL training phase, RLVRR introduces only a **0.71%** computational overhead compared to the Random Reward baseline (refer to Table 6). These results underscore RLVRR's practicality for real-world deployment, with minimal financial and computational burdens.

5.5 RLVRR does not Compromise Diversity

A potential concern with RLVRR's reference-based verifiable reward is that it could restrict output diversity. To examine this, we set the decoding temperature to 1.0 and sampled five responses per method across five open-ended benchmarks, reporting average best@5 and Self-BLEU in Table 7. The relative performance improvements of RLVRR over baselines remain consistent with Table 1. Notably, RLVRR attains a Self-BLEU of 24.0, comparable to RM (23.9) and the Instruct model (23.7). These findings indicate that **RLVRR does not sacrifice diversity despite its reliance on verifiable references**, and in fact, enhances the model's ability to generate diverse responses relative to other reward strategies.

6 Conclusion

In this paper, we propose RLVRR, a novel framework that extends verifiable reward learning beyond reasoning tasks to open-ended generation. By constructing rule-based verifiers derived from high-quality references across content and style dimensions, RLVRR retains RL's exploratory dynamics but injects SFT-like token-level guidance, thus providing reliable and low-cost training signals. Our results establish RLVRR as an efficient and scalable path toward verifiable reinforcement learning for general-purpose LLMs.

Reproducibility Statement

We are committed to ensuring the transparency and reproducibility of our research. To support this commitment, we will publicly release our annotated dataset and all source code, facilitating future extensions and community research. Comprehensive details of our methodology are provided throughout this paper: the prompts used for data construction are illustrated in Appendix A.2; the evaluation details are shown in Appendix B. Furthermore, the experimental implementations can be found in Appendix A.1. We believe that releasing these assets will lower the barrier for replication, enable fair comparisons, and foster further exploration in this line of research.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. In arXiv, 2022. URL https://arxiv.org/abs/2204.05862.

Bytedance-Seed-Foundation-Code-Team, :, Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, Aoyan Li, Bo Li, Bowen Li, Linyi Li, Boyi Liu, Jiaheng Liu, Kaibo Liu, Qi Liu, Shukai Liu, Siyao Liu, Tianyi Liu, Tingkai Liu, Yongfei Liu, Rui Long, Jing Mai, Guanghan Ning, Z. Y. Peng, Kai Shen, Jiahao Su, Jing Su, Tao Sun, Yifan Sun, Yunzhe Tao, Guoyin Wang, Siwei Wang, Xuwu Wang, Yite Wang, Zihan Wang, Jinxiang Xia, Liang Xiang, Xia Xiao, Yongsheng Xiao, Chenguang Xi, Shulin Xin, Jingjing Xu, Shikun Xu, Hongxia Yang, Jack Yang, Yingxiang Yang, Jianbo Yuan, Jun Zhang, Yufeng Zhang, Yuyu Zhang, Shen Zheng, He Zhu, and Ming Zhu. Fullstack bench: Evaluating llms as full stack coders, 2025. URL https://arxiv.org/abs/2412.00535.

Yapei Chang, Yekyung Kim, Michael Krumdick, Amir Zadeh, Chuan Li, Chris Tanner, and Mohit Iyyer. Bleuberi: Bleu is a surprisingly effective reward for instruction following, 2025. URL https://arxiv.org/abs/2505.11080.

Lichang Chen, Chen Zhu, Jiuhai Chen, Davit Soselia, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. ODIN: disentangled reward mitigates hacking in RLHF. In Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=zcIV8OQFVF.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Advances in Neural Information Processing Systems, pp. 4299–4307, 2017.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=dYur3yabMj.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

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

OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

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

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pp. 10835–10866. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains, 2025. URL https://arxiv.org/abs/2507.17746.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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

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.

Ruipeng Jia, Yunyi Yang, Yongbo Gai, Kai Luo, Shihao Huang, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Writing-zero: Bridge the gap between non-verifiable tasks and verifiable rewards, 2025. URL https://arxiv.org/abs/2506.00103.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 4667–4688, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.257. URL https://aclanthology.org/2024.acl-long.257.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Xinyi Dai, Yan Xu, Weinan Gan, Yasheng Wang, Xin Jiang, Lifeng Shang, Ruiming Tang, and Wei Wang. Instruction-tuning data synthesis from scratch via web reconstruction. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pp. 6603–6618. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.343/.

Hynek Kydlíček. Math-Verify: Math Verification Library, 2024. URL https://github.com/huggingface/math-verify. If you use this software, please cite it using the metadata from this file.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), Findings of the Association for Computational Linguistics: NAACL 2025, pp. 1755–1797, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.96. URL https://aclanthology.org/2025.findings-naacl.96/.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, 2024.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 2023.

Minpeng Liao, Chengxi Li, Wei Luo, Wu Jing, and Kai Fan. MARIO: MAth reasoning with code interpreter output - a reproducible pipeline. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Findings of the Association for Computational Linguistics: ACL 2024, pp. 905–924, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.53. URL https://aclanthology.org/2024.findings-acl.53/.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. arXiv preprint arXiv:2410.18451, 2024.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy, 2025a. URL https://arxiv.org/abs/2507.01352.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In Conference on Language Modeling (COLM), 2025b.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. arXiv:2505.14652, 2025. URL https://arxiv.org/abs/2505.14652.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In Advances in Neural Information Processing Systems (NeurIPS), 2024.

George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological review, 63(2):81, 1956.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), Proceedings of the 60th Annual Meeting of the Association for

Computational Linguistics (Volume 1: Long Papers), pp. 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.244. URL https://aclanthology.org/2022.acl-long.244.

Elissa L Newport. Maturational constraints on language learning. Cognitive science, 14(1):11–28, 1990.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In Advances in neural information processing systems, pp. 27730–27744, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Advances in Neural Information Processing Systems, 2023.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Florian Schmidt. Generalization in generation: A closer look at exposure bias. In Alexandra Birch, Andrew Finch, Hiroaki Hayashi, Ioannis Konstas, Thang Luong, Graham Neubig, Yusuke Oda, and Katsuhito Sudoh (eds.), Proceedings of the 3rd Workshop on Neural Generation and Translation, pp. 157–167, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5616. URL https://aclanthology.org/D19-5616/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. Advances in Neural Information Processing Systems, 35:9460–9471, 2022.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 28(1):11–21, 1972.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains, 2025. URL https://arxiv.org/abs/2503.23829.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. Mathscale: Scaling instruction tuning for mathematical reasoning. In Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=Kjww7ZN47M.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Robert A Wagner and Michael J Fischer. The string-to-string correction problem. Journal of the ACM (JACM), 21(1):168–173, 1974.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=Pnk7vMbznK.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025a. URL https://arxiv.org/abs/2503.14476.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. Rlpr: Extrapolating rlvr to general domains without verifiers, 2025b. URL https://arxiv.org/abs/2506.18254.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL https://arxiv.org/abs/2503.18892.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4334–4343, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1426. URL https://aclanthology.org/P19-1426/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

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

APPENDICES

A DETAILED EXPERIMENTAL SETUP

A.1 IMPLEMENTATION DETAILS

We adopt the OpenRLHF (Hu et al., 2024) framework for efficient training. During SFT, we train models for 3 epochs with a learning rate of 2e-5, a batch size of 128, a max sequence length of 2048, and a cosine learning rate schedule with 10% warmup steps. During GRPO, we set the epoch to 1, the learning rate to 5e-7, the number of rollouts to 8, max prompt length and max generation length to 1024 tokens, and maintain the same global batch size of 128. During DPO, we train models for 1 epoch with a learning rate of 5e-7, a batch size of 128, a max sequence length of 2048, and a β of 1e-2. All experiments are conducted on 8 NVIDIA A800 GPUs. We report the average performance of three random runs.

A.2 PROMPT TEMPLATE FOR DATA CONSTRUCTION

Prompt Template of Generating Key Points for Answering the Question

You are given a **Question**. Your task is to identify **all essential key points** that the AI assistant must notice when answering the question. Return your output as a Python list of strings, where each string is a key point.

[Question]:
{question}

Figure 4: Prompt template of generating key points for answering the question.

Prompt Template of Generating Keywords

You are given a **Question**, **Key Points of the Question**, and a **Reference Answer**. Your task is to identify **all essential keywords (less than 3 words for each keyword)** from the reference answer that match each key point. These keywords should be explicitly mentioned or accurately reflected in a good AI-generated answer. These keywords should represent the core facts, concepts, or entities required to assess the correctness and relevance of the response.

Output Format:
Please return your schema in the following JSON format:
json
```
{{
  "key_points": [
    {{
      "point": "<Key point>",,
      "keywords": <a Python list of strings>,
    }}
  ],
}}
```

[Question]:
{question}

[Key Points]:
{keypoint}

[Reference Answer]:
{reference}

Figure 5: Prompt template of generating keywords.

Prompt Template of Generating Code for Style Conformity Checking

```
You are given a **Reference Answer**. Your task is to evaluate whether an **AI-generated answer** follows a similar **format and style** as the reference answer. The goal is not to assess content correctness or completeness, but to compare the presentation, structure, and formatting features of the two answers.

### Key Evaluation Focus:
Your evaluation should focus on **style-related aspects** such as:
- Overall structure and organization (e.g., use of sections or bullet points)
- Length similarity (in terms of word count, within a reasonable range)
- Paragraph count and distribution (roughly comparable, not necessarily identical)
- Use of markdown elements like bold text, headers, lists, code blocks
- Visual layout and clarity

**Do NOT evaluate the factual content, relevance, or correctness of the answer.**

### Instructions:
1. **Identify 3–6 Key Style Evaluation Points:** For each point, define a **specific and measurable** style-related criterion. Allow for small variations instead of requiring exact matches.
2. **Define a Python Function for Each Point:** The function should be named `evaluate(answer: str) -> bool` and return:
   - `True` if the AI-generated answer satisfies the style point (even approximately),
   - `False` otherwise.
3. **Assign a Weight to Each Point:** Each point should be assigned a weight that reflects its relative importance. All weights must sum to **1.0**.

### Output Format:
Please return your evaluation schema in the following JSON format:
```json
{{
  "key_points": [
    {{
      "point": "<Key evaluation point>",
      "explanation": "<Why this point helps assess format/style similarity>",
      "verification_code": "def evaluate(answer: str) -> bool:\\n    # Your logic here\\n    return ...",
      "weight": <float weight>
    }}
  ],
  "total_weight": 1.0
}}
```

[Reference Answer]:
{reference}
```

Figure 6: Prompt template of generating code for style conformity checking.

A.3 Template for Mathematical Reasoning

Figure 7 shows the training and evaluation template for mathematical reasoning, where we first require the model to think step by step and then output the final answer within "boxed{}".

Training and Evaluation Template for Mathematical Reasoning

```
<|im_start|>user
{question}
Please reason step by step, and put your final answer within \\boxed{}.<|im_end|>
<|im_start|>assistant
```

Figure 7: Training and evaluation template for mathematical reasoning.

B Evaluation Details

Table 8 lists the evaluation details for AlpacaEval 2 (Li et al., 2023), Arena-Hard (Li et al., 2024), MT-Bench (Zheng et al., 2023), IFEval (Zhou et al., 2023), and FollowBench (Jiang et al., 2024). AlpacaEval 2 comprises 805 questions from 5 datasets, and MT-Bench spans 8 categories with a total of 80 questions. Arena-Hard is an enhanced version of MT-Bench, featuring 500 well-defined technical problem-solving queries. IFEval comprises 541 samples designed to evaluate instruction-following LLMs through diverse, verifiable instructions that include numerous lexical and formatting constraints. FollowBench is a multi-level, fine-grained benchmark for evaluating constraint-following capabilities, featuring 820 samples across five constraint types and five difficulty levels. To balance cost and performance, we select GPT-4.1-mini as the judge. Evaluation metrics are reported in accordance with each benchmark's protocol. For tasks across multiple domains, we align our evaluation settings with OpenCompass (Contributors, 2023).

Table 8: Evaluation details for AlpacaEval 2, Arena-Hard, MT-Bench, IFEval, and FollowBench. The baseline model refers to the model compared against.

Benchmark	# Exs.	Baseline Model	Judge Model	Scoring Type	Metric
AlpacaEval 2	805	GPT-4 Turbo	GPT-4.1-mini	Pairwise comparison	Length-controlled win rate
Arena-Hard	500	GPT-4-0314	GPT-4.1-mini	Pairwise comparison	Win rate
MT-Bench	80	-	GPT-4.1-mini	Single-answer grading	Rating of 1-10
IFEval	541	-	-	Rule-based verification	Accuracy
FollowBench	820	-	GPT-4.1-mini	Rule and LLM verification	Satisfaction rate

C Experimental Results of Llama3.1

Table 9 presents results on Llama3.1-8B-Instruct, as prior work shows that effective GRPO training requires a sufficiently strong base model (Liu et al., 2025b). RLVRR consistently outperforms all baselines by more than 2 points, with a comparable improvement observed on Qwen2.5. These findings confirm that **our approach generalizes robustly across different model architectures**.

Table 9: Evaluation results of Llama3.1-8B across five open-ended benchmarks and four other tasks.

Method	#Data	Alpaca Eval 2	Arena Hard	MT Bench	IF Eval	Follow Bench	Avg.	MMLU	ARC	MATH	Human Eval	Avg.
Instruct	-	30.9	34.3	8.4	76.8	54.2	40.9	69.4	83.4	51.9	72.6	69.3
↪ SFT	10K	31.2	46.9	8.5	75.6	53.2	43.1	67.7	83.7	50.6	69.9	68.0
↪ SFT	100K	33.1	51.0	8.5	75.9	55.3	44.8	65.4	81.6	51.7	70.8	67.4
↪ GRPO (Random)	10K	5.2	6.7	7.6	26.3	17.5	12.7	66.7	79.7	40.7	66.8	63.5
↪ GRPO (BLEU)	10K	30.4	39.6	8.2	69.2	49.8	39.4	69.2	82.0	50.2	72.8	68.6
↪ GRPO (RM)	10K	32.5	50.7	8.7	76.0	53.7	44.3	69.6	84.2	52.4	72.1	69.6
↪ GRPO (GRM)	10K	31.1	48.5	8.4	75.4	54.3	43.5	69.5	83.2	51.2	70.9	68.7
↪ GRPO (RLPR)	10K	30.8	48.6	8.3	74.6	54.4	43.3	68.7	82.8	**52.7**	72.3	69.1
↪ DPO	10K	32.0	48.1	8.6	77.3	53.6	43.9	69.1	83.8	51.0	72.2	69.0
↪ GRPO (RLVRR)	10K	**36.7**	**52.3**	**8.7**	**77.7**	**56.2**	**46.3**	**70.2**	**84.9**	52.6	**73.0**	**70.2**

D Cost Analysis

D.1 Cost of Data Construction

The data construction phase, responsible for synthesizing verifiable components for content and style reward, operates exclusively **offline**, meaning it incurs no runtime cost during model training. For context, we estimated the budget for data synthesis using the GPT-4o-mini API, based on the API's pricing of $0.15 per 1M input tokens and $0.60 per 1M output tokens. Table 10 lists the breakdown of the estimated costs, which demonstrates that the overall expenditure (**$21.36**) is both reasonable and manageable.

Can an open-source LLM be utilized as an alternative? In Table 4, we explore the impact of LLMs on generating verifiable components during the data construction phase. Our findings indicate that substituting the GPT-4o-mini model with a less powerful yet open-source alternative,

Table 10: Estimated budget for data construction using the GPT-4o-mini API.

Task	# of Samples	Avg. Input Token Length	Avg. Output Token Length	Cost ($)
Multiple References	20,000	170	652	8.33
Key Points	10,000	202	234	1.71
Keywords	30,000	1,156	103	7.06
Style	10,000	853	497	4.26

such as Llama3-70B-Instruct, **yields comparable performance while significantly surpassing SFT trained with 10× more data**. The Llama3-70B-Instruct model can be deployed on only 2 NVIDIA 3090 GPUs, with the option to further reduce hardware requirements through low-bit quantization[2]. This provides an economical alternative for RLVRR without compromising performance. Overall, our framework demonstrates robustness in leveraging diverse LLMs for verifiable component generation, confirming its adaptability and effectiveness.

D.2 Cost of RL Training

RLVRR incurs negligible computational overhead. As shown in Table 11, we report the average runtime per training step on 8 NVIDIA A800 GPUs across various reward strategies. RLVRR increases runtime by only **0.71%** compared to the Random Reward baseline, comparable to the lightweight BLEU-based reward (+0.67%). In contrast, RM introduces a substantial 8.28% overhead due to the need to maintain and query a learned reward model, while RLPR incurs a 6.43% increase from additional reference forward passes. These results highlight that RLVRR achieves verifiability with minimal runtime cost, making it a scalable choice for real-world RL training scenarios.

Table 11: Average runtime per step for different reward strategies in RL training.

Method	Time (s)	Δ Random (%)
Random	121.56	0.00%
BLEU	122.38	0.67%
RM	131.62	8.28%
GRM	128.92	6.05%
RLPR	129.38	6.43%
RLVRR	122.42	0.71%

E Reward Curves

Figure 8 presents the training dynamics of RLVRR in terms of **content** and **style** rewards. Both rewards exhibit a consistent upward trend in the early stages, indicating effective optimization across dimensions. Notably, the style reward plateaus after approximately 60 steps, suggesting that stylistic improvements saturate relatively quickly. In contrast, the content reward continues to increase, albeit more gradually, highlighting the model's sustained ability to refine content quality over time.

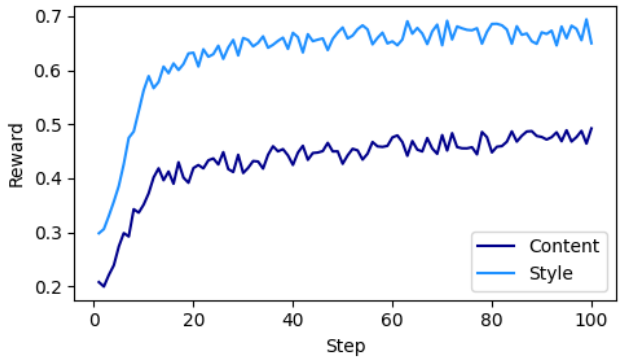

Figure 8: Content and style rewards of RLVRR during training, based on Qwen2.5-3B-Base.

[2] https://github.com/ollama/ollama

F CASE STUDY

In this case study, we analyze the performance of various methods, all based on the Qwen2.5-3B-Base model, using a sample instruction from AlpacaEval 2. Table 12 displays the responses generated by four different methods. The SFT model produces a concise and factually accurate answer, although it lacks detail and context regarding the name change. In contrast, models further trained with BLEU and RM yield incorrect responses, asserting that Facebook Corporation did not change its legal name and providing an inaccurate account of the rebranding process. Our proposed method, RLVRR, demonstrates a notable improvement by providing a response that is both factually accurate and comprehensive. Additionally, the response generated by our method is significantly shorter than those produced by BLEU and RM. This combination of detail, accuracy, and brevity highlights the superiority of our approach in delivering informative and precise answers.

Table 12: Generated responses from different methods for a sampled instruction in AlpacaEval 2.

Instruction	Did Facebook corporation change its name?
SFT	Yes, Facebook Corporation changed its name to Meta Platforms, Inc. In 2021.
BLEU	No, the Facebook Corporation did not alter its legal name. Despite the fact that the company has introduced a diverse array of products and services beyond its primary social networking platform, which includes popular applications such as Instagram, WhatsApp, Messenger, and various others, the official name of the corporation continues to be Facebook, Inc. Nevertheless, in October 2021, Facebook made an announcement indicating their intention to modify their corporate brand name to better reflect their expanding range of offerings. As of now, the new brand name has not been disclosed to the public, and it is anticipated that the complete transition to this new name will require a significant amount of time.
RM	No, Facebook Corporation did not change its name. Facebook Corporation, also known as Facebook, Inc., is the parent company of the Facebook social media platform, and it has not gone through any significant name change. The company holds other online platforms and services such as Instagram and WhatsApp, but the name Facebook Corporation remains the same. There have been some changes in the branding, such as the rebranding of WhatsApp and Facebook Messenger with the Meta logo, but the parent company's name has not changed. The announcement made by Facebook in October 2021 about rebranding its parent company as Meta did not materialize, and Facebook Corporation still holds the name Facebook Corporation.
RLVRR	Yes, Facebook Corporation changed its name to Meta Platforms, Inc. In October 2021. The rebranding aims to reflect the company's expanded focus beyond social media and into building the "metaverse," a virtual space where people can interact and communicate in immersive, 3D environments. The new name "Meta" represents the company's ambition to drive breakthroughs in areas such as virtual and augmented reality.

G LLM USAGE

We utilized large language models to support both manuscript polishing and data construction. In particular, the GPT-4o-mini API is employed to assist with the construction of the training dataset. Further details of this process are provided in Section 3 and Appendix A.2.