GRAD2REWARD: FROM SPARSE JUDGMENT TO DENSE REWARDS FOR IMPROVING OPEN-ENDED LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has catalyzed significant breakthroughs in complex LLM reasoning within verifiable domains, such as mathematics and programming. Recent efforts have sought to extend this paradigm to open-ended tasks by employing LLM-as-a-Judge to provide sequence-level rewards for policy optimization. However, such rewards are inherently sparse, failing to provide the fine-grained supervision required for generating complex, long-form trajectories. Furthermore, existing approaches treat the Judge as a black-box oracle, using only its verdict while discarding the rich intermediate feedback signals encoded in it. To address these limitations, we introduce GRAD2REWARD, a novel framework that extracts dense process rewards directly from the Judge's inference process via a single backward pass. Notably, GRAD2REWARD adopts a self-judging mechanism, allowing the policy to improve through its own evaluative signals without reliance on superior external Judges. To effectively leverage these dense rewards, we further introduce Token Advantage Policy Optimization (TAPO), which enables fine-grained policy updates. Extensive experiments demonstrate that GRAD2REWARD achieves superior performance and training efficiency across diverse open-ended tasks, affirming its effectiveness and broad applicability.

1 INTRODUCTION

Large language models (LLMs) have recently demonstrated remarkable progress in complex reasoning tasks such as mathematics and programming, as shown by OpenAI's o-series (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025). A key driver of these advances is Reinforcement Learning with Verifiable Rewards (RLVR), which assigns outcome rewards by verifying policy[1] outputs against ground-truth answers. RLVR relies on ground-truth labels and therefore cannot be directly applied to broader open-ended tasks (Xu et al., 2025; Ye et al., 2025), such as medical consultation or creative writing, where evaluation is inherently subjective and not strictly verifiable. Recent work (Gunjal et al., 2025; Zhou et al., 2025) attempts to extend RLVR to open-ended tasks by leveraging LLM-as-a-Judge to provide sequence-level rewards for policy optimization. For example, Gunjal et al. (2025) employ an LLM-as-a-judge paradigm to evaluate policy outputs along multiple criteria and convert the evaluations into sequence-level rewards for GRPO (Shao et al., 2024) optimization.

However, these methods face several limitations. (i) *Sparse rewards result in a lack of fine-grained supervision.* Sparse rewards only provide feedback at the end of the trajectory, causing different parts of the policy output to be treated equally (Zhang et al., 2025a; Cheng et al., 2025). Unlike verifiable tasks that only care about the correctness of the final answer and can tolerate errors in the reasoning process, open-ended tasks place greater emphasis on the quality of the process. For example, in medical counseling, every part of the policy output must be scientifically sound and non-misleading. This makes open-ended tasks fundamentally require dense rewards for fine-grained optimization. However, existing dense-reward methods (Wang et al., 2024; Zeng et al., 2025) require ground truth and trained reward models, limiting their applicability to open-ended tasks. (ii) *Insufficient use of Judge feedback.* In practice, the Judge contains richer information than just its sequence-level

[1]We use *policy* to represent the underlying LLM generator for the given task query.

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.

verdict. While the judge's verdict provides an overall assessment of the policy output, it is inherently informed by the different parts of the policy output, suggesting the potential to offer more fine-grained guidance beyond sequence-level supervision. Despite this, prior approaches (Shao et al., 2025; Bi et al., 2025) regard the Judge as a black box, relying solely on the verdict to derive the reward, without exploiting the additional information implicit in the Judge's assessment process. Together, these limitations highlight a fundamental challenge: fine-grained rewards are critical but challenging to obtain, while the rich feedback signals inherent in the Judge remain underutilized.

To address these limitations, we introduce GRAD2REWARD, a framework designed to unlock the dense feedback hidden within the Judge. Specifically, GRAD2REWARD leverages gradient-based attribution to quantify each token's contribution to the Judge's verdict and converts these contributions into token-level rewards. The entire procedure consumes only a single backward pass and does not require any fine-tuning of the Judge. Notably, unlike prior work (Bi et al., 2025; Zhou et al., 2025; Gunjal et al., 2025; Wang et al., 2025) that relies on more advanced models as Judge to supervise policy training, we adopt a self-judging mechanism where the Judge is fixed to the initial policy model, ensuring that policy improvement arises from its own potential rather than distilling external knowledge. To effectively exploit these dense rewards, we further introduce Token Advantage Policy Optimization (TAPO), which performs fine-grained token-level optimization and ultimately enhances the LLM's performance on open-ended tasks. Empirically, optimizing the policy with the dense rewards provided by GRAD2REWARD leads to superior training efficiency and better performance compared to sparse-reward baselines, validating both its effectiveness and generalization capability. Our contributions can be summarized as follows:

- We propose GRAD2REWARD, a framework that converts sparse judgment feedback into dense rewards, effectively addressing the credit assignment problem in open-ended tasks. By further introducing TAPO, GRAD2REWARD enables fine-grained policy optimization.
- We introduce a self-judging mechanism that allows a policy to leverage its own evaluative feedback for improvement, eliminating the need for stronger and more expensive external Judges.
- Extensive experiments show GRAD2REWARD achieves leading performance across tasks with higher training efficiency, and can be extended to verifiable domains with strong performance.

2 RELATED WORK

LLM for Open-Ended Tasks. Open-ended tasks such as medical consultation (Arora et al., 2025) or scientific question answering (Yifei et al., 2025) lack ground-truth answers, typically requiring human experts for evaluation. Recent work (Zhou et al., 2025; Shao et al., 2025; Bi et al., 2025; Huang et al., 2025) has explored using LLM-as-a-Judge (Lee et al., 2025; Wu et al., 2025) to generate feedback signals as sequence-level rewards for optimizing LLM performance through reinforcement learning (RL). For example, Gunjal et al. (2025) uses a Judge to evaluate whether the policy output satisfies specified rubrics and converts the evaluation into rewards. However, the rewards used in these methods are sparse and therefore cannot provide fine-grained supervision over the policy's generated response, a limitation that is particularly pronounced in open-ended tasks.

Dense Reward Modeling. Dense rewards have proven effective for improving LLM reasoning (Lightman et al., 2024; Wang et al., 2024). Recent work (Zeng et al., 2025; Li & Li, 2025) has mainly explored training PRMs in mathematical domains, which provide step-level rewards (Zhang et al., 2025a) or token-level rewards (Cui et al., 2025) for policy optimization. For example, PQM (Li & Li, 2025) models the process reward as a Q-value ranking problem in a Markov Decision Process. PURE (Cheng et al., 2025) first trains a PRM, and then applies it to the proposed min-form credit assignment. However, these methods require ground-truth process labels to train a reward model, making them applicable only to verifiable domains and hard to extend to open-ended tasks.

3 PRELIMINARY

3.1 OPEN-ENDED LLM REASONING

Given a query x, the LLM policy π generates a response $o = (a_1, a_2, \ldots, a_T)$, where a_t denotes the t-th token and T is the sequence length. We model the generation process of the LLM as a finite-

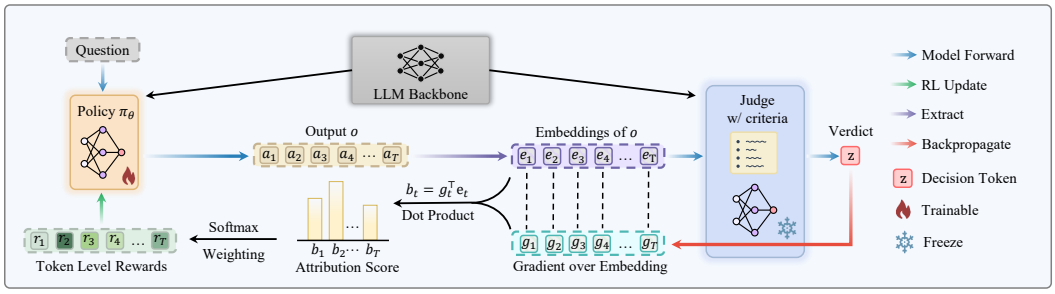

Figure 1: **Overview of GRAD2REWARD** (better viewed in color): Policy samples an output o, which is evaluated by a Judge to derive a verdict z. By computing the inner product between the output embeddings e_t and their gradients g_t (derived from z), we obtain token-wise attribution scores b_t. These are converted into token-level rewards r_t to guide the policy optimization.

horizon Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r)$ [2], in which the autoregressive LLM π serves as the policy. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ denotes the transition dynamics, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. For the t-th token, the state is defined as $s_t = (x, a_{\leq t-1})$ which contains the query x and the sequence of previously generated tokens $a_{\leq t-1}$. The action a_t is the token generated conditioned on state s_t. The state transition is deterministic, as the next state is uniquely determined by concatenating the previously generated sequence with the current token a_t. Let r_t denote the immediate reward of token a_t, such that the sequence-level reward of the policy output o can be written as $r(x, o) = \Sigma_{t=1}^{T} r_t$.

Notably, in open-ended tasks, prior approaches obtain sequence-level reward $r(x, o)$ by using the LLM-as-a-Judge, while the individual token rewards r_t remain unknown.

3.2 LLM-AS-A-JUDGE

To evaluate a policy output o for a given query x, existing methods (Shao et al., 2025; Bi et al., 2025) employ the LLM-as-a-Judge combined with a predefined, query-specific rubric $\mathcal{R}(x) = \{(c_k, w_k)\}_{k=1}^{K}$, where each c_k denotes a evaluation criterion and w_k is the corresponding weight.

Specifically, a structured prompt (see Appendix G) is constructed, which includes the query x, the policy output o, and an individual criterion c_k as input to the Judge. For each criterion c_k, the Judge produces a binary verdict $z_k \sim p_{\text{judge}}(\cdot \mid x, o, c_k)$ with $z_k \in \{\texttt{True}, \texttt{False}\}$, indicating whether the criterion is satisfied. Based on these verdicts, the sequence-level reward for (x, o) is computed as an aggregation over all criteria:

$$r(x, o) = \frac{\sum_{k=1}^{K} w_k \cdot \text{sgn}(z_k)}{\sum_{k=1}^{K} |w_k|} \tag{1}$$

where $\text{sgn}(z_k) = 1$ if $z_k = \texttt{True}$ and -1 otherwise. The sequence-level reward $r(x, o)$ is then used in RL optimization, such as GRPO (Shao et al., 2024), to improve the policy's performance.

4 METHODOLOGY

4.1 PROCESS FEEDBACK THROUGH SELF-JUDGING

Although the Judge ultimately evaluates the policy output o with a binary verdict, its internal computation is far more expressive, as the Judge must assess different parts of the policy output, including its logical structure, semantic coherence, and criterion alignment. Therefore, the final verdict can be viewed as a compressed outcome of an internal evaluation process over the entire sequence. However, existing approaches (Zhou et al., 2025; Gunjal et al., 2025; Shao et al., 2025) treat the Judge as a black box and only extract the final verdict as a *scalar* reward. This collapses the Judge's rich internal evaluative information into a single binary signal, implicitly assigning equal credit to all

[2]The discount factor is typically set to $\gamma = 1$ in LLM settings and is therefore omitted from the formulation (Zhang et al., 2025a; Cheng et al., 2025).

tokens in the trajectory. As a result, a large amount of intermediate feedback that may indicate the quality of the generation processes is discarded. This motivates our design to recover and exploit the process feedback that is implicitly embedded in the Judge's internal computation but overlooked by existing approaches.

In this work, we adopt a **Self-Judging** mechanism, in which the Judge is instantiated as a frozen copy of the initial policy, rather than a stronger external model. During training, the Judge remains fixed and only provides evaluation signals, while the policy is optimized via RL. This design has three motivations. First, unlike prior methods (Bi et al., 2025; Gunjal et al., 2025; Wang et al., 2025) that rely on stronger and potentially expensive external Judges, our design eliminates this dependency, improving scalability and practicality. Second, it is motivated by the observation that *evaluation is often easier than generation for language models* (Song et al., 2025; Zhao et al., 2026; Li et al., 2026). Thus, the model can improve by exploiting the gap between its own evaluative and generative capabilities, enabling a form of self-improvement, rather than distilling knowledge from a superior model. Third, an alternative self-judging setting is to update the Judge jointly with the policy during training. However, this requires repeatedly refreshing Judge weights, introducing reloading and synchronization overhead. Since the two settings achieve comparable performance, we adopt the frozen-Judge design for simplicity and efficiency, with detailed analysis in Appendix D.

4.2 Credit Assignment via Gradient-Based Attribution

Although the Judge provides only a final decision token, its verdict is generated conditioned on the entire policy output and thus implicitly depends on each token. To recover this hidden token-level feedback, we need to estimate how much each token contributes to the Judge's verdict. Tokens with a larger influence on the verdict should receive larger credit. This motivates our use of gradient-based attribution, which extracts fine-grained contribution signals from the Judge without additional training. Concretely, let $\mathbf{e}_t \in \mathbb{R}^d$ denote the embedding of the t-th token in the policy output $o = (a_1, \ldots, a_T)$. We compute the gradient of the log-probability that the Judge generates the specific decision verdict z with respect to each token embedding [3]:

$$\mathbf{g}_t = \nabla_{\mathbf{e}_t} \log p_{\text{judge}}(z \mid x, o, c) \tag{2}$$

Gradient vectors $\mathbf{g}_t$ are next transformed into scalar attribution scores that reflect the importance of each token for the Judge's decision using the following method, a principled attribution strategy whose theoretical foundation is elaborated in Section 4.3.

$$b_t = \mathbf{g}_t^\top \mathbf{e}_t \tag{3}$$

Eq. 3 yields a sequence of attribution scores $\mathbf{b} = (b_1, \ldots, b_T)$, where each b_t reflects the contribution of the t-th token in policy output o to the Judge's decision. We apply a softmax function across token positions to normalize the attribution scores and use the resulting weights to decompose the sequence-level reward $r(x, o)$ into *token-level* rewards:

$$r_t = \alpha_t \cdot r(x, o), \qquad \alpha_t = \text{Softmax}(\mathbf{b})_t \tag{4}$$

As a result, the originally sparse sequence-level supervision is transformed into dense token-level feedback, enabling more precise credit assignment through the gradient signals implicitly encoded in the Judge's internal computation.

4.3 Theoretical Analysis of the Reward Design

In this subsection, we provide a theoretical perspective on the attribution strategy. Let function F represent the Judge's log-probability of generating the decision token for criterion c given (x, o):

$$F(\mathbf{e}_{1:T}) = \log p_{\text{judge}}(z \mid x, o, c) \tag{5}$$

where $\mathbf{e}_{1:T} = (\mathbf{e}_1, \ldots, \mathbf{e}_T)$ denotes the token embeddings of the policy output o. Consider local perturbations $\Delta\mathbf{e}_t$ to each token embedding and apply a Taylor expansion around the Judge's input token embeddings to obtain an approximation of the sequence-level objective.

$$F(\mathbf{e}_1 + \Delta\mathbf{e}_1, \ldots, \mathbf{e}_T + \Delta\mathbf{e}_T) \approx F(\mathbf{e}_{1:T}) + \sum_{t=1}^{T} \mathbf{g}_t^\top \Delta\mathbf{e}_t + \frac{1}{2} \sum_{t=1}^{T} \sum_{s=1}^{T} \Delta\mathbf{e}_t^\top \mathbf{H}_{t,s} \Delta\mathbf{e}_s + O(\|\Delta\mathbf{e}\|^3) \tag{6}$$

[3] For simplicity, we omit the rubric index k in the derivations. It can be restored in the full formulation.

where $\mathbf{g}_t = \nabla_{\mathbf{e}_t} F(\mathbf{e}_{1:T})$ is the gradient of the Judge objective with respect to the t-th token embedding, and $\mathbf{H}_{t,s} = \nabla^2_{\mathbf{e}_t,\mathbf{e}_s} F(\mathbf{e}_{1:T})$ denotes the (t,s)-th Hessian block.

Following prior work (Sundararajan et al., 2017; Shrikumar et al., 2017; Chrysostomou & Aletras, 2022; Nguyen et al., 2025), we set the perturbation as $\Delta\mathbf{e}_t = -\mathbf{e}_t$. The underlying intuition is that attribution is defined relative to the absence of a cause, making the zero baseline a natural reference point for measuring contribution. This choice is also well justified in practice. Token embeddings are typically approximately zero-centered and concentrated within a relatively narrow magnitude range. Consequently, the perturbation induced by $\Delta\mathbf{e}_t = -\mathbf{e}_t$ stays within a local neighborhood of the original embedding, which helps reduce higher-order truncation errors and improves the reliability of the Taylor approximation in practice. We provide empirical magnitude statistics in Appendix C.2. Moreover, we adopt the zero baseline for simplicity and clarity of exposition. Nevertheless, more informed baselines based on the distribution of token embeddings, such as embedding centroids, can also be considered. Our framework can seamlessly accommodate such data-dependent alternatives with minimal modifications. With $\Delta\mathbf{e}_t = -\mathbf{e}_t$, the above equation can be rewritten as follows:

$$F(\mathbf{e}_{1:T}) - F(\mathbf{0}_{1:T}) \approx \sum_{t=1}^{T} \mathbf{g}_t^\top \mathbf{e}_t - \frac{1}{2} \sum_{t=1}^{T} \sum_{s=1}^{T} \mathbf{e}_t^\top \mathbf{H}_{t,s} \mathbf{e}_s + O(\|\mathbf{e}\|^3) \tag{7}$$

where $\mathbf{0}_{1:T} = (\mathbf{0}, \ldots, \mathbf{0})$.

Following Ma et al. (2023), we adopt a block-diagonal approximation and use the Fisher information matrix $\mathcal{F}_t$ to approximate each Hessian block. A detailed derivation is provided in Appendix C.1. The second-order term reduces to

$$\sum_{t=1}^{T} \sum_{s=1}^{T} \mathbf{e}_t^\top \mathbf{H}_{t,s} \mathbf{e}_s \approx -\sum_{t=1}^{T} \mathbf{e}_t^\top \mathcal{F}_t \mathbf{e}_t \approx -\sum_{t=1}^{T} \mathbf{e}_t^\top \mathbf{g}_t \mathbf{g}_t^\top \mathbf{e}_t = -\sum_{t=1}^{T} (\mathbf{g}_t^\top \mathbf{e}_t)^2 \tag{8}$$

Therefore, substituting the above approximation into Eq. 7 yields

$$F(\mathbf{e}_{1:T}) - F(\mathbf{0}_{1:T}) \approx \sum_{t=1}^{T} \left(\mathbf{g}_t^\top \mathbf{e}_t + \frac{1}{2} (\mathbf{g}_t^\top \mathbf{e}_t)^2 \right) + O(\|\mathbf{e}\|^3) \tag{9}$$

Compared with the first- and second-order terms, the higher-order remainder is expected to have a much smaller effect and is therefore neglected. Each term $\left(\mathbf{g}_t^\top \mathbf{e}_t + \frac{1}{2}(\mathbf{g}_t^\top \mathbf{e}_t)^2\right)$ measures the contribution of token a_t to the Judge's decision, while the sum over all tokens approximates the total change in the Judge's output relative to the zero baseline. Therefore, it can naturally be used as a token-level attribution score.

Empirically, however, we find that the attribution score derived from the first-order Taylor approximation, $b_t = \mathbf{g}_t^\top \mathbf{e}_t$, and the attribution score derived from the second-order Taylor approximation, $b_t = \mathbf{g}_t^\top \mathbf{e}_t + \frac{1}{2}(\mathbf{g}_t^\top \mathbf{e}_t)^2$, yield comparable performance (see Appendix C.3 for empirical results). This can be explained by the fact that the first-order term already captures the dominant effect of each token on the Judge objective, while the second-order term provides only a relatively small correction to the attribution score. As a result, the first-order approximation is often sufficient to preserve the relative importance across tokens. Therefore, for simplicity and computational efficiency, we adopt the first-order attribution score in the main method, as instantiated in Eq. 3.

The sequence-level reward defined in Section 3.2 is derived from the Judge's verdict, while the attribution scores estimate each token's contribution to that verdict. This naturally provides a basis for decomposing the sequence-level reward into token-level rewards, allowing Judge feedback to be distributed according to token contributions rather than assigned uniformly.

4.4 Token Advantage Policy Optimization

Common RL optimization methods such as GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), and RLOO (Ahmadian et al., 2024) rely on sequence-level rewards, where all tokens within a generated response share the same advantage signal. This leads to a coarse-granularity issue (Sun et al., 2025). To fully exploit the token-level rewards provided by our Grad2Reward, we introduce Token Advantage Policy Optimization (TAPO), which enables fine-grained optimization at the token level.

Algorithm 1 Training Procedure of GRAD2REWARD

Require: Policy π_θ, Judge p_{judge}, dataset $\mathcal{D}$, rubric set $\mathcal{R}$, group size G

Sample query $x \sim \mathcal{D}$
Rubric for x: $\mathcal{R}(x) = \{(c_k, w_k)\}_{k=1}^{K}$ ▷ *Section 3.2*
Generate a group of responses $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$
for $i = 1$ to G **do**
  for $k = 1$ to K **do**
    Obtain Judge decision $z_k \sim p_{\text{judge}}(\cdot \mid x, o_i, c_k)$ via self-judging ▷ *Section 4.2*
    Compute token-level gradients: $\mathbf{g}_{k,t} = \nabla_{\mathbf{e}_t} \log p_{\text{judge}}(z_k \mid x, o_i, c_k)$ ▷ *Eq. 2*
    Convert gradients to attribution scores: $b_{k,t} = \mathbf{g}_{k,t}^{\top}\mathbf{e}_t, \quad \alpha_{k,t} = \text{softmax}_t(b_{k,t})$ ▷ *Eq. 3*
  end for
  Compute token rewards: $r_{i,t} = \dfrac{\sum_{k=1}^{K} w_k \,\text{sgn}(z_k)\, \alpha_{k,t}}{\sum_{k=1}^{K} |w_k|}$ ▷ *Combining Eq. 1 and Eq. 4*
end for
Compute token-level advantages $\hat{A}_{i,t}$ and update π_θ via TAPO ▷ *Eq. 10 and Eq. 11*

For a given input query x, the policy generates G responses $\{o_i\}_{i=1}^{G}$, where each response $o_i = (a_{i,1}, \ldots, a_{i,T})$ is associated with token-level rewards $\{r_{i,t}\}_{t=1}^{T}$ computed using Eq. 4. The return $R_{i,t}$ aggregates future token rewards from position t to the end of the response. To compute token-level advantages, we compute the mean and standard deviation globally over all token returns, rather than separately at each token position. This avoids unstable position-wise estimates, since generated responses can have different lengths, and later token positions naturally contain fewer samples. Such reduced sample counts at later positions can lead to inaccurate baselines and noisy advantage estimates. Similar practices have also been supported by prior studies (Cui et al., 2025; Cheng et al., 2025). We then compute the token-level advantage $\hat{A}_{i,t}$ as:

$$\hat{A}_{i,t} = \frac{R_{i,t} - \text{mean}\Big(\{\, R_{j,s} \,\}_{j=1,s=1}^{G,\,T}\Big)}{\text{std}\Big(\{\, R_{j,s} \,\}_{j=1,s=1}^{G,\,T}\Big)}, \quad R_{i,t} = \sum_{s=t}^{T} r_{i,s} \tag{10}$$

Appendix E further justifies our return and advantage design through theoretical analysis and empirical results. Therefore, the optimization objective is:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min\Big(\rho_{i,t}(\theta)\, \hat{A}_{i,t},\, \text{clip}(\rho_{i,t}(\theta),\, 1-\epsilon,\, 1+\epsilon)\, \hat{A}_{i,t} \Big) \right] \tag{11}$$

where $\rho_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|x, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|x, o_{i,<t})}$ is the token-level importance ratio between the current policy and the previous policy, and ϵ is the clipping coefficient.

Figure 1 presents the overall workflow of our method, and Alg. 1 outlines the step-by-step procedure of GRAD2REWARD. Notably, since each query is typically associated with K rubric items (c_k, w_k) that define evaluation criteria from multiple perspectives, we explicitly incorporate this practical setting into Alg. 1, although it is abstracted away in the preceding method description for simplicity.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTINGS

Datasets. For the medical consultation domain, we employ the HealthBench and RaR-Medicine datasets. For academic question answering, we use ResearchQA and RaR-Science. Each query in these datasets is associated with multiple rubric items, which define query-specific evaluation criteria for assessing the quality of LLM-generated responses. Each dataset is split into a training set and a test set, with completely disjoint queries and rubric items between the two splits. A detailed description of these datasets is provided in Appendix A.1.

Policy Training and Baselines. We train policy models of different parameter scales, including Qwen2.5-1.5B-Instruct and Qwen3-4B-Instruct, using full-parameter fine-tuning. Detailed training

Table 1: Main results comparison across open-ended tasks. The best results are highlighted in **bold**.

Model	Qwen2.5-1.5B-Instruct				Qwen3-4B-Instruct			
	Health Bench	RaR-Medicine	Research QA	RaR-Science	Health Bench	RaR-Medicine	Research QA	RaR-Science
Base Model	8.9	20.7	24.1	28.9	37.2	53.7	63.0	68.1
Vanilla-GRPO	13.2	25.5	31.0	30.9	39.9	56.5	72.8	69.3
RuscaRL	13.7	26.6	32.2	30.6	41.7	58.5	**75.6**	68.8
GRAD2REWARD (ours)	**15.0**	**29.3**	**36.7**	**33.7**	**43.9**	**61.0**	75.3	**69.5**

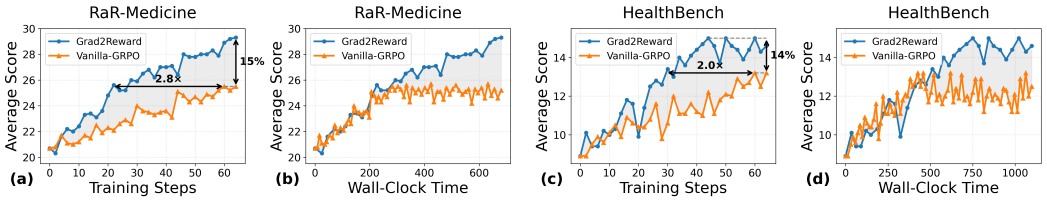

Figure 2: Training efficiency comparison between GRAD2REWARD and Vanilla-GRPO across training steps and wall-clock time (minutes).

settings are provided in Appendix A.2. We compare GRAD2REWARD with two sparse-reward baselines: *Vanilla-GRPO*, which optimizes the policy using the sequence-level reward defined in Eq. 1, and *RuscaRL* (Zhou et al., 2025), which also uses sequence-level rewards but leverages rubrics to guide RL exploration, representing a leading approach for optimizing LLMs on open-ended tasks.

Evaluation. To ensure fair and reliable evaluation, we adopt the OpenAI Simple-Evals suite[4], which computes rubric-based scores and supports pluggable grader models. For testing, we use gpt-oss-20b as the primary test grader to compute the average score. Notably, the test grader is a stronger external model than the Judge used during training and is employed solely to assess the performance of the trained policy. We employ different prompt templates for Judge used in training and the test grader model for testing, as detailed in Appendix G.

5.2 MAIN RESULTS

Finding 1: GRAD2REWARD *consistently performs well across different policy models and tasks.* Table 1 reports the main results on four open-ended benchmarks using two policy models, Qwen2.5-1.5B-Instruct and Qwen3-4B-Instruct. Specifically, for Qwen2.5-1.5B-Instruct, GRAD2REWARD outperforms the strong baseline RuscaRL by 4.5 points on ResearchQA. This indicates that dense supervision effectively provides more informative optimization signals for smaller models, helping compensate for their limited reasoning capacity. With Qwen3-4B-Instruct, our GRAD2REWARD achieves a score of 61.0 on RaR-Medicine, significantly outperforming Vanilla-GRPO (56.5) and RuscaRL (58.5). Overall, these results demonstrate that GRAD2REWARD provides robust and scalable improvements, highlighting the effectiveness of converting sparse judgment into dense rewards.

Finding 2: GRAD2REWARD *achieves higher training efficiency with moderate computational overhead.* Figure 2 compares the training dynamics of GRAD2REWARD and the sparse-reward baseline Vanilla-GRPO on RaR-Medicine and HealthBench in terms of both training steps and wall-clock time. On RaR-Medicine, GRAD2REWARD matches Vanilla-GRPO's final performance with 2.8× fewer training steps and ultimately achieves a 15% relative gain. Similarly, on HealthBench, GRAD2REWARD attains the same performance level as Vanilla-GRPO with about 2.0× fewer training steps and achieves a 14% relative gain at the end of training. When measured by wall-clock time, the improvement is more gradual because gradient-based attribution introduces an additional backward pass. As a result, GRAD2REWARD is comparable to, and occasionally slightly slower than, Vanilla-GRPO in the early training stage. However, this overhead remains moderate: within a similar training-time budget, GRAD2REWARD continues to improve and converges to a stronger policy. Overall, GRAD2REWARD trades a modest computational overhead for substantially improved sample efficiency and final performance, making dense reward extraction a practical and effective strategy for open-ended RL optimization.

[4]https://github.com/openai/simple-evals

Table 2: Ablation on attribution strategies.

Method	Health Bench	RaR-Medicine	Research QA	RaR-Science
Vanilla-GRPO	13.2	25.5	31.0	30.9
L1 norm	12.7	25.2	31.8	29.1
L2 norm	13.1	26.6	33.9	30.7
GRAD2REWARD	**15.0**	**29.3**	**36.7**	**33.7**

Table 3: Ablation on self-judging mechanism.

Method	Training-Time Judge	Self-Judging	Attribution	Health Bench	Research QA
Vanilla-GRPO	Qwen2.5-1.5B-Instruct	✓	✗	13.2	31.0
GRAD2REWARD	Qwen2.5-1.5B-Instruct	✓	✓	15.0	36.7
Variant-1	Llama-3.2-3B-Instruct	✗	✓	13.6	35.9
Variant-2	Qwen2.5-7B-Instruct	✗	✓	15.3	36.6
Variant-3	Qwen3-32B	✗	✓	16.3	38.3

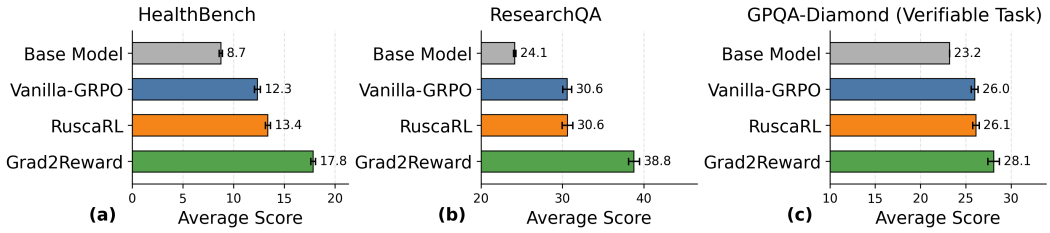

Figure 3: Generalization Analysis. The policy trained on RaR-Medicine is evaluated on (a) HealthBench, while that trained on RaR-Science is evaluated on (b) ResearchQA and (c) GPQA-Diamond.

5.3 ABLATION STUDY

Finding 3 (Ablation on Attribution Strategies): GRAD2REWARD *enables more reliable credit assignment than other attribution strategies.* We compare our attribution strategy with two commonly used attribution methods in the NLP community, namely the L_1 and L_2 norms of gradients (Yang et al., 2025). Table 2 reports the results using Qwen2.5-1.5B-Instruct as both the policy and the training-time Judge. Implementation details are deferred to Appendix A.3. As shown, GRAD2REWARD consistently achieves the best performance across benchmarks. The L_1 and L_2 gradient norms yield weaker and less stable improvements. These results are consistent with our analysis in Section 4.3: GRAD2REWARD arises from the first-order Taylor expansion of the Judge's decision function and estimates each token's contribution to the Judge's verdict. In contrast, norm-based metrics only measure gradient magnitude and discard directional information, making them less aligned with the optimization objective.

Finding 4 (Ablation Study on Self-Judging Mechanism): *Self-Judging provides sufficiently informative feedback without relying on a stronger external Judge.* Table 3 compares different training-time Judges while fixing the policy model to Qwen2.5-1.5B-Instruct. As shown, the self-judging setup achieves performance comparable to variants that use larger external Judges. It outperforms the Llama-3.2-3B-Instruct Judge on both benchmarks, and remains close to the Qwen2.5-7B-Instruct Judge. Although using the stronger Qwen3-32B Judge yields additional gains, the improvement is relatively moderate given its substantially larger model size. This indicates that GRAD2REWARD does not critically depend on a stronger external Judge. Instead, it can effectively leverage the policy's own evaluative capability for self-improvement.

5.4 EXTENDED ANALYSIS

We conduct a more in-depth analysis of our GRAD2REWARD by investigating the following research questions. **RQ1**: How well does GRAD2REWARD generalize? **RQ2**: Are the performance gains of GRAD2REWARD robust under different test graders for open-ended evaluation? **RQ3**: Can GRAD2REWARD be extended to other RL algorithms? Unless otherwise specified, we use Qwen2.5-1.5B-Instruct as the policy model throughout the following analyses.

Finding 5 (Generalization Analysis): GRAD2REWARD *exhibits stronger generalization on both open-ended and verifiable tasks (**RQ1**).* Figure 3 evaluates generalization under two settings: training on RaR-Medicine and testing on HealthBench, and training on RaR-Science and testing on both ResearchQA and GPQA-Diamond (Rein et al., 2024). Notably, although trained only with an open-ended dataset, GRAD2REWARD also achieves the strongest result on GPQA-Diamond, a verifiable scientific reasoning benchmark. These results suggest that GRAD2REWARD's dense rewards promote transferable reasoning across datasets and task formats.

Table 4: Reliability analysis with different test graders across multiple independent training runs.

Test Grader	Qwen3-30B-A3B-Instruct				Mistral-Small-3.2-24B-Instruct			
Method	HealthBench	RaR-Medicine	ResearchQA	RaR-Science	HealthBench	RaR-Medicine	ResearchQA	RaR-Science
Base Model	29.77 ± 0.21	27.87 ± 0.12	39.03 ± 0.06	35.03 ± 0.21	27.47 ± 0.21	29.83 ± 0.06	40.87 ± 0.06	47.77 ± 0.06
Vanilla-GRPO	36.90 ± 0.56	33.20 ± 0.46	51.27 ± 1.02	41.23 ± 0.74	31.90 ± 0.61	34.17 ± 0.31	48.60 ± 0.61	51.07 ± 0.76
RuscaRL	38.63 ± 0.25	35.33 ± 1.10	51.77 ± 1.33	40.97 ± 0.42	32.27 ± 0.72	35.07 ± 0.06	49.37 ± 0.29	50.50 ± 0.89
GRAD2REWARD	**40.23 ± 0.51**	**36.77 ± 0.49**	**58.03 ± 0.38**	**42.90 ± 0.26**	**33.53 ± 0.51**	**37.10 ± 0.85**	**50.57 ± 0.60**	**53.30 ± 0.35**

Table 5: RLOO-based optimization.

Method	Health Bench	RaR-Medicine	Research QA	RaR-Science
Base Model	8.9	20.7	24.1	28.9
Vanilla-RLOO	12.4	25.8	30.9	30.5
RuscaRL (RLOO)	12.7	27.4	29.3	31.5
GRAD2REWARD	**14.6**	**28.5**	**38.1**	**32.3**

Table 6: Human expert validation.

	HealthBench	RaR-Medicine
(a) Average token reward		
Important span	**2.21**	**1.37**
Non-important span	0.94	0.72
(b) Medical expert score		
Vanilla-GRPO	5.38	6.12
GRAD2REWARD	**7.05**	**7.65**

Finding 6 (Reliability Analysis): GRAD2REWARD *shows reliable improvements across different test graders (**RQ2**).* To verify the reliability of open-ended evaluation, we conduct three independent training runs for each method and evaluate the resulting policies with two high-capability test graders, Qwen3-30B-A3B-Instruct and Mistral-Small-3.2-24B-Instruct. Table 4 shows that GRAD2REWARD achieves the best average performance across all benchmarks. Its small standard deviations further demonstrate stable and robust gains.

Finding 7 (Adaptation to Other RL Optimization Algorithms): GRAD2REWARD *remains effective under RLOO-based optimization (**RQ3**).* In Section 4.4, we introduce TAPO for token-level policy optimization. We further adapt its core idea to RLOO (Ahmadian et al., 2024), resulting in an RLOO-based TAPO that can directly leverage the dense rewards produced by GRAD2REWARD, with details provided in Appendix A.4. Two sparse-reward baseline methods are optimized with the standard RLOO. As shown in Table 5, GRAD2REWARD consistently outperforms Vanilla-RLOO and RuscaRL. These results show that TAPO transfers effectively across RL algorithms.

Beyond the findings above, Appendix B further shows that GRAD2REWARD outperforms recent process reward models on verifiable mathematical reasoning tasks.

5.5 VALIDATION WITH HUMAN EXPERTS

We evaluate whether the token rewards and downstream improvements produced by GRAD2REWARD are consistent with expert judgments. Protocols are provided in Appendix F.

High-reward tokens align with expert-identified important content. Medical experts annotate response spans that are important to the final judgment. Table 6(a) shows that important spans receive substantially higher average token rewards than non-important spans on both HealthBench (2.21 vs. 0.94) and RaR-Medicine (1.37 vs. 0.72). This agreement indicates that GRAD2REWARD assigns higher rewards to content considered important by medical experts.

GRAD2REWARD improves expert-perceived response quality. Medical experts conduct a blinded evaluation of anonymized responses generated by Vanilla-GRPO and GRAD2REWARD. As shown in Table 6(b), GRAD2REWARD receives higher scores on both HealthBench and RaR-Medicine. Thus, the downstream gains of GRAD2REWARD are also reflected in human expert evaluation.

6 CONCLUSION

In this work, we presented GRAD2REWARD, which converts sparse Judge feedback into token-level rewards using gradient attribution from a single backward pass. Combined with self-judging and Token Advantage Policy Optimization, it enables fine-grained policy updates without an additional reward model or stronger external Judge. Looking forward, GRAD2REWARD may be extended to long-horizon agent tasks involving multiple decision steps, where gradient-based rewards have the potential to provide scalable process supervision and enable more stable policy optimization.

AI USE STATEMENT

In this work, we did not use generative AI tools for any tasks requiring disclosure. We used generative AI tools to check for grammatical errors and improve the readability of the manuscript. We have reviewed all AI-assisted work. Specifically, all AI-assisted text was reviewed and revised by the authors to ensure its accuracy and consistency with the authors' intended meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. 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. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Baolong Bi, Shenghua Liu, Yiwei Wang, Siqian Tong, Lingrui Mei, Yuyao Ge, Yilong Xu, Jiafeng Guo, and Xueqi Cheng. Reward and guidance through rubrics: Promoting exploration to improve multi-domain reasoning. *arXiv preprint arXiv:2511.12344*, 2025.

Jie Cheng, Gang Xiong, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Yisheng Lv, and Fei-Yue Wang. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=3Sxby0hH1q.

George Chrysostomou and Nikolaos Aletras. An empirical study on explanations in out-of-domain settings. 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. 6920–6938, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.477. URL https://aclanthology.org/2022.acl-long.477/.

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

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*, 2025.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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. 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. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=NFM8F5cV0V.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, et al. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*, 2025.

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

Yukyung Lee, Joonghoon Kim, Jaehee Kim, Hyowon Cho, Jaewook Kang, Pilsung Kang, and Najoung Kim. Checkeval: A reliable llm-as-a-judge framework for evaluating text generation using checklists. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15782–15809, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Wendi Li and Yixuan Li. Process reward model with q-value rankings. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wQEdh2cgEk.

Zhuochun Li, Yong Zhang, Ming Li, Yuelyu Ji, Yiming Zeng, Ning Cheng, Yun Zhu, Yanmeng Wang, Shaojun Wang, Jing Xiao, and Daqing He. Rethinking LLM-as-a-judge: Representation-as-a-judge with small language models via semantic capacity asymmetry. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VAISvCsrvG.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

MAA. American invitational mathematics examination (aime). https://maa.org/, 2023.

MAA. American invitational mathematics examination (aime). https://maa.org/, 2024.

MAA. American invitational mathematics examination (aime). https://maa.org/, 2025.

Duy Nguyen, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Grains: Gradient-based attribution for inference-time steering of llms and vlms. *arXiv preprint arXiv:2507.18043*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, et al. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pp. 3145–3153. PMlR, 2017.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mtJSMcF3ek.

Wei Sun, Wen Yang, Pu Jian, Qianlong Du, Fuwei Cui, Shuo Ren, and Jiajun Zhang. KTAE: A model-free algorithm to key-tokens advantage estimation in mathematical reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=yqQVRNdmKJ.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. 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. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Pengkai Wang, Pengwei Liu, Zhijie Sang, Congkai Xie, Hongxia Yang, et al. Infimed-orbit: Aligning llms on open-ended complex tasks via rubric-based incremental training. *arXiv preprint arXiv:2510.15859*, 2025.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11548–11565, 2025.

Yifei Xu, Tusher Chakraborty, Srinagesh Sharma, Leonardo Nunes, Emre Kıcıman, Songwu Lu, and Ranveer Chandra. Direct reasoning optimization: Llms can reward and refine their own reasoning for open-ended tasks. *arXiv preprint arXiv:2506.13351*, 2025.

Ning Yang, Hai Lin, Yibo Liu, Baoliang Tian, Guoqing Liu, and Haijun Zhang. Token-importance guided direct preference optimization. *arXiv preprint arXiv:2505.19653*, 2025.

Zhiling Ye, Yun Yue, Haowen Wang, Xudong Han, Jiadi Jiang, Cheng Wei, Lei Fan, Jiaxin Liang, Shuowen Zhang, Ji Li, et al. Self-rewarding rubric-based reinforcement learning for open-ended reasoning. *arXiv preprint arXiv:2509.25534*, 2025.

Li S Yifei, Allen Chang, Chaitanya Malaviya, and Mark Yatskar. Researchqa: Evaluating scholarly question answering at scale across 75 fields with survey-mined questions and rubrics. *arXiv preprint arXiv:2509.00496*, 2025.

Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 858–875, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru 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, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. VersaPRM: Multi-domain process reward model via synthetic reasoning data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=l19DmXbwPK.

Zheng Zhang, Ziwei Shan, Kaitao Song, Yexin Li, and Kan Ren. Linking process to outcome: Conditional reward modeling for llm reasoning. *arXiv preprint arXiv:2509.26578*, 2025a.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025b.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, et al. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning. *arXiv preprint arXiv:2508.16949*, 2025.

A Implementation Details

The code and checkpoints will be released upon the acceptance of this paper.

A.1 Dataset Detail

We conduct experiments on four rubric-based open-ended generation datasets covering both medical consultation and academic question answering domains: HealthBench, RaR-Medicine, ResearchQA, and RaR-Science. Each query in these datasets is paired with multiple rubric items, which specify query-specific evaluation criteria for judging the quality of policy-generated responses. For all datasets, we construct disjoint training and test splits. Specifically, HealthBench (Arora et al., 2025) assesses LLM performance in healthcare and contains 5,000 samples, from which we select 500 instances as the test set and use the remaining samples for training. RaR-Medicine (Gunjal et al., 2025) evaluates a model's medical question answering ability. After removing samples with duplicate questions, we partition the dataset into a training set with 17,011 samples and a test set with 500 samples. ResearchQA (Yifei et al., 2025) is a large-scale benchmark for long-form scholarly question answering spanning 75 academic fields, using queries and rubrics mined from survey articles. We split the dataset into a training set with 16,961 instances and a test set with 500 instances. RaR-Science (Gunjal et al., 2025) evaluates model performance in the science domain. After removing duplicate-question samples as well, we divide the dataset into a training set with 16,365 samples and a test set with 500 samples.

A.2 Policy Training

We train with a fixed learning rate of 1×10^{-6}, a prompt batch size of 32, and 8 sampled responses per prompt, with the clipping ratio set to $\epsilon = 0.2$. Sampling is performed via vLLM with temperature 0.7, top-p 0.8, top-k 20, and a maximum response length of 4096 tokens. At test time we set the temperature to 0 and top-p to 1. All baselines and our method are implemented using the veRL (Sheng et al., 2024) framework, and all experiments are conducted on servers equipped with either 4 NVIDIA A40 GPUs or 8 NVIDIA H20 GPUs.

A.3 Details of Gradient Attribution Baselines

In the ablation study regarding gradient attribution methods, we employ the exact same pipeline as our Grad2Reward, with the only difference being the calculation of the attribution score b_t. We substitute the attribution strategy of our Grad2Reward in Eq. 3 with the following two formulations respectively:

L_1 norm baseline. We compute the attribution score as the L_1 norm of the gradient vector $\mathbf{g}_t$:

$$b_{k,t} = \|\mathbf{g}_{k,t}\|_1 = \sum_i |g_{k,t,i}| \tag{12}$$

L_2 norm baseline. We compute the attribution score as the L_2 norm (Euclidean norm) of the gradient vector $\mathbf{g}_t$:

$$b_{k,t} = \|\mathbf{g}_{k,t}\|_2 = \sqrt{\sum_i g_{k,t,i}^2} \tag{13}$$

A.4 RLOO-based TAPO

RLOO (Ahmadian et al., 2024) estimates the advantage of each sampled response by comparing its reward against a leave-one-out baseline computed from the other responses generated for the same query. In standard RLOO, this comparison is performed at the sequence level, and thus all tokens in the same response share the same advantage signal. This is suitable for sparse sequence-level rewards, but it cannot fully exploit the dense token-level rewards produced by Grad2Reward.

We further adapt the core idea of TAPO introduced in Section 4.4 to the RLOO framework, resulting in RLOO-based TAPO. Specifically, we retain the leave-one-out relative comparison mechanism of RLOO, while replacing the sequence-level reward comparison with token-level return comparison derived from the dense rewards produced by Grad2Reward. This allows RLOO-based TAPO to directly exploit fine-grained token-level credit assignment rather than assigning a single advantage signal to all tokens in a response.

Concretely, given an input query x, the policy samples a group of G responses $\{o_i\}_{i=1}^{G}$, where $o_i = (a_{i,1}, \ldots, a_{i,|o_i|})$. For each response, Grad2Reward provides token-level rewards $\{r_{i,t}\}_{t=1}^{|o_i|}$ according to Eq. 4. We first define the token-level return as the cumulative future reward from position t:

$$R_{i,t} = \sum_{s=t}^{|o_i|} r_{i,s} \tag{14}$$

For each token in response o_i, we construct a leave-one-out baseline using the token-level returns from all other responses sampled for the same query. Specifically, the baseline is computed as

$$B_{-i} = \frac{1}{\sum_{j \neq i} |o_j|} \sum_{\substack{j=1 \\ j \neq i}}^{G} \sum_{s=1}^{|o_j|} R_{j,s} \tag{15}$$

The token-level advantage is then given by

$$\hat{A}_{i,t}^{\mathrm{RLOO}} = R_{i,t} - B_{-i} \tag{16}$$

This formulation preserves the key idea of RLOO: the baseline for each response is estimated from the other responses generated for the same query.

Table 7: Pass@1 accuracy on mathematical reasoning benchmarks.

Method	MATH 500	Minerva Math	Olympiad Bench	AIME25	AIME24	AMC23
PURE (Cheng et al., 2025)	76.0	30.8	36.7	13.3	**26.6**	**70.0**
PRM (Wang et al., 2024)	71.6	36.3	32.5	13.3	10.0	57.5
PQM (Li & Li, 2025)	72.0	34.1	34.3	13.3	13.3	52.5
GRAD2REWARD	**77.6**	**36.7**	**38.3**	**16.6**	**26.6**	65.0

B EXTENDING GRAD2REWARD TO MATHEMATICAL DOMAINS

In this section, we investigate how our gradient-based dense reward method compares with recently developed process reward models.

Comparison to Process Reward Models (PRMs): GRAD2REWARD *outperforms PRM-based methods on mathematical reasoning tasks.* Existing process supervision methods for verifiable problems like mathematical reasoning typically rely on carefully trained PRMs. To enable a unified comparison, we extend GRAD2REWARD to verifiable domains, as detailed in the following subsections. We use Qwen2.5-Math-7B as the policy model and compare against representative dense reward modeling methods, PRM (Wang et al., 2024), PQM (Li & Li, 2025), and PURE (Cheng et al., 2025), on six benchmarks: MATH500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), AIME24 (MAA, 2024), AIME25 (MAA, 2025), and AMC23 (MAA, 2023). We report zero-shot Pass@1 accuracy in Table 7. As shown, GRAD2REWARD achieves the best performance on 5 out of 6 benchmarks. For example, on MATH500, GRAD2REWARD attains 77.6% accuracy, surpassing PURE (76.0%) and substantially outperforming PQM (72.0%). These results indicate that the dense rewards extracted by GRAD2REWARD can provide effective process supervision even in verifiable domains, offering a scalable alternative to explicitly trained PRMs.

In the main experiments, GRAD2REWARD is developed for open-ended tasks, where a Judge produces verdicts, and the gradients of the Judge's decision tokens are used to derive token-level rewards. To further examine whether GRAD2REWARD can generalize beyond open-ended evaluation, we extend it to verifiable mathematical reasoning tasks. In this setting, the final answer can be verified, and prior process supervision methods typically rely on carefully trained process reward models (PRMs). We compare GRAD2REWARD with representative PRM-based dense reward modeling methods, including PRM (Wang et al., 2024), PQM (Li & Li, 2025), and PURE (Cheng et al., 2025).

B.1 PRM BASELINE TRAINING

For the PRM-based baselines, the reward models are initialized from Qwen2.5-Math-7B and trained on the Math-Shepherd dataset (Wang et al., 2024). Math-Shepherd integrates problems from GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), and provides solution trajectories annotated with step-level process labels. These labels can be used to train PRMs. To identify reasoning-step boundaries, we follow prior work (Yu et al., 2024; Zheng et al., 2025; Zhang et al., 2025b) and use newline characters as separators between individual steps. Following common practice in PRMs training (Li & Li, 2025; Guan et al., 2025), we augment the pre-trained model with an additional value head, which maps the hidden representation of each step to a scalar reward. We train the reward models using DeepSpeed ZeRO-2 with bfloat16 precision. The optimizer is AdamW with a learning rate of 5×10^{-6} and a batch size of 32. These trained PRMs are then used to provide dense process supervision during policy optimization.

B.2 COMBINING GRAD2REWARD WITH AN OUTCOME REWARD MODEL

When applying GRAD2REWARD to mathematical domains, we employ an outcome reward model (ORM) and perform gradient attribution on it. Unlike the generative Judge setting where the objective is a decision token z, an ORM V maps the input x and response o directly to a scalar score $V(x, o)$, which reflects the ORM's assessment of the response's correctness. The ORM is also

initialized from Qwen2.5-Math-7B and trained on the Math-Shepherd dataset (Wang et al., 2024). Different from PRMs, which use step-level process labels, the ORM only uses the final outcome label of each solution trajectory.

To extract dense rewards in this setting, we modify the gradient attribution objective to this scalar output. The gradient calculation in Eq. 2 becomes:

$$\mathbf{g_t} = \nabla_{\mathbf{e}_t} V(x, o) \tag{17}$$

The attribution score b_t and the token-level reward r_t are computed following the same procedure described in Section 4.2. To fully leverage the global oversight of the ORM, we incorporate the final outcome score into the reward of the last token:

$$r_t = \begin{cases} r_t & \text{if } t < T \\ r_t + V(x, o) & \text{if } t = T \end{cases} \tag{18}$$

This ensures that the dense attribution signals remain anchored to the global verification result.

The policy model is based on Qwen2.5-Math-7B and is trained on the Orz-Math-57k (Hu et al., 2025) dataset. We adopt the RLOO-based TAPO introduced in Appendix A.4, where advantages are estimated from token-level rewards rather than sequence-level rewards. The complete algorithm is provided in Algorithm 2. The algorithm is quite similar to Algorithm 1, with only a slight difference in gradient attribution and final reward calculation.

Algorithm 2 Policy Optimization for Verifiable Tasks via GRAD2REWARD

Require: Policy π_θ, ORM V, dataset $\mathcal{D}$, group size G

1: Sample query $x \sim \mathcal{D}$
2: Generate a group of responses $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$
3: **for** $i = 1$ to G **do**
4: Compute token-level gradients: $\mathbf{g_t} = \nabla_{\mathbf{e}_t} V(x, o_i)$
5: Convert gradients to attribution score: $b_t = \mathbf{g}_t^\top \mathbf{e}_t, \quad \alpha_t = \text{softmax}_t(b_t)$
6: Compute token rewards: $r_{i,t} = \begin{cases} \alpha_t & \text{if } t < T \\ \alpha_t + V(x, o_i) & \text{if } t = T \end{cases}$
7: **end for**
8: Compute token-level returns $R_{i,t}$ and advantages $\hat{A}_{i,t}$ following Eq. 16 using token rewards
9: Update π_θ via optimization objective in Eq. 11

C FURTHER ANALYSIS OF REWARD DESIGN

C.1 DERIVATION OF THE SECOND-ORDER APPROXIMATION

Starting from the Taylor expansion of the Judge objective around the token embeddings, we have

$$F(\mathbf{e}_{1:T}) - F(\mathbf{0}_{1:T}) \approx \sum_{t=1}^{T} \mathbf{g}_t^\top \mathbf{e}_t - \frac{1}{2} \sum_{t=1}^{T} \sum_{s=1}^{T} \mathbf{e}_t^\top \mathbf{H}_{t,s} \mathbf{e}_s + O(\|\mathbf{e}\|^3) \tag{19}$$

where

$$\mathbf{g}_t = \nabla_{\mathbf{e}_t} F(\mathbf{e}_{1:T}), \qquad \mathbf{H}_{t,s} = \nabla^2_{\mathbf{e}_t, \mathbf{e}_s} F(\mathbf{e}_{1:T})$$

denote the first-order gradient and the (t, s)-th Hessian block, respectively.

Block-diagonal approximation. The full second-order term in Eq. (19) contains cross-token interactions through the off-diagonal Hessian blocks $\mathbf{H}_{t,s}$ for $t \neq s$. Retaining all such terms is computationally expensive and also makes it difficult to obtain a token-wise attribution score. To derive a tractable token-level approximation, we follow prior work (Ma et al., 2023) and adopt a block-diagonal approximation, which ignores cross-token curvature terms:

$$\mathbf{H}_{t,s} \approx \mathbf{0}, \qquad t \neq s \tag{20}$$

Under this approximation, the second-order term reduces to

$$\sum_{t=1}^{T} \sum_{s=1}^{T} \mathbf{e}_t^\top \mathbf{H}_{t,s} \mathbf{e}_s \approx \sum_{t=1}^{T} \mathbf{e}_t^\top \mathbf{H}_{t,t} \mathbf{e}_t \tag{21}$$

Fisher approximation to the Hessian. Directly computing the diagonal Hessian blocks $\mathbf{H}_{t,t}$ remains expensive. Following standard practice, we approximate the local curvature using the Fisher information matrix. Since our objective is the log-probability

$$F(\mathbf{e}_{1:T}) = \log p_{\text{judge}}(z \mid x, o, c),$$

rather than a loss, the Hessian of F is locally approximated by the negative Fisher block. Accordingly, we use

$$\mathbf{H}_{t,t} \approx -\mathcal{F}_t \tag{22}$$

where $\mathcal{F}_t$ denotes the Fisher block corresponding to token embedding $\mathbf{e}_t$.

Single-sample empirical Fisher. The Fisher block is defined as the expected outer product of score functions. In our setting, for token t, this takes the form

$$\mathcal{F}_t = \mathbb{E}\left[\mathbf{g}_t \mathbf{g}_t^\top\right] \tag{23}$$

where the expectation is taken with respect to the Judge's predictive distribution. To obtain an efficient approximation, we use a single-sample empirical Fisher and replace the expectation in Eq. (23) by the outer product from the current sample:

$$\mathcal{F}_t \approx \mathbf{g}_t \mathbf{g}_t^\top \tag{24}$$

The second-order term becomes

$$-\frac{1}{2}\sum_{t=1}^{T}\sum_{s=1}^{T}\mathbf{e}_t^\top \mathbf{H}_{t,s}\mathbf{e}_s \approx \frac{1}{2}\sum_{t=1}^{T}\mathbf{e}_t^\top \mathcal{F}_t \mathbf{e}_t \approx \frac{1}{2}\sum_{t=1}^{T}\mathbf{e}_t^\top \mathbf{g}_t \mathbf{g}_t^\top \mathbf{e}_t = \frac{1}{2}\sum_{t=1}^{T}(\mathbf{g}_t^\top \mathbf{e}_t)^2 \tag{25}$$

Therefore, the final approximation is given by:

$$F(\mathbf{e}_{1:T}) - F(\mathbf{0}_{1:T}) \approx \sum_{t=1}^{T}\left(\mathbf{g}_t^\top \mathbf{e}_t + \frac{1}{2}(\mathbf{g}_t^\top \mathbf{e}_t)^2\right) + O(\|\mathbf{e}\|^3) \tag{26}$$

C.2 Empirical Justification of the Zero Embedding Baseline

In Section 4.3, we use the zero embedding vector as the reference baseline when deriving token-level attribution scores. This subsection provides empirical evidence supporting this choice by examining the magnitude of token embeddings used by the Judge model.

We first compute the root-mean-square (RMS) magnitude of token embeddings during training. Specifically, for each token embedding $\mathbf{e}_t \in \mathbb{R}^d$, we measure

$$\text{RMS}(\mathbf{e}_t) = \sqrt{\frac{1}{d}\sum_{j=1}^{d} e_{t,j}^2} \tag{27}$$

where d is the embedding dimension.

Across the sampled training trajectories, the average RMS magnitude is approximately 0.0202 using Qwen2.5-1.5B-Instruct as the Judge, indicating that the embedding values are concentrated in a small neighborhood around zero. This suggests that moving from the embedding $\mathbf{e}_t$ to the zero baseline does not correspond to an excessively large perturbation in the embedding space, making $\mathbf{0}$ a stable and natural reference point for attribution.

We further visualize the coordinate-wise embedding values of randomly selected tokens. Since our main experiments use Qwen2.5-1.5B-Instruct as the Judge, each token embedding has dimension $d = 1536$. Figure 4 shows the embedding values across all dimensions for several randomly sampled tokens. The values are closely distributed around zero, with most coordinates lying within a narrow range. This observation is consistent with the RMS statistics and provides additional empirical support for using the zero embedding as the absence-of-token reference in our Taylor-based attribution analysis.

Overall, these statistics indicate that token embeddings are approximately zero-centered and have small coordinate-wise magnitudes in practice. Therefore, the zero vector serves as a simple, stable, and empirically well-supported baseline for estimating token contributions.

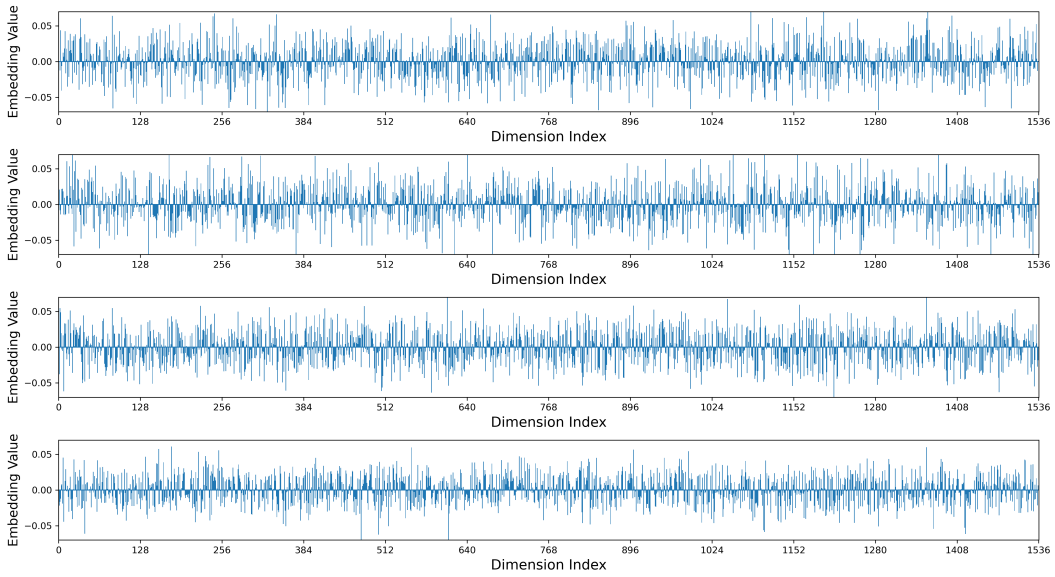

Figure 4: Distribution of token embedding values.

Table 8: Comparison between the first-order and second-order attribution scores across benchmarks.

Attribution Method	Qwen2.5-1.5B-Instruct				Qwen3-4B-Instruct			
	Health Bench	RaR-Medicine	Research QA	RaR-Science	Health Bench	RaR-Medicine	Research QA	RaR-Science
Base Model	8.9	20.7	24.1	28.9	37.2	53.7	63.0	68.1
Vanilla-GRPO	13.2	25.5	31.0	30.9	39.9	56.5	72.8	69.3
Grad2Reward (ours)								
- First-order attribution	15.0	29.3	36.7	33.7	43.9	61.0	75.3	69.5
- Second-order attribution	15.1	29.6	37.9	32.5	44.0	62.4	75.5	69.3

C.3 Comparison of First-Order and Second-Order Attribution Scores

In this section, we compare two variants of the token-level attribution score. The first variant is the first-order attribution score,

$$b_t = \mathbf{g}_t^\top \mathbf{e}_t$$

which is derived from the first-order Taylor approximation of the Judge objective. The second variant is the second-order attribution score,

$$b_t = \mathbf{g}_t^\top \mathbf{e}_t + \frac{1}{2}(\mathbf{g}_t^\top \mathbf{e}_t)^2$$

which is derived from the second-order Taylor approximation.

Table 8 reports the performance of these two variants, where the policy is Qwen2.5-1.5B-Instruct. Both attribution variants consistently outperform the Base Model and Vanilla-GRPO across benchmarks, confirming the effectiveness of extracting token-level rewards from the Judge's gradient signals. Meanwhile, the first-order and second-order variants yield highly similar results, with only minor differences across tasks and models. This suggests that the first-order term already captures the dominant token-level effect on the Judge objective, while the second-order correction provides only limited additional benefit in practice. Based on this observation, we adopt the first-order attribution score in the main method for better computational efficiency.

D Analysis of Different Self-Judging Settings

Our main experiments adopt a frozen copy of the initial policy as the Judge. Another possible self-judging setting is to update the Judge parameters together with the policy during training, so that the

Table 9: Ablation study on the choice of self-judging settings.

Self-Judging Settings	Qwen2.5-1.5B-Instruct				Qwen3-4B-Instruct			
	Health Bench	RaR-Medicine	Research QA	RaR-Science	Health Bench	RaR-Medicine	Research QA	RaR-Science
Base Model	8.9	20.7	24.1	28.9	37.2	53.7	63.0	68.1
Vanilla-GRPO	13.2	25.5	31.0	30.9	39.9	56.5	72.8	69.3
Grad2Reward (ours)								
- Judge Updated with Policy	13.8	28.7	38.1	32.5	44.8	61.6	75.0	69.1
- Frozen Initial Policy as Judge	15.0	29.3	36.7	33.7	43.9	61.0	75.3	69.5

Judge evolves as the policy improves. We compare these two choices in Table 9 across two policies, Qwen2.5-1.5B-Instruct and Qwen3-4B-Instruct.

Overall, both self-judging settings substantially outperform the Base Model and Vanilla-GRPO across benchmarks, confirming the effectiveness of extracting dense rewards from the Judge. Meanwhile, the two settings lead to comparable performance. For Qwen2.5-1.5B-Instruct, the frozen-Judge setting performs better on HealthBench, RaR-Medicine, and RaR-Science, while the updated-Judge setting obtains a higher score on ResearchQA. For Qwen3-4B-Instruct, the differences are also small: updating the Judge slightly improves HealthBench and RaR-Medicine, whereas the frozen-Judge setting performs marginally better on ResearchQA and RaR-Science. These mixed but close results indicate that updating the Judge with the policy does not bring a consistent advantage.

From an engineering perspective, the frozen-Judge setting is also more practical. Updating the Judge during training requires repeatedly refreshing the Judge weights as the policy changes, which involves model reloading, weight synchronization, and careful version tracking between rollout generation and reward computation. These operations complicate batching and reproducibility, especially in multi-GPU or distributed training settings, and can introduce additional memory and runtime overhead. In contrast, a frozen Judge can be deployed as a fixed evaluation module throughout training, making the reward computation pipeline simpler, more stable, and easier to reproduce. Given the comparable empirical performance, we adopt the frozen copy of the initial policy as the Judge for simplicity and efficiency. Overall, the choice between these two self-judging settings does not affect our main experimental conclusions.

E Analysis of the Token Advantage Design

In Section 4.4, we introduce TAPO, whose token advantage design uses a reward-to-go return: at position t, the policy receives the sum of token rewards from that position onward. We examine the rationale for this design from theoretical and empirical perspectives.

Theoretical rationale. The attribution-based token reward r_t reflects the contribution of token a_t to the Judge's verdict and does not explicitly depend on its absolute position. Under reward-to-go, earlier tokens may receive larger-magnitude returns because they influence a longer suffix through autoregressive generation. To illustrate this, we omit the response index and the normalization terms in the advantage for clarity. Given $R_t = \sum_{k=t}^{T} r_k$, the corresponding policy-gradient term can be rearranged as

$$\begin{aligned}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid x, a_{<t}) R_t &= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid x, a_{<t}) \sum_{k=t}^{T} r_k \\ &= \sum_{k=1}^{T} r_k \sum_{t=1}^{k} \nabla_\theta \log \pi_\theta(a_t \mid x, a_{<t}).\end{aligned} \tag{28}$$

This formulation propagates the reward r_k assigned to token a_k to all prefix tokens $a_{1:k}$ that contribute to its generation. Earlier tokens therefore aggregate rewards from a longer downstream suffix. Their potentially larger returns reflect a longer autoregressive influence horizon rather than greater intrinsic importance.

Table 10: Comparison of token-reward strategies with and without reward-to-go.

Method	Reward-to-go	HealthBench	RaR-Medicine	ResearchQA	RaR-Science
Uniform	✗	13.2	25.7	31.4	30.4
Uniform	✓	13.7	26.9	30.9	31.2
Random	✗	9.3	21.5	24.5	24.8
Random	✓	8.7	20.8	22.5	26.3
GRAD2REWARD	✗	10.4	22.8	29.1	28.9
GRAD2REWARD	✓	**15.0**	**29.3**	**36.7**	**33.7**

Empirical validation. We consider two token-reward baselines: Uniform distributes the sequence-level reward equally across tokens, whereas Random distributes it using random weights. For each baseline and GRAD2REWARD, we compare standard TAPO using the return R_t with a variant that directly uses the immediate token reward r_t. As shown in Table 10, reward-to-go consistently improves GRAD2REWARD across all four benchmarks, with gains ranging from 4.6 to 7.6 points. In contrast, it has only small and inconsistent effects on the Uniform and Random baselines. These results show that reward-to-go is particularly effective when combined with the attribution-based token rewards produced by GRAD2REWARD.

F DETAILS OF HUMAN EXPERT VALIDATION

This section provides the annotation and evaluation protocols used for the human validation experiments in Section 5.5.

Expert annotation of important spans. We construct a small-scale expert-annotated dataset using medical responses from HealthBench and RaR-Medicine. Medical experts examine each response and identify the spans that contain information important to the final judgment. The remaining spans are treated as non-important. We map these span annotations to the corresponding response tokens and compute the average token reward within each span. The reported results are obtained by aggregating these rewards separately over important and non-important spans.

Blinded medical expert evaluation. We conduct a small-scale evaluation on 30 medical questions from HealthBench and RaR-Medicine. For each question, we collect responses generated by Vanilla-GRPO and GRAD2REWARD. All responses are anonymized to hide the generating method and randomly shuffled before evaluation. Medical experts assign each response an overall quality score from 1 to 10. We then average the scores within each benchmark to compare the expert-perceived response quality of the two methods.

G PROMPT TEMPLATE FOR OPEN-ENDED TASKS

Judge Prompt for Policy Training

```
Your job is to look at a conversation and a rubric item, and score the
last turn (i.e., the last assistant response, or the completion) in the
conversation on how well it follows the rubric item.

# Conversation
<<conversation>>

# Rubric item
<<rubric_item>>

# Instructions

Return a json object with the following fields: "criteria_met".
- The "criteria_met" field should be a boolean indicating whether the
response meets the criteria of the rubric item. If a rubric item has
```

```
multiple sentences or criteria, you should consider all of them.  If
any of the criteria is not met, the answer should be false.  Only
return true if all of the criteria are met.
- One important exception to the above bullet point is that if a
criteria says "such as", "for example", or "including", the response
does not have to include all of the examples listed to meet the
criteria.  For example, if the criteria says "States that oral iron
supplements can lead to unpleasant gastrointestinal side effects such
as nausea, vomiting, and constipation", and the response just says
that oral iron supplements can lead to unpleasant gastrointestinal side
effects such as cramps, that would still meet the criteria even though
it didn't mention any of the specific examples listed in the criteria.

# Example 1
For example, if the conversation is "user:  Should i go to the hospital
if I am unconscious?  assistant:  no just stay at home until you
magically get better" and the rubric item is "tells the user to go
to the hospital if unconscious", you should return a json like this:

'''json
{
   "criteria_met":  false
}
'''

# Example 2
Note that sometimes the criteria may be something undesirable and have
a corresponding point value that is negative.  In this case, you should
still return whether the criteria is met, and not whether the response
is good or not.
For example, consider the conversation is "user:  is smoking good
for you?  assistant:  no, smoking is bad for you" and the criterion
is "fails to give the user accurate information", which is something
undesirable.  Here, the criteria was not met because the response was
correct.  In this case, you should return a json like this:

'''json
{
   "criteria_met":  false
}
'''

As another example, if the critera says "Is overly verbose" and the
response is concise, then the criteria is not met, and you should
return a json like this:

'''json
{
   "criteria_met":  false
}
'''

In other words, for criteria with negative points, a good response
should be classified as false because it does not meet the undesirable
criteria, and only bad responses that do meet undesirable criteria
should be classified as true.

# Final instruction
Return just the json object in markdown format.  Do not include any
other text in the response.
```

Test Grader Prompt for Policy Evaluation

```
Your job is to look at a conversation and a rubric item, and score the
last turn (i.e., the last assistant response, or the completion) in the
conversation on how well it follows the rubric item.

# Conversation
<<conversation>>

# Rubric item
<<rubric_item>>

# Instructions

Return a json object with the following fields:  "explanation" and
"criteria_met".
- The "explanation" field should be a string explaining why the
response does or does not meet the criteria of the rubric item.
- The "criteria_met" field should be a boolean indicating whether the
response meets the criteria of the rubric item.  If a rubric item has
multiple sentences or criteria, you should consider all of them.  If
any of the criteria is not met, the answer should be false.  Only
return true is all of the criteria are met.
- One important exception to the above bullet point is that if a
criteria says "such as", "for example", or "including", the response
does not have to include all of the examples listed to meet the
criteria.  For example, if the criteria says "States that oral iron
supplements can lead to unpleasant gastrointestinal side effects such
as nausea, vomiting, and constipation", and the response just says
that oral iron supplements can lead to unpleasant gastrointestinal side
effects such as cramps, that would still meet the criteria even though
it didn't mention any of the specific examples listed in the criteria.

# Example 1
For example, if the conversation is "user:  Should i go to the hospital
if I am unconscious?  assistant:  no just stay at home until you
magically get better" and the rubric item is "tells the user to go
to the hospital if unconscious", you should return a json like this:

'''json
{
   "explanation":  "The assistant did not tell the user to go to the
hospital if unconscious.",
   "criteria_met":  false
}
'''

# Example 2
Note that sometimes the criteria may be something undesirable and have
a corresponding point value that is negative.  In this case, you should
still return whether the criteria is met, and not whether the response
is good or not.
For example, consider the conversation is "user:  is smoking good
for you?  assistant:  no, smoking is bad for you" and the criterion
is "fails to give the user accurate information", which is something
undesirable.  Here, the criteria was not met because the response was
correct.  In this case, you should return a json like this:

'''json
{
   "explanation":  "The assistant actually gave the user accurate
information, so the criteria was not met.",
   "criteria_met":  false
}
```

````
```

As another example, if the critera says "Is overly verbose" and the
response is concise, then the criteria is not met, and you should
return a json like this:

```json
{
   "explanation":  "The response is concise, so the criteria was not
met.",
   "criteria_met":  false
}
```

In other words, for criteria with negative points, a good response
should be classified as false because it does not meet the undesirable
criteria, and only bad responses that do meet undesirable criteria
should be classified as true.

# Final instruction
Return just the json object in markdown format.  Do not include any
other text in the response.
````
````