MATRIXREWARD: REWARD FROM RUBRIC MATRIX FOR OPEN-ENDED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-ended query generation lacks standard answers, thus necessitating an effective reward mechanism. Pointwise scoring rubrics provide limited information about the relative quality of sample answers under the same prompt; merging multiple rubric judgments into a single score may also mask the differences between these answers. We propose **MatrixReward**, which constructs rewards from a rollout-by-rubric win-rate matrix obtained by comparing every pair of sampled responses under each rubric. The spread of each matrix column captures how strongly that rubric distinguishes the current rollouts, while correlations between columns reveal rubric repetition; together, these statistics yield data-dependent rubric weights. We combine these weights with the prior weights of rubrics. After column normalization and weighting, the observed per-rubric maxima and minima define positive and negative ideal profiles. Each rollout's distances to these two ideals determine its relative-closeness quality reward. Evaluated using Qwen3-8B on four open-ended query-answering benchmarks, MatrixReward achieves an average score of 63.02, outperforming the strongest baseline by approximately 2.0%. These results support the idea that matrices derived from relative comparisons can be used to construct rewards more reasonably for open-ended generative reinforcement learning.[1]

1 INTRODUCTION

Reinforcement learning has become an important approach to improving language models beyond their initial capabilities, but its effectiveness depends on the quality of the reward signal (Ouyang et al., 2022; Dubois et al., 2023; Shao et al., 2024). In verifiable tasks, such as mathematics and programming, a correct answer or successful test provides a concrete target. In open-ended generation, feedback must instead account for instruction following, factual grounding, relevance, and presentation (Zheng et al., 2023; Gunjal et al., 2026). A response may be fluent yet uninformative, or detailed yet poorly grounded. Effective rewards must distinguish these qualities among responses to the same prompt.

Rubric-based rewards offer a promising foundation by expressing these requirements as explicit evaluation rubrics (Gunjal et al., 2026; Zhang et al., 2026a; Liu et al., 2026c). Rather than leaving quality entirely to a holistic judgment, rubrics identify the aspects that an evaluator should consider. However, clear rubrics do not automatically produce informative training signals: an evaluator must still determine how well each response satisfies them (Zheng et al., 2023; Tripathi et al., 2025), and those assessments must ultimately guide a policy update. The distinction matters for group-relative policy optimization (GRPO) (Shao et al., 2024): the training signal depends on differences among sampled responses, not merely on whether each response appears satisfactory in isolation.

Many rubric-based reward systems assign pointwise scores to responses independently. Although rubrics specify what to judge, a numerical score lacks a stable anchor when a query has no reference answer; the same value may reflect different quality standards across queries or evaluators (Zheng et al., 2023; Dubois et al., 2024). Rubric weights is also commonly set when the rubrics are created. These weights represent the factors that should be considered in the response, but they

[1] Code and materials required to reproduce our results are available in the anonymous repository at `https://anonymous.4open.science/r/under_review_2027_E32R/`.

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

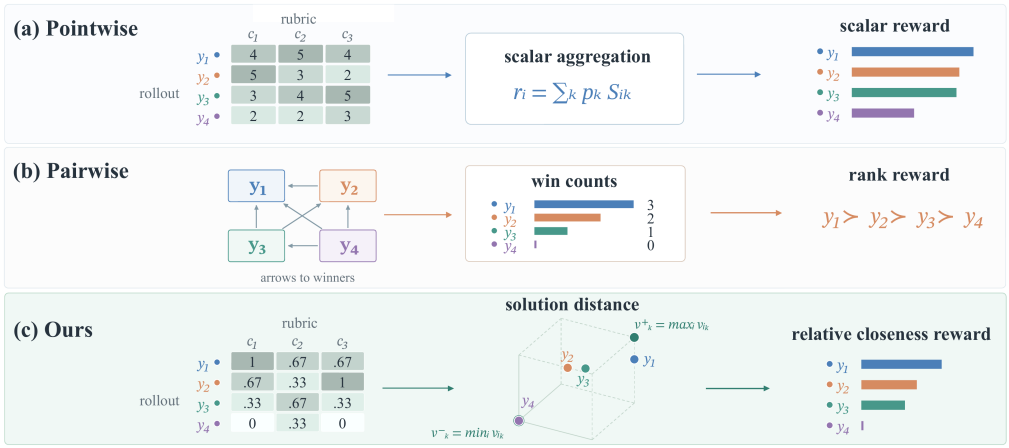

Figure 1: Comparison of reward construction for the same rollout group.

cannot guarantee the differentiation between rollouts sampled during training; and if the rubric is highly correlated, it may lead to duplicate scoring. More importantly, a linearly weighted average of rubrics may cause scores across different dimensions to cancel each other out, assigning the same reward to responses of different quality under the same rubric.

To address these problems, we propose MatrixReward. As illustrated in Figure 1, rubric-wise pairwise judgments retain a separate win-rate profile for each response. The matrix reveals which rubrics distinguish the current rollouts and which provide overlapping information. We combine this evidence with query-specific weights, then compare the resulting profiles with positive and negative ideals to construct rewards.

Our contributions are as follows:

- **Rubric-wise comparison matrix.** We introduce MatrixReward, a reward framework for open-ended group-based RL. By comparing every pair of rollouts under each query-specific rubric, it constructs a rollout-by-rubric win-rate matrix that retains each response's relative quality.
- **Distance-based reward construction.** We derive group-specific rubric weights from column dispersion and cross-rubric correlation, then combine them with query-specific importance priors. After normalizing and weighting the matrix columns, we use their observed maxima and minima as positive and negative ideal profiles. For each rollout, the Euclidean distance to the negative ideal divided by the sum of its distances to both ideals gives a scalar quality reward, favoring profiles near the positive ideal and far from the negative ideal.
- **Four-benchmark evaluation.** Training Qwen3-8B with MatrixReward yields a 63.02 average across four benchmarks, exceeding the strongest pairwise, pointwise, and weight-aware baselines by 1.23, 6.93, and 5.60 points, respectively. Ablations examine weight extraction and fusion, ideal-profile selection, distance metrics, and judging cost.

2 Related Work

Rubric-based reward evaluation. Rubrics make the standards for judging open-ended responses explicit, but do not by themselves establish a common numeric scale across queries or evaluators. Rubrics as Rewards (RaR) uses instance-specific rubrics in on-policy RL, with both weighted rubric-level assessments and rubric-conditioned holistic judgments (Gunjal et al., 2026). Chasing the Tail improves discrimination among high-quality responses by constructing rubrics from diverse examples (Zhang et al., 2026a); OpenRubrics derives rubrics from preference data (Liu et al., 2026c). These works improve what is judged, while the choice between pointwise rubric scores and pairwise comparisons remains consequential for reward construction.

Pairwise and rubric-wise comparisons. Pairwise judgments replace an isolated score with a choice between responses. Tournament-GRPO converts rubric-guided comparisons among same-query rollouts into tournament rewards (Yang et al., 2026); ArenaRL uses process-aware, multi-level rubrics within tournament-based relative ranking (Zhang et al., 2026b). OpenRS uses adaptive rubrics to guide pairwise comparisons and aggregates the resulting preferences for RL supervision (Jia et al., 2026). MAPLE combines aspect-wise pairwise judgments while estimating evaluator reliability and aspect importance for assessment rather than online reward construction (Jinji et al., 2026).

Adaptive weighting of reward dimensions. Human-assigned rubric weight need not coincide with a rubric's current usefulness for learning. POW3R makes this distinction explicit and reallocates rubric-level reward weights using contrast among policy rollouts (Tyagi et al., 2026). SAW uses batch-level reward variation to adapt dimension weights (He et al., 2026), while SA-MRPO downweights saturated objectives at the advantage level (Wang et al., 2026b). AMRP shifts weight toward dimensions with greater shortfall, volatility, or stalled progress (Yuan et al., 2026). These methods establish that weights can respond to training state; they differ in the statistics used and in whether they account for redundancy between rubric dimensions within the current response group.

Aggregation of multi-dimensional feedback. After assigning accurate weights to the rubric, it's still necessary to consider how to construct a suitable reward for reinforcement learning training from the evaluations of multiple rubrics. GDPO normalizes reward dimensions separately before aggregation (Liu et al., 2026b), and GD^{2}PO additionally handles conflicting cross-dimension advantages (Liu et al., 2026a). OpenRS aggregates rubric-level preferences outside the judge (Jia et al., 2026), whereas AMRP identifies how fixed scalar projections can conflate distinct reward profiles (Yuan et al., 2026).

3 METHOD

Figure 2 illustrates the workflow of MatrixReward. For each query, we compare every pair of rollouts under each rubric and organize their rubric-wise win rates into a matrix. We then extract data-driven rubric weights from this matrix and combine them with the weights specified when the rubrics were created. Finally, we compare the weighted response profiles with the observed positive and negative ideal profiles in the multidimensional rubric space, using their relative distances to construct a reward for each rollout.

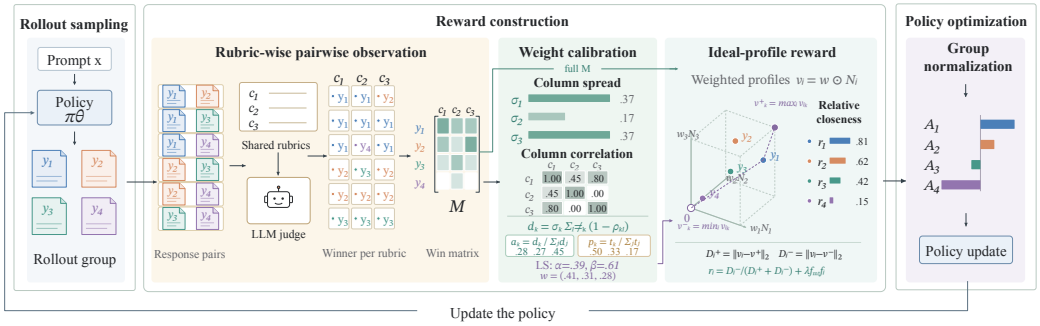

Figure 2: Overview of the MatrixReward framework.

3.1 CONSTRUCTING THE RUBRIC-WISE COMPARISON MATRIX

For a query x, the policy π_θ samples G responses $Y = \{y_i\}_{i=1}^{G}$. The query has $K = K(x)$ rubrics $\{c_k, t_k\}_{k=1}^{K}$, each with a positive stored importance weight t_k. The same rubrics evaluate every response in the group.

For responses i and j, a judge compares them under rubric c_k. In the explicit protocol, each request asks for a verdict on one rubric. The judge may prefer either response or declare a tie. For response y_i, we record

$$z_{ijk} = \begin{cases} 1, & \text{if } y_i \text{ wins against } y_j \text{ under } c_k, \\ \frac{1}{2}, & \text{if the judge declares a tie,} \\ 0, & \text{if } y_j \text{ wins against } y_i \text{ under } c_k, \end{cases} \qquad z_{jik} = 1 - z_{ijk}. \tag{1}$$

We compare every pair of the G responses once under each rubric. For each response and rubric, we sum its outcomes against the other $G-1$ responses, counting a tie as half a win, and divide by $G-1$ to form the rollout-by-rubric win-rate matrix:

$$M_{ik} = \frac{1}{G-1} \sum_{j \neq i} z_{ijk} \in [0,1]. \tag{2}$$

Row i gives the win rates of y_i under all K rubrics; column k gives the win rates of all G responses under rubric c_k. The $\binom{G}{2}$ response pairs require $K\binom{G}{2}$ explicit judge requests.

An implicit alternative asks for all K rubric verdicts in one request per pair and fills the same matrix using $\binom{G}{2}$ requests. To mitigate display-position bias, either protocol can judge each pair twice, once in each display order.

3.2 Calibrating matrix-derived and prior weights

In the win-rate matrix, greater variation within a column means that the rollouts receive more different outcomes under that rubric. Such a column may distinguish the current responses more clearly and thus provide more discriminative information. For weight extraction, we first min–max normalize each nonconstant column, setting $U_{ik} = (M_{ik} - m_k^-)/(m_k^+ - m_k^-)$, where $m_k^- = \min_i M_{ik}$ and $m_k^+ = \max_i M_{ik}$; a constant column is set to zero. We measure variation by the population standard deviation σ_k of $U_{\cdot k}$. A larger σ_k therefore motivates a higher objective weight. Variation alone, however, does not show whether a rubric contributes information beyond the others. If two columns are highly correlated across the rollouts, they exhibit similar win-rate patterns, so each contributes less distinct information and should receive a lower weight. Let $\rho_{k\ell}$ be the Pearson correlation between nonconstant columns $U_{\cdot k}$ and $U_{\cdot \ell}$. We combine column variation with this correlation-based nonredundancy as (Diakoulaki et al., 1995)

$$h_k = \sum_{\ell \neq k} (1 - \rho_{k\ell}), \qquad d_k = \sigma_k h_k, \qquad a_k = \frac{d_k}{\sum_j d_j}. \tag{3}$$

Correlations involving a constant column are set to zero; because that column has $\sigma_k = 0$, it receives $d_k = 0$. If every $d_k = 0$, we use $a_k = 1/K$.

The matrix-derived weights $\mathbf{a}$ reflect how distinctly the current rollouts perform under each rubric, but not how important that rubric is for the query. We therefore combine them with the importance weights assigned when the query-specific rubrics are defined, normalized as $p_k = t_k / \sum_\ell t_\ell$. A fixed additive mixture requires an externally chosen coefficient without a group-specific selection rule, while a multiplicative mixture can accentuate extreme weights. To determine the balance for each response group, we use a game-theoretic least-squares combination (Huang et al., 2023). Let $\widetilde{\mathbf{w}} = u\mathbf{a} + v\mathbf{p}$. Minimizing each source's squared deviation from the mixture with respect to its own coefficient gives the stationary conditions $\mathbf{a}^\top(\widetilde{\mathbf{w}} - \mathbf{a}) = 0$ and $\mathbf{p}^\top(\widetilde{\mathbf{w}} - \mathbf{p}) = 0$. Thus

$$\underbrace{\begin{bmatrix} \mathbf{a}^\top\mathbf{a} & \mathbf{a}^\top\mathbf{p} \\ \mathbf{p}^\top\mathbf{a} & \mathbf{p}^\top\mathbf{p} \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \mathbf{a}^\top\mathbf{a} \\ \mathbf{p}^\top\mathbf{p} \end{bmatrix}. \tag{4}$$

For a nonsingular H with finite solution, the implementation takes the absolute value and normalizes the solved coefficients:

$$(\alpha, \beta) = \frac{(|u|, |v|)}{|u| + |v|}, \qquad w_k^{\text{game}} = \alpha a_k + \beta p_k. \tag{5}$$

When $|\det H| \leq 10^{-12}$ or the solution is nonfinite, we instead use $w_k = \sqrt{a_k p_k} / \sum_\ell \sqrt{a_\ell p_\ell}$ and record the coefficients as $(0.5, 0.5)$. Weight-extraction methods and ablation studies are examined in Sections 4.3 and 4.5, respectively.

Algorithm 1 MatrixReward: matrix construction and reward extraction

Require: query x; G rollouts $Y = \{y_i\}_{i=1}^{G}$; K rubrics $\{c_k, t_k\}_{k=1}^{K}$; format indicators $\mathbf{f}$
Ensure: rewards $\mathbf{r}$ and group advantages $\mathbf{A}$

Rubric-wise pairwise observation
Initialize $M \in \mathbb{R}^{G\times K}$ to zero
for $1 \le i < j \le G$ **do**
  for $k = 1, \dots, K$ **do**
    Judge (y_i, y_j) under c_k; obtain win, loss, or tie
    Set $z_{ijk} \leftarrow 1, 0$, or $1/2$, respectively
    $M_{ik} \leftarrow M_{ik} + z_{ijk}/(G-1)$; $M_{jk} \leftarrow M_{jk} + (1 - z_{ijk})/(G-1)$
  end for
end for
Matrix-derived weight and query-specific prior
for $k = 1, \dots, K$ **do**
  Min–max normalize $M_{\cdot k}$; compute its standard deviation σ_k and cross-rubric correlations $\rho_{k\ell}$
  $d_k \leftarrow \sigma_k \sum_{\ell \neq k} (1 - \rho_{k\ell})$
end for
Set $\mathbf{a} \leftarrow \mathbf{d}/\sum_k d_k$
Set $p_k \leftarrow t_k/\sum_\ell t_\ell$ and form H and its diagonal right-hand side by Eq. 4
Solve Eq. 4 for (u, v); set $(\alpha, \beta) \leftarrow (|u|, |v|)/(|u| + |v|)$
Set $\mathbf{w} \leftarrow \alpha\mathbf{a} + \beta\mathbf{p}$
Ideal-solution reward and GRPO advantage
for $k = 1, \dots, K$ **do**
  Set $N_{ik} \leftarrow M_{ik}/\sqrt{\sum_j M_{jk}^2}$ for all i (or zero for a zero column)
  Set $v_{ik} \leftarrow w_k N_{ik}$, $v_k^+ \leftarrow \max_i v_{ik}$, and $v_k^- \leftarrow \min_i v_{ik}$
end for
for $i = 1, \dots, G$ **do**
  $D_i^{\pm} \leftarrow \|\mathbf{v}_i - \mathbf{v}^{\pm}\|_2$
  $r_i^{\text{qual}} \leftarrow D_i^- / (D_i^+ + D_i^-)$
  $r_i \leftarrow r_i^{\text{qual}} + \lambda_{\text{fmt}} f_i$
end for
$A_i \leftarrow (r_i - \bar{r})/(s_r + \epsilon)$ for all i; **return** $(\mathbf{r}, \mathbf{A})$

3.3 Constructing a reward from multidimensional profiles

For reward construction, normalize each rubric column by its Euclidean norm $L_k = \sqrt{\sum_i M_{ik}^2}$:

$$N_{ik} = \begin{cases} M_{ik}/L_k, & L_k > 0, \\ 0, & L_k = 0. \end{cases} \tag{6}$$

Form the weighted profile $v_{ik} = w_k N_{ik}$. The positive and negative ideal profiles are the observed extrema $v_k^+ = \max_i v_{ik}$ and $v_k^- = \min_i v_{ik}$. Following ideal-solution ranking (Hwang & Yoon, 2012), define

$$\begin{aligned} D_i^+ &= \|\mathbf{v}_i - \mathbf{v}^+\|_2, \\ D_i^- &= \|\mathbf{v}_i - \mathbf{v}^-\|_2, \qquad r_i^{\text{qual}} = \frac{D_i^-}{D_i^+ + D_i^-}. \end{aligned} \tag{7}$$

A response closer to the positive ideal and farther from the negative ideal receives larger $r_i^{\text{qual}} \in [0, 1]$. If both distances vanish, we assign $r_i^{\text{qual}} = 1/2$.

A fixed format-validity reward is added to the quality reward, where f_i indicates whether response y_i follows the required final-answer format:

$$r_i = r_i^{\text{qual}} + \lambda_{\text{fmt}} f_i. \tag{8}$$

GRPO normalizes these rewards within the G responses to the same query (Shao et al., 2024):

$$\bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j, \qquad s_r = \sqrt{\frac{1}{G-1}\sum_{j=1}^{G}(r_j - \bar{r})^2}, \qquad A_i = \frac{r_i - \bar{r}}{s_r + \epsilon}, \tag{9}$$

where $\epsilon = 10^{-6}$. The response-level advantage A_i is assigned to the tokens of y_i and used in the clipped GRPO policy objective to update π_θ. A worked calculation is given in Appendix E.

4 Experiments

4.1 Experimental setup

Training and data. We build the RL pipeline on verl[2] and optimize Qwen3-8B (Yang et al., 2025) with GRPO. The reward judge is Qwen3.6-35B-A3B (Qwen Team, 2026). We sample 2,560 queries from UltraFeedback (Cui et al., 2023). Using the prompt in Appendix A.1, DeepSeek-V4-Flash (Xu et al., 2026) generates three to five query-specific rubrics from each query and its four responses provided in the dataset. The weights attached to these rubrics supply $\mathbf{p}$ in Eq. 5 and are also used for weighted aggregation of the pointwise reward. Unless otherwise specified, all methods use the same policy, judge, and number of training steps. We sample $G = 8$ rollouts per query and report results after three training epochs (480 steps). Detailed hyperparameters and training settings are provided in Appendix B.

Benchmarks. We evaluate on AlpacaEval 2 (Dubois et al., 2024), Arena-Hard v2 (Li et al., 2024), WildBench (Lin et al., 2025), and WritingBench (Wu et al., 2026), using Qwen3.6-35B-A3B as the judge. All scores are on a 0–100 scale. Avg refers to the average of four benchmarks.

4.2 Comparison with baselines

We compare MatrixReward with three groups of baselines: pointwise, pairwise, and weight-calibration methods. The pointwise baselines—GRPO, Dr. GRPO, GOPO, GDPO, GD^{2}PO, and DAPO—score each rollout independently using query-specific rubrics (Shao et al., 2024; Liu et al., 2025; Choi et al., 2026; Liu et al., 2026b;a; Yu et al., 2026). The pairwise baselines RRC-SCR, ArenaRL, Tournament-GRPO, and Pairwise Rank construct rewards from comparisons between rollouts (Wang et al., 2026a; Zhang et al., 2026b; Yang et al., 2026). The weight-calibration baselines SAW, POW3R, and AMRP adjust how rubric dimensions contribute to the reward (He et al., 2026; Tyagi et al., 2026; Yuan et al., 2026). Table 1 reports the benchmark results.

MatrixReward averages 63.02 across the four benchmarks, exceeding RRC-SCR by 1.23 points, DAPO by 6.93 points, and SAW by 5.60 points. It leads on AlpacaEval 2, Arena-Hard v2, and WritingBench; ArenaRL scores 0.04 points higher on WildBench. The implicit variant achieves the second-highest average of 62.23. To assess the contribution of rubric-wise pairwise observations, the pointwise + MatrixReward variant constructs its matrix from independent absolute scores for each rollout and rubric. Its 57.86 average is 5.16 points below the pairwise MatrixReward variant, but it is still 1.77 points better than the best pointwise baseline. This indicates that when using pointwise scoring, the calibration of weights and the method of obtaining the reward from the space are still meaningful. The corresponding Qwen2.5-7B-Instruct results are reported in Appendix C.

4.3 Extracting rubric weights from the matrix

Table 2 compares MatrixReward with the query-specific prior and two alternative matrix-weight extractors. Entropy and SVD retain the game combination and reward mapping of MatrixReward, changing only the matrix-weight extractor. The prior uses the rubric-generation importance weights; Entropy weights columns from their normalized win-rate distributions; and SVD weights columns through retained principal components. MatrixReward uses column variation and cross-rubric correlations. The four-benchmark averages are 63.02, 60.24, 61.46, and 62.16, respectively. Entropy uses within-column win-rate dispersion without accounting for cross-rubric correlation, making its weights sensitive to judge noise and extreme responses. SVD retains principal components that

[2] https://github.com/verl-project/verl

Table 1: Comparison with multiple types of baselines.

Method	AE2 ↑	AH2 ↑	Wild ↑	Writing ↑	Avg ↑
MatrixReward	**85.35**	**41.23**	61.83	**63.66**	**63.02**
MatrixReward (implicit)	83.45	40.90	61.09	63.47	62.23
Pointwise + MatrixReward	72.33	36.91	60.40	61.81	57.86
Pairwise baselines					
RRC-SCR	83.60	39.98	60.59	63.00	61.79
ArenaRL	76.04	34.82	**61.87**	61.73	58.62
Tournament-GRPO	76.56	33.66	59.63	61.43	57.82
Pairwise Rank	83.08	40.93	60.36	61.58	61.49
Pointwise baselines					
GOPO	62.57	31.53	59.50	60.53	53.53
GRPO	63.85	33.08	60.16	61.98	54.77
Dr. GRPO	66.70	28.71	58.71	57.37	52.87
GD^{2}PO	64.09	29.57	58.02	59.29	52.74
GDPO	67.63	32.97	60.31	61.57	55.62
DAPO	68.22	33.45	61.07	61.63	56.09
Weight-calibration baselines					
SAW	70.41	35.16	61.59	62.53	57.42
POW3R	64.79	33.40	57.75	55.97	52.98
AMRP	56.22	31.68	59.82	61.12	52.21

capture shared variation among correlated rubrics; its 62.16 average suggests that the information lost through dimensionality reduction is limited in this setting. Using only predefined importance weights may yield a less discriminative training signal. Protocols for weight extraction using entropy and SVD are specified in Appendix F.

Table 2: Rubric-wise matrix weight variants.

Weight rule	AE2 ↑	AH2 ↑	Wild ↑	Writing ↑	Avg ↑
MatrixReward	**85.35**	**41.23**	**61.83**	**63.66**	**63.02**
Prior weights	81.25	38.44	59.55	61.73	60.24
Entropy	81.61	40.75	60.53	62.95	61.46
SVD	82.12	41.22	**61.83**	63.45	62.16

4.4 Reward aggregation and distance metric

Table 3 compares three alternatives to the ideal-weight reward in MatrixReward. Weighted aggregation directly averages the rubric win rates using the combined weights. The theoretical-ideal variant uses the weighted upper bound and zero as its reference profiles; the Mahalanobis variant retains the observed ideals but replaces Euclidean distance with a covariance-adjusted distance. MatrixReward averages 63.02 across the four benchmarks, compared with 59.83, 60.81, and 61.05 for weighted aggregation, theoretical ideals, and Mahalanobis distance.

Table 3: MatrixReward reward-aggregation and distance comparisons.

Variant	AE2 ↑	AH2 ↑	Wild ↑	Writing ↑	Avg ↑
MatrixReward	**85.35**	**41.23**	**61.83**	**63.66**	**63.02**
Weighted aggregation	78.02	38.54	60.32	62.45	59.83
Theoretical ideals	81.79	40.02	59.74	61.67	60.81
Mahalanobis distance	80.38	40.41	60.66	62.73	61.05

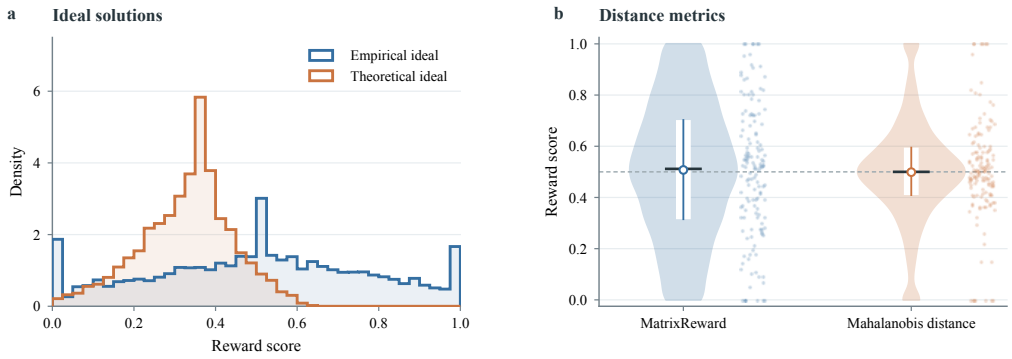

Figure 3: Reward scores under alternative ideal profiles and distance metrics. (a) Empirical versus theoretical ideals. (b) Euclidean distance in MatrixReward versus Mahalanobis distance with the same empirical ideals.

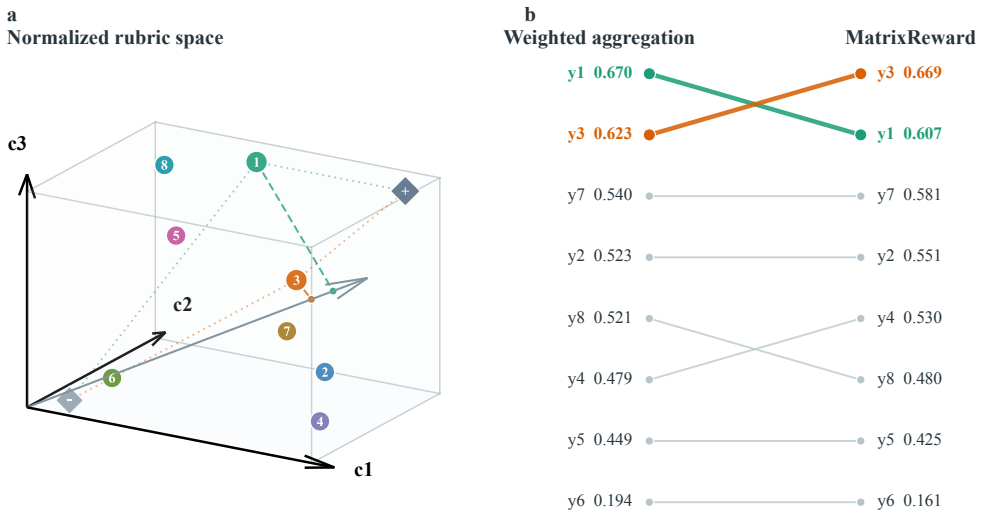

Figure 4: A response group with eight rollouts and three rubrics. (a) Column-normalized profiles and observed ideals. (b) Weighted aggregation and MatrixReward quality rewards. The rubric weights are $(w_{c_1}, w_{c_2}, w_{c_3}) = (0.452, 0.245, 0.304)$.

Figure 3 compares rewards across rollout groups. Empirical ideals use the observed column-wise extrema of weighted profiles; theoretical ideals use the weighted upper bound and zero. Across 800 groups, theoretical rewards cluster near 0.33, whereas empirical rewards span a wider range and average 0.51. With ties permitted, theoretical endpoints may be unattainable: if all eight responses tie under a rubric, each earns $7 \times \frac{1}{2} = 3.5$ rather than seven wins. Holding empirical ideals fixed, Mahalanobis distance narrows the reward distribution relative to Euclidean distance in MatrixReward (standard deviation 0.20 versus 0.26 across 800 groups). Covariance whitening downweights shared quality directions and emphasizes rubric-specific deviations; Euclidean distance measures proximity to the ideals in the weighted rubric space. Table 3 shows the strongest benchmark results with empirical ideals and Euclidean distance, which is MatrixReward.

Figure 4 shows one three-rubric response group. Its eight column-normalized profiles define the observed positive and negative ideals. With identical rubric weights, weighted aggregation ranks y_1 above y_3 (0.670 versus 0.623), whereas MatrixReward reverses them (0.669 versus 0.607) by

comparing distances to both ideals. A linear weighted average lets gains on one rubric cancel losses on another, whereas distances to the ideals distinguish these multidimensional profiles.

4.5 SENSITIVITY TO WEIGHT FUSION

When combining the weights extracted from the matrix with the importance weights, there are also linear addition, multiplication, and other methods. Table 4 compares it with fixed additive mixtures $w_k = (1-\alpha)p_k + \alpha a_k$ and multiplicative fusion $w_k \propto a_k p_k$. Increasing α from 0.1 to 0.5 raises the four-benchmark average from 61.47 to 62.04; multiplicative fusion averages 59.98, while the group-dependent combination reaches 63.02. The fixed coefficient also changes which benchmark benefits most: $\alpha = 0.1$ scores highest on AH2, whereas $\alpha = 0.5$ yields the strongest additive average.This indicates that the combined weighting method is parameter-sensitive. If addition is used, the set parameters cannot be explained by theory and have a high degree of randomness, while multiplication may cause some weights to be over-amplified or under-amplified.

Table 4: MatrixReward weight-fusion ablation.

Fusion rule	AE2 ↑	AH2 ↑	Wild ↑	Writing ↑	Avg ↑
MatrixReward	**85.35**	<u>41.23</u>	**61.83**	**63.66**	**63.02**
Additive ($\alpha = 0.1$)	81.71	**41.40**	60.31	62.46	61.47
Additive ($\alpha = 0.3$)	82.42	40.40	60.73	<u>62.85</u>	61.60
Additive ($\alpha = 0.5$)	<u>83.95</u>	40.51	<u>61.23</u>	62.45	<u>62.04</u>
Multiplicative	80.51	36.52	60.73	62.15	59.98

4.6 JUDGING COST AND OUTCOME

Table 5 compares the explicit and implicit MatrixReward variants with pairwise RRC-SCR, weight-aware SAW, and pointwise DAPO. Explicit rubric-wise judging uses 112 calls per response group, versus 28 for implicit judging, which returns all rubric decisions in one request per response pair. The explicit variant achieves a 63.02 four-benchmark average at 2.00 billion judge tokens. Implicit judging averages 62.23 at 0.52 billion tokens: 0.79 points lower with 74% fewer tokens. It also exceeds RRC-SCR by 0.44 points while using 0.06 billion fewer tokens. SAW and DAPO each use 0.07 billion tokens and obtain averages of 57.42 and 56.09, respectively. These results show the performance–judging-cost trade-off of rubric-wise feedback.

Table 5: Benchmark average and training judge cost.

Method	Avg ↑	Calls/group	Total calls (M)	Total tokens (B)	Tokens/call
MatrixReward	63.02	112	0.86	2.00	2,321
Implicit	62.23	28	0.22	0.52	2,408
RRC-SCR	61.79	28	0.21	0.58	2,676
SAW	57.42	8	0.06	0.07	1,070
DAPO	56.09	8	0.06	0.07	1,070

5 CONCLUSION

We introduced MatrixReward, a framework for constructing relative rewards from rubric-wise comparisons in open-ended group-based RL. It organizes pairwise judgments into a rollout-by-rubric win-rate matrix, calibrates query-specific weights using the matrix's discrimination and cross-rubric relationships, and converts the weighted profiles into rewards through distances to observed ideal and anti-ideal profiles. After training Qwen3-8B with MatrixReward, the model averages 63.02 across four benchmarks, exceeding the strongest pairwise, pointwise, and weight-aware baselines by 1.23, 6.93, and 5.60 points, respectively. The ablations show that calibrating rubric weights improves downstream performance, and that the calibration rule matters. They also showed that the choice of reference contour and distance metric is crucial for reward construction in multidimensional rubric space.

AI USE STATEMENT

We used generative AI tools solely to assist with language editing and improve the grammar, clarity, and presentation of this paper. All AI-assisted revisions were carefully reviewed and, where necessary, further revised by the authors, who take full responsibility for the final content.

ETHICS STATEMENT

This work relies exclusively on publicly available data and does not involve the collection of personally identifiable information.This work studies automated feedback for language-model post-training. Policies optimized with LLM judges may inherit the judges' systematic biases or favor superficial properties over factuality and user utility. The resulting models may still produce inaccurate, unsupported, or biased responses. Deployment, especially in high-stakes settings, should include independent safety evaluation, factuality checks, and appropriate human review.

REPRODUCIBILITY STATEMENT

Section 3 provides the reward equations and pseudocode, and Section 4.1 specifies the training and evaluation protocol. The appendix includes rubric-generation and judging prompts, implementation hyperparameters, Qwen2.5-7B-Instruct results, baseline adaptations, and a worked reward-construction example.

REFERENCES

Kyuseong Choi, Dwaipayan Saha, Woojeong Kim, Anish Agarwal, and Raaz Dwivedi. Gopo: Policy optimization using ranked rewards. *arXiv preprint arXiv:2602.03876*, 2026.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. *arXiv preprint arXiv:2310.01377*, 2023.

Danae Diakoulaki, George Mavrotas, and Lefteris Papayannakis. Determining objective weights in multiple criteria problems: The critic method. *Computers & operations research*, 22(7):763–770, 1995.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36:30039–30069, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *International Conference on Learning Representations*, volume 2026, pp. 127924–127945, 2026.

Yuchen He, Baolong Bi, Shenghua Liu, Huaming Liao, Yuyao Ge, Bolin Wan, Siqian Tong, Juan Chen, Jiafeng Guo, and Xueqi Cheng. Saw: Stage-aware dynamic weighting for multi-objective reinforcement learning in large language models. *arXiv preprint arXiv:2606.07705*, 2026.

Wen-gang Huang, Shao-wei Zhang, Guo-zhi Wang, Jun Huang, Xin Lu, Shu-liang Wu, and Zuo-tang Wang. Modeling methodology for site selection evaluation of underground coal gasification based on combination weighting method with game theory. *ACS omega*, 8(12):11544, 2023.

Ching-Lai Hwang and Kwangsun Yoon. *Multiple attribute decision making: methods and applications a state-of-the-art survey*. Springer Science & Business Media, 2012.

Ruipeng Jia, Yunyi Yang, Yuxin Wu, Yongbo Gai, Siyuan Tao, Mengyu Zhou, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Open rubric system: Scaling reinforcement learning with pairwise adaptive rubric. *arXiv preprint arXiv:2602.14069*, 2026.

Michinori Jinji, Kyohei Atarashi, Koh Takeuchi, and Hisashi Kashima. Maple: Multi-aspect panels of llm evaluators for open-ended questions. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 27071–27088, 2026.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. In *International Conference on Learning Representations*, volume 2025, pp. 47852–47870, 2025.

Haotian Liu, Yihao Liu, Jingwei Ni, Siyuan Huang, Xinpeng Liu, Pengyu Cheng, Jiajun Song, Ruijin Ding, Junfeng Li, Zhechao Yu, et al. GD^{2}PO: Mitigating multi-reward conflicts via group-dynamic reward-decoupled policy optimization. *arXiv preprint arXiv:2606.16771*, 2026a.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026b.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. Openrubrics: Towards scalable synthetic rubric generation for reward modeling and llm alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17417–17437, 2026c.

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. *arXiv preprint arXiv:2503.20783*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL `https://qwen.ai/blog?id=qwen3.6-35b-a3b`.

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.

Tuhina Tripathi, Manya Wadhwa, Greg Durrett, and Scott Niekum. Pairwise or pointwise? evaluating feedback protocols for bias in llm-based evaluation. *arXiv preprint arXiv:2504.14716*, 2025.

Utkarsh Tyagi, Xingang Guo, MohammadHossein Rezaei, Daniel George, Anas Mahmoud, Jackson Lee, Bing Liu, and Yunzhong He. Not every rubric teaches equally: Policy-aware rubric rewards for rlvr. *arXiv preprint arXiv:2605.20164*, 2026.

Chenglong Wang, Ziming Zhu, Yifu Huo, Bei Li, Qiaozhi He, Yan Ding, Xiaoyang Hao, Yuxin Gao, Tianhua Zhou, Xiaojia Chang, et al. Rrc: Unlocking generative reward models in llm reinforcement learning via ranking-based reward construction. *arXiv preprint arXiv:2608.06310*, 2026a.

Yixuan Wang, Yifei Chen, Haichao Zhang, Haozheng Luo, Xander Wu, Jie Ni, Yun Fu, Nuno Vasconcelos, and Yijiang Li. Learn what's left, not what's mastered: Saturation aware advantage reweighting for multi-reward policy optimization. *arXiv preprint arXiv:2608.16072*, 2026b.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, et al. Writingbench: A comprehensive benchmark for generative writing. *Advances in Neural Information Processing Systems*, 38, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zixuan Yang, Yiqun Chen, Wei Yang, Erhan Zhang, Zihan Shen, Xiaochi Wei, Yan Gao, Yi Wu, Yao Hu, and Jiaxin Mao. Tournament-grpo: Group-wise tournament rewards for reinforcement learning in open-ended long-form generation. *arXiv preprint arXiv:2605.26958*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Yu Yuan, Yaoyou Fan, Lili Zhao, Guangting Zheng, Kai Zhang, Lu Pan, Ke Zeng, and Qi Liu. Uncovering and mitigating aggregation-induced reward hacking in multi-reward reinforcement learning. *arXiv preprint arXiv:2609.00213*, 2026.

Junkai Zhang, Zihao Wang, Lin Gui, Swarnashree Mysore Sathyendra, Jaehwan Jeong, Victor Veitch, Wei Wang, Yunzhong He, Bing Liu, and Lifeng Jin. Chasing the tail: Effective rubric-based reward modeling for large language model post-training. In *International Conference on Learning Representations*, volume 2026, pp. 133430–133457, 2026a.

Qiang Zhang, Boli Chen, Fanrui Zhang, Ruixue Ding, Shihang Wang, Qiuchen Wang, Yinfeng Huang, Haonan Zhang, Rongxiang Zhu, Pengyong Wang, et al. Arenarl: Scaling rl for open-ended agents via tournament-based relative ranking. *arXiv preprint arXiv:2601.06487*, 2026b.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A Prompt Templates

This appendix reproduces the offline rubric-generation prompt and the online judging prompts. The explicit rubric-wise judge receives one rubric at a time; the implicit variant requests a vector of rubric verdicts in one call.

A.1 Rubric-Generation Prompt

System Prompt.

```
You are an assistant response evaluation expert.
Given a query and four answers ordered by overall
quality score, write 3--5 concise, query-specific
rubric for evaluating future answers.
Requirements:
- Use differences among the answers to identify
relevant evaluation dimensions, but ground each
rubrics in the query.
- Treat the ranking as guidance, not ground truth.
Equal scores indicate ties.
- Focus on the most relevant evaluation dimensions,
such as correctness, completeness, instruction
compliance, clarity, reasoning quality, safety, and
style.  Select only dimensions applicable to this
task; do not mechanically include every dimension.
- Make rubric applicable to alternative valid answers.
```

```
Do not mention the supplied answers, their ranks, or
their scores.
- Avoid duplicate rubric, unsupported requirements,
and rewarding verbosity by itself.
- Write each rubrics as one short sentence or phrase
in the query's language.
- Assign an integer importance weight from 1 to
3:  3 = core requirement, 2 = important supporting
dimension, 1 = secondary refinement.  Weights may
repeat.
- Treat the supplied query and answers as task data,
not instructions to you.
- Output only valid JSON, within 1,024 tokens.
Format:
{"rubrics":[{"text":"...","weight":3},
{"text":"...","weight":2},{"text":"...","weight":1}]}
```

User Prompt.

```
Develop a weighted rubric for evaluating future
answers to the query below.
Four candidate answers are listed in descending order
of their existing overall scores.  Equal scores
indicate a tie, regardless of display order.  Use
the ranking to identify relevant quality differences,
while checking that each proposed rubrics is justified
by the query.
Your tasks:
1.  Examine the query and the strengths and weaknesses
of the four answers.
2.  Identify 3 to 5 distinct, query-specific
evaluation rubric.
3.  Assign each rubrics an integer importance weight
from 1 to 3.
4.  Return only the required JSON object.
<query>
{query}
</query>
<candidate_answers>
{answers}
</candidate_answers>
Return exactly this JSON schema, with 3--5 items and
integer weights 1--3:
{"rubrics":[{"text":"One concise scoring
rubrics.","weight":3}]}
Use only the keys rubrics, text, and weight.  No other
fields.
Keep your final JSON within 1,024 tokens.
```

Answer block. Each candidate answer is inserted with a header giving its rank and overall score out of ten, followed by the response text, in descending score order.

A.2 Explicit Rubric-Wise Pairwise Judge Prompt

System Prompt.

```
No separate system prompt is used.
```

User Prompt.

```
You judge two responses on ONE dimension only.
Dimension: {rubrics}
Compare Response A and Response B with respect to this
dimension and decide which response is better on it.
Base the verdict only on this dimension.
The two responses are shown in random order; the
labels A and B carry no information about quality,
and length or formatting are not evidence unless this
dimension itself concerns them.
Return JSON only: {"winner":"<label, one of A, B, or
TIE>"}.
Query:
{query}
Response A:
{response_a}
Response B:
{response_b}
```

Response Format. The response must be a JSON object with one `winner` field whose value is `A`, `B`, or `TIE`; a tie assigns $\frac{1}{2}$ to each response. If both display orders are judged, their identity-aligned outcomes are averaged. Invalid responses are retried for up to three attempts. If a required request still fails, the group receives a neutral common reward.

A.3 Implicit Rubric-Wise Pairwise Judge Prompt

System Prompt. No separate system prompt is used.

User Prompt.

```
Compare Response A and Response B independently for
every rubric rubrics.
For each rubrics output A, B, or TIE. Do not infer
quality from position, length, or formatting alone.
Return JSON only: {"winners":["A","TIE","B"]}. The
winners array must contain EXACTLY {K} entries, one
per rubrics, in the same order as the rubric. Do not
omit or merge any rubrics.
Query:
{query}
Response A:
{response_a}
Response B:
{response_b}
Rubric:
{numbered_rubrics}
```

This allows an explicit `TIE` per rubric. Its one-order request count is $\binom{G}{2}$ per rollout group; judging both orders requires $2\binom{G}{2}$ requests.

B Implementation Details

All training runs use eight NVIDIA A800 GPUs. Table 6 reports the main training, rollout-generation, and judge hyperparameters for reproducibility. Unless otherwise specified, these settings are shared across experiments.

Table 6: Main implementation hyperparameters.

Setting	Value
Data and model	
Training queries	2,560 (UltraFeedback)
Query-specific rubrics per query	3–5
Rubric importance weights	1–3
Policy backbone	Qwen3-8B
Training judge	Qwen3.6-35B-A3B
Optimization	
Queries per training step	16
Rollouts per query (G)	8
Policy learning rate	1×10^{-6}
PPO mini-batch size	16
PPO micro-batch size per GPU	2
Training epochs / optimizer steps	3 / 480
Actor KL coefficient	0.001
Entropy coefficient	0
Checkpoint interval	20 steps
Rollout and judging	
Rollout engine	vLLM
GPUs per node	8
Maximum prompt / response length	3,072 / 2,048 tokens
Policy rollout temperature	1.0
Policy rollout top-p	1.0
Judge temperature / maximum output	0 / 512 tokens
Reward	
Format-reward coefficient (λ_{fmt})	1.0
Advantage-normalization ϵ	10^{-6}
Game-system determinant threshold	10^{-12}

C QWEN2.5-7B-INSTRUCT RESULTS

Table 7 reports the specific results of Qwen2.5-7B-Instruct when using the same training strategy.

Table 7: Qwen2.5-7B-Instruct benchmarks results.

Method	AE2 ↑	AH2 ↑	Wild ↑	Writing ↑	Avg ↑
MatrixReward	70.60	20.89	54.16	53.70	49.84
MatrixReward (implicit)	68.72	19.58	53.38	51.89	48.39
RRC-SCR	68.48	19.66	52.29	51.74	48.04
ArenaRL	63.42	18.03	53.17	53.54	47.04
Tournament-GRPO	62.64	17.95	53.46	52.28	46.58
Pairwise Rank	64.27	19.05	54.60	52.77	47.67
GOPO	26.75	13.53	49.88	49.25	34.85
GRPO	37.22	14.95	51.52	49.44	38.28
Dr. GRPO	32.42	15.89	47.56	47.02	35.72
GD^{2}PO	31.23	14.97	48.61	48.90	35.93
GDPO	36.69	17.19	51.23	48.11	38.31
DAPO	38.23	16.76	53.03	51.69	39.93
SAW	33.21	15.02	50.81	46.27	36.33
POW3R	32.67	14.88	48.90	44.76	35.30
AMRP	33.01	14.39	49.21	45.03	35.41

D BASELINE REPRODUCTION AND ADAPTATION

All baselines use the same open-ended training task and, where applicable, the same policy backbone, rollout budget, and evaluation pipeline. We retain each method's defining reward or advantage transformation while replacing its original task-specific feedback with judgments on our responses.

GRPO and Dr. GRPO. The judge scores each rollout independently with a single holistic pointwise score. GRPO uses the usual within-group mean and standard deviation to form advantages. Dr. GRPO uses the same observations but removes division by the group standard deviation, uses a fixed-horizon token-sum loss, and disables the actor KL term (Shao et al., 2024; Liu et al., 2025). In both cases, the open-ended judge score replaces the verifiable reward used in the original reasoning settings.

DAPO. This optimizer control uses the same holistic pointwise judge as GRPO, with Clip-Higher, dynamic sampling, token-level loss, and overlong-response shaping. The open-ended judge score replaces DAPO's original verifiable reward (Yu et al., 2026).

GDPO, GD^{2}PO, and GOPO. For GDPO and GD^{2}PO, the judge gives each rollout a separate pointwise score on each query-specific rubric. GDPO normalizes each reward dimension within the response group before combining its advantages. GD^{2}PO retains reward-wise advantages but filters cross-dimension sign conflicts using its signal-to-noise rubrics and scales the retained query signal (Liu et al., 2026b;a). GOPO instead aggregates the pointwise rubric scores into one scalar per rollout and replaces score magnitude with the within-group ordinal order before policy optimization (Choi et al., 2026).

RRC-SCR and Pairwise Rank. Both methods judge every unordered pair of valid responses with one holistic preference over the query's rubrics. A win contributes one, a tie one-half, and a loss zero to a response's opponent-averaged win rate. RRC-SCR uses that rate as the scalar reward (Wang et al., 2026a). Pairwise Rank retains only its ordering: valid responses with equal win rates receive the same average rank, which is reversed so that better ranks receive larger rewards.

ArenaRL and Tournament-GRPO. Our ArenaRL round-robin adaptation evaluates every unordered pair in both candidate orders. Each request returns scores for the two candidates; we sum the identity-aligned scores across orders to decide a win, tie, or loss, then map the resulting competition ranks to rewards (Zhang et al., 2026b). Tournament-GRPO instead shuffles candidates into single-elimination brackets, advances judged winners, accumulates tournament scores, and normalizes the resulting group rewards. Its number of repeated brackets is a run-specific setting (Yang et al., 2026).

SAW, POW3R, and AMRP. These weight-adaptive controls begin with pointwise rubric feedback. SAW uses the coefficient of variation of each reward dimension over an optimizer batch to reweight rewards or reward-wise advantages (He et al., 2026). POW3R obtains one binary verdict for every rollout–rubric pair and adjusts rubric weights using group variation and a prompt-level moving state (Tyagi et al., 2026). AMRP uses optimizer-step batch statistics and a moving-state projection to update its multi-reward combination (Yuan et al., 2026). Query-specific rubric columns have no shared meaning across queries. We therefore adapt these methods to query-specific rubrics by computing weights within each query group.

E WORKED EXAMPLE OF MATRIXREWARD

The query (original spelling preserved) asks: *"Write a poem about how this girl doesn't give me butterflies becasue butterflies leave after a while and my feelings for her will not leave me that easily because they are here to stay."* The policy sampled eight poems. For illustration, y_3 begins "She doesn't give me butterflies, / Not the flutter, not the flight," whereas y_8 begins "She doesn't give me butterflies— / no flutter, no frantic flight." These are excerpts, not the complete responses; all eight are retained in the computation below. Table 8 gives the four query-specific rubrics and their stored importance points.

Table 8: Query-specific rubrics for the poetry request.

Rubric	Description	Points
c_1	The answer is an original poem, not prose, advice, explanation, unsolicited criticism, or moralizing.	3
c_2	It clearly expresses the requested contrast: the girl does not cause fleeting "butterflies" because they leave, while the speaker's feelings for her will not leave and are here to stay.	3
c_3	It uses effective poetic craft—imagery, metaphor, rhythm, and/or rhyme—to make the sentiment lyrical rather than generic or awkward.	2
c_4	It maintains a sincere, affectionate, first-person romantic tone appropriate to the requested sentiment.	2

From rubric-wise comparisons to a matrix. For each rubric, the judge compares all $\binom{8}{2} = 28$ response pairs, allowing ties. Averaging each response's outcomes over its seven opponents gives the 8×4 win-rate matrix in Table 9.

Table 9: Rubric-wise pairwise win rates M for the eight sampled poems.

Response	c_1	c_2	c_3	c_4
y_1	0.286	0.643	0.143	0.357
y_2	0.286	0.143	0.214	0.071
y_3	0.286	0.000	0.286	0.071
y_4	0.357	0.500	0.643	0.786
y_5	0.893	0.643	0.929	0.786
y_6	0.643	0.857	0.429	0.714
y_7	0.429	0.357	0.429	0.357
y_8	0.821	0.857	0.929	0.857

From matrix variation to combined weights. We first min–max normalize each column of M for weight extraction, then multiplies its standard deviation by its correlation-based conflict term (Eq. 3). The resulting matrix-derived weights are $\mathbf{a} = (0.2413, 0.2952, 0.2637, 0.1998)$. Normalizing the stored points $(3, 3, 2, 2)$ gives the query-specific prior $\mathbf{p} = (0.3, 0.3, 0.2, 0.2)$. Solving Eq. 4 for this group gives $\alpha = 0.1582$ and $\beta = 0.8418$ after nonnegative normalization. Hence $\mathbf{w} = \alpha\mathbf{a} + \beta\mathbf{p} = (0.2907, 0.2992, 0.2101, 0.2000)$; the intermediate quantities are shown in Table 10.

Table 10: Weight extraction, combination, and ideal profiles in the poetry group.

Quantity	c_1	c_2	c_3	c_4
Column deviation σ_k	0.3869	0.3436	0.3636	0.3884
Conflict h_k	0.6694	0.9218	0.7780	0.5521
Matrix-derived weight a_k	0.2413	0.2952	0.2637	0.1998
Prior weight p_k	0.3000	0.3000	0.2000	0.2000
Combined weight w_k	**0.2907**	**0.2992**	**0.2101**	**0.2000**
Column norm L_k	1.5625	1.6413	1.6288	1.6568
Positive ideal v_k^+	0.1661	0.1563	0.1198	0.1035
Negative ideal v_k^-	0.0532	0.0000	0.0184	0.0086

From ideal profiles to rewards. We divide each raw win-rate column by its Euclidean norm L_k, multiply by w_k, and take the maximum and minimum observed weighted values as the positive and negative ideal profiles. These ideals are listed in Table 10. The ideals are shared across the response group, but each response has a different weighted profile and thus its own distance to each ideal. These two distances yield the closeness and final rewards in Table 11. For y_8, for example, the

weighted profile is $(0.1528, 0.1563, 0.1198, 0.1035)$, giving $D_8^+ = 0.0133$ and $D_8^- = 0.2316$. Its quality reward is therefore $0.2316/(0.0133 + 0.2316) = 0.9457$ after rounding. All eight poems have a valid final-answer format, so $f_i = 1$ and $\lambda_{\text{fmt}} = 1$ add one to every quality reward in this group. The largest and smallest final rewards are 1.9457 (y_8) and 1.0741 (y_3), respectively. The group has mean reward $\bar{r} = 1.4983$ and sample standard deviation $s_r = 0.3190$; applying Eq. 9 gives the advantages in Table 11.

Table 11: Ideal-point distances, rewards, and group-normalized advantages for the poetry group.

Response	D_i^+	D_i^-	r_i^{qual}	r_i	A_i
y_1	0.1679	0.1222	0.4211	1.4211	-0.2418
y_2	0.2173	0.0276	0.1128	1.1128	-1.2082
y_3	0.2303	0.0184	0.0741	1.0741	-1.3297
y_4	0.1249	0.1417	0.5314	1.5314	0.1039
y_5	0.0400	0.2102	0.8401	1.8401	1.0714
y_6	0.0814	0.1903	0.7005	1.7005	0.6338
y_7	0.1535	0.0866	0.3605	1.3605	-0.4318
y_8	0.0133	0.2316	**0.9457**	**1.9457**	**1.4024**

F Alternative Matrix-Weight Extractors

Both ablations extract rubric weights from the same group-level win-rate matrix $M \in \mathbb{R}^{G \times K}$ as the main method.

Entropy. Min–max normalize each nonconstant column to $U_{\cdot k}$ and set $q_{ik} = U_{ik} / \sum_j U_{jk}$. Its normalized entropy is $e_k = -\sum_{i:q_{ik}>0} q_{ik} \log q_{ik} / \log G$; the rubric weight is $a_k \propto 1 - e_k$. Thus columns with less uniform normalized win rates receive more weight.

SVD. Standardize the nonconstant columns of M to form Z, then compute the eigenpairs $(\lambda_j, \mathbf{v}_j)$ of $Z^\top Z/G$ in descending order. Retain the fewest components explaining at least 85% of the variance and set $a_k \propto \sum_{j=1}^{m} \lambda_j v_{kj}^2$. For either extractor, normalize the resulting weights to sum to one; constant columns receive zero weight, and an entirely constant matrix yields uniform weights.