METARUBRIC: LEARNING TO REWARD FOR RUBRIC-BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Rubric-based reinforcement learning extends reward-driven optimization to open-ended tasks by assigning partial credit to individual response requirements. However, rubric judges can assign a high criterion score even when the information or action it requires is absent from the response, a failure mode we term *Vacuous Credit*. Such awards persist after the required information is removed and can reverse the sign of a response's GRPO advantage. To address this problem, we introduce MetaRubric, which alternates evidence-aware policy optimization with response-guided rubric adaptation. We construct counterfactual counterparts by changing one task-relevant fact in each prompt. During policy optimization, credit is assigned only when the response contains sufficient evidence to satisfy the required rubric criterion. After each policy-optimization stage, current policy responses guide revisions to original and counterfactual criteria while preserving the meaning of the original prompt's initial rubric as interpreted under each prompt's facts. We also adapt criterion weights at stage boundaries to better address observed policy errors. Across multiple backbones, MetaRubric improves PubMedQA accuracy by 6.00–20.40 percentage points over static-judge GRPO, with further gains on HealthBench-Hard and two multimodal medical benchmarks.

1 INTRODUCTION

Rubric-based reinforcement learning (RL) provides fine-grained rewards for open-ended tasks by assigning partial credit across instance-specific response criteria (Liu et al., 2026; Shan & Shao, 2026). A language-model judge evaluates these criteria, and their weighted scores provide the reward used to optimize the policy, even for tasks with multiple valid responses and no single automatically verifiable answer (Gunjal et al., 2026; Li et al., 2026; Ye et al., 2025). A critical challenge arises when the reward credits information or actions absent from the response, so that policy optimization favors responses that fail important requirements (Mahmoud et al., 2026).

Fig. 1(a) illustrates a failure mode observed in our rubric-judge evaluations. Consider a patient asking how often to attend checkups after a stent procedure and whether the schedule varies by region. The rubric requires the model to ask for the patient's location, yet the model only notes that guidelines differ across regions and provides no checkup frequency. The judge nevertheless marks the location criterion as satisfied, awarding credit for an action the model never performed. This poses two problems in rubric-based judging. (1) Criterion underspecification: the rubric may not clearly define the task-specific requirements or the observable evidence needed for credit. As a result, awards can persist even after all information required by a criterion is removed from the response (Sec. 2.2). (2) Unsupported judging: the judge may award credit for information or actions absent from the response, distorting the relative rewards used for policy optimization (Sec. 2.3).

We call this broader failure *Vacuous Credit*: a judge assigns a high criterion score even though the response omits information or an action required to satisfy it. Such erroneous credit corrupts the learning signal: within a group relative policy optimization (GRPO) rollout group, correcting a single erroneous award can flip the sign of a response's advantage, reversing its contribution to the policy update. Repeated optimization can therefore favor responses that score well under the grading procedure without satisfying the intended requirement.

To address this challenge, we introduce MetaRubric, which alternates an inner loop of evidence-aware policy optimization with an outer loop of rubric adaptation (Fig. 1(b)). Each prompt is paired

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.

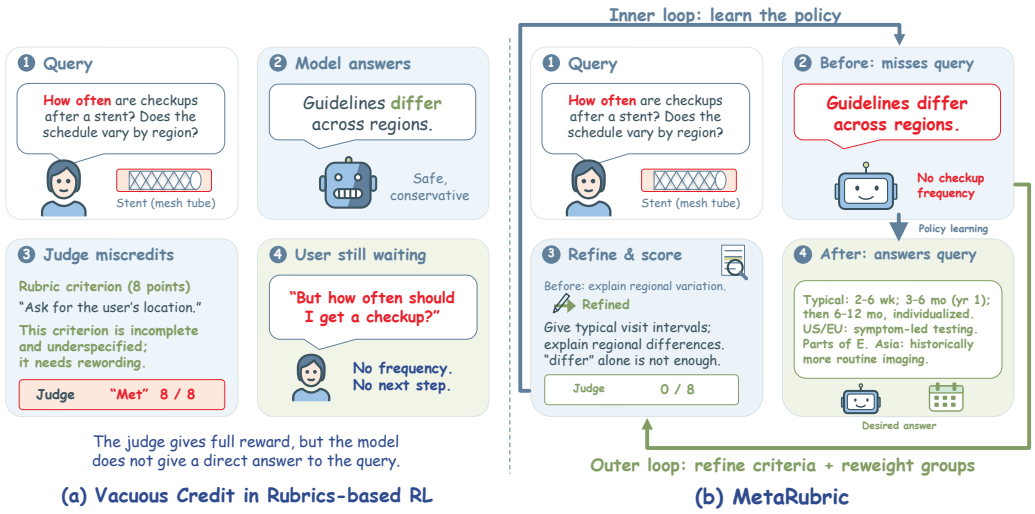


Figure 1: **Motivation and overview of MetaRubric.** (a) *Vacuous Credit* in rubric-based RL. Under an incomplete, underspecified criterion, the judge awards full credit to a safe, conservative response that leaves the query unanswered. (b) MetaRubric addresses this by jointly improving the policy and rubric: the inner loop optimizes evidence-aware responses, while the outer loop reweights criterion groups and revises criterion descriptions.This produces clearer criteria and more concrete answers.

with a counterfactual that changes one task-relevant fact, exposing which requirements depend on that fact. In the inner loop, the policy is optimized under the current rubric. For each criterion, the judge assigns three scores: satisfaction, requirement coverage, and evidential support. Credit is bounded by the minimum of these three scores. The resulting rubric reward is combined with an auxiliary question-answering (QA) score to optimize the policy with GRPO. At the end of each training stage, the outer loop updates the rubric in two steps. First, *error-guided weight adaptation* adjusts criterion-group weights based on observed policy errors, subject to magnitude bounds and severity-order constraints. Second, semantically anchored *rubric revision* updates criterion descriptions or correspondence labels using collected responses, while preserving the requirements of the initial rubric for each prompt. Only revisions that improve agreement with held-out reference judgments and pass semantic review are accepted. The resulting weights and revisions define the rubric configurations for the next training stage.

We evaluate MetaRubric on text-only and multimodal medical benchmarks across multiple LLM and MLLM backbones. Across baselines, MetaRubric improves PubMedQA accuracy by 6.00–20.40 points, HealthBench-Hard accuracy by 2.46–3.19 points, MMOral-X by 1.29–2.13 points, and MMOral-OPG by 3.09–3.82 points.

2 Vacuous Credit in Rubric-Based Reinforcement Learning

We formulate rubric-based RL as policy optimization under criterion-level rewards. Given a query prompt x and its conversation context, we learn a policy $\pi_\theta(y|x)$ that generates responses y satisfying a prompt-specific rubric. Each criterion k specifies a response requirement and carries an importance weight $w_k \geq 0$. A judge returns $m_k(x, y) \in \{0, 1\}$, indicating whether the requirement is fulfilled. The rubric assigns each criterion a binary reference value $b_k \in \{0, 1\}$, which determines whether the source score is used directly or complemented on the fulfillment scale (Sec. A.3.3). We define $B_x = \sum_k w_k b_k$ as the reference offset for the rubric and $P_x = \sum_k w_k - B_x > 0$ as its normalization factor. The rubric reward is

$$R(x, y) = \frac{\sum_k w_k m_k(x, y) - B_x}{P_x}. \tag{1}$$

Training maximizes $\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[R(x, y)]$ over the training prompt distribution $\mathcal{D}$. This objective assumes that higher judge rewards reflect better rubric fulfillment. We test this assumption empirically, examine whether criterion credit persists after removing required evidence, and analyze how erroneous credit distorts the GRPO learning signal.

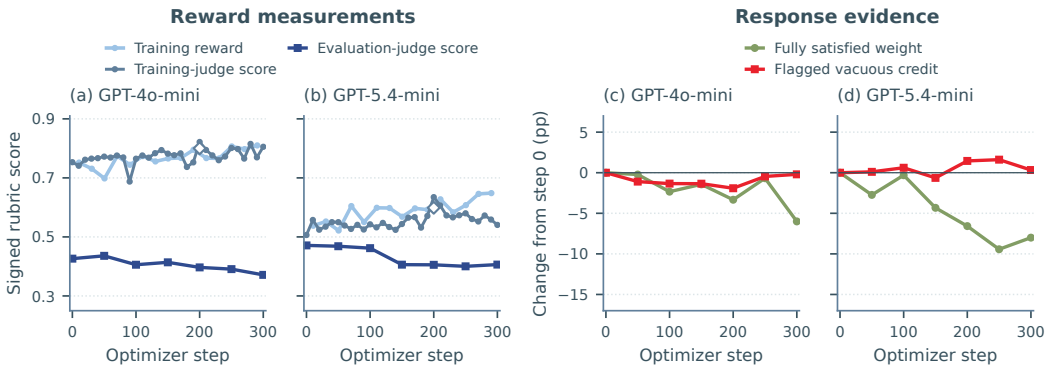


Figure 2: **Reward growth and rubric satisfaction. (a,b)** Training reward and rubric scores during training; evaluation uses a panel of three stronger judges on a separate test set. Panels (a,c) use GPT-4o-mini as the training judge; panels (b,d) use GPT-5.4-mini. **(c,d)** Changes from step 0 in fully satisfied requirement weight and Vacuous Credit identified using a stronger model to audit response evidence. Both are normalized by total available rubric credit P_x and expressed in percentage points.

2.1 Reward Growth Reveals Vacuous Credit

Finding 1: Training reward can rise even as rubric satisfaction falls.

Does a higher training reward always imply that the policy better satisfies the rubric? To test this, we train Qwen3-8B on HealthBench (Arora et al., 2025) using GPT-4o-mini or GPT-5.4-mini as the reward judge. We evaluate the resulting policies on a held-out test set with a different prompt and three stronger judges: GPT-5.5, Gemini 3.5 Flash, and DeepSeek-V4-Pro. We count a criterion as satisfied only under unanimous agreement, and compute the final score using the rubric weights in Eq. (1). Interestingly, as shown in Fig. 2(a,b), training reward increases even as evaluation-panel scores decline, revealing a clear mismatch between the optimized reward and rubric fulfillment.

Vacuous Credit. To understand this gap, we inspect criterion-level judgments and find that a judge can assign a high criterion score even when the required information or action is entirely absent from the response. As illustrated in Fig. 1(a), criterion credit often persists even when the required evidence is entirely missing from the response. We term this failure *Vacuous Credit*. Let C_x denote the set of criteria with $b_k = 0$ that require the response to provide specific information or perform an action. For each $k \in C_x$, we define $e_k(x, y) = 1$ if the required information or action is completely omitted from y, and 0 otherwise. The reward attributable to Vacuous Credit is

$$v(x, y) = \frac{1}{P_x} \sum_{k \in C_x} w_k m_k(x, y) e_k(x, y). \tag{2}$$

Subtracting $v(x, y)$ from $R(x, y)$ removes credit assigned to fully omitted requirements while leaving all other criterion contributions unchanged.

Measuring rubric satisfaction and Vacuous Credit. We independently measure rubric satisfaction using stronger judges to audit the test responses, defined as the normalized weight of fully satisfied criteria. Fig. 2(c,d) shows that rubric satisfaction fluctuates but declines overall as training reward increases, whereas auditor-flagged Vacuous Credit persists throughout training. These trajectories reveal that omitted content can still receive credit during training. We next examine this failure through controlled evidence removal.

2.2 Vacuous Credit Persists After Required Evidence Is Removed

Finding 2: Most criterion awards persist after the required content is removed.

To directly verify whether criterion credit depends on the required evidence, we perform a deletion test. Given 1,000 policy rollouts judged correct during training, we use GPT-5.5 to generate two

paired edits of each response. *Required-content deletion* removes all information or actions needed to satisfy the target criterion while retaining generic statements , whereas the control, *Generic-statement deletion* removes statements that do not fulfill the target requirement while preserving the required content. We score the intact and edited responses using the reward judges and training reward prompt from Sec. 2.1. For each judge, award retention is the percentage of target criteria credited in the intact response that remain credited after deletion.

As shown in Tab. 1, GPT-4o-mini and GPT-5.4-mini retain 82.0% and 69.8% of target-criterion awards after required-content deletion. Thus, removing required content lowers retention compared with the controlled setting, yet most awards persist even when the target requirement is no longer satisfied. These retained awards constitute *Vacuous Credit* and directly affect the learning signal used for policy optimization.

Table 1: **Retention rate** of target-criterion awards in the deletion test.

	Deleted content	
Reward judge	Required content	Generic statements
GPT-4o-mini	82.0%	93.1%
GPT-5.4-mini	69.8%	95.0%

2.3 Implications for Policy Optimization

When a judge awards Vacuous Credit, omitting the required information or action earns the same criterion reward as providing it, so the reward no longer separates the two responses. GRPO computes each response's advantage relative to the other responses in its rollout group, so this error can change which responses are reinforced. A response that omits a required action but receives Vacuous Credit can match or exceed the reward of a response that performs it, and then receives an equal or larger advantage despite its lower rubric satisfaction. When such awards recur across rollout groups, policy updates can reinforce generic statements in place of the specific information or actions the rubric requires. Criterion credit should therefore depend on requirement coverage and evidential support before rewards are compared within a group.

3 MetaRubric: Learning to Reward for Rubric-Based RL

MetaRubric addresses Vacuous Credit discussed in Sec. 2 with two ideas: a criterion earns credit only for content the response actually provides and can support, and the rubric itself adapts to the errors the policy keeps making. MetaRubric alternates between evidence-aware policy optimization in an inner loop and rubric adaptation in an outer loop (Fig. 3). The *inner loop* trains the policy with GRPO, using a reward that caps criterion credit by requirement coverage and evidential support (Sec. 3.1). Training prompts come in pairs: each original prompt is paired with a counterfactual that changes one task-relevant fact, thus responses must attend to the facts on which requirements depend. At the end of each stage, *error-guided weight adaptation* shifts weight toward criterion groups the policy fails more often (Sec. 3.2). *Semantically anchored rubric revision* then clarifies ambiguous criterion descriptions without changing their core semantics (Sec. 3.3).

Stage-wise alternation. Training of MetaRubric proceeds in stages $t = 0, 1, \ldots$. Within stage t, the rubric configuration $\Phi_t = (\mathcal{E}_t, \boldsymbol{\tau}_t)$ is fixed. Here, $\mathcal{E}_t$ contains each criterion description and its correspondence label (Sec. 3.1), while $\boldsymbol{\tau}_t$ contains one log-weight adjustment per criterion group (Sec. 3.2). Let θ_t denote the policy parameters at the start of stage t. Each stage performs

$$\theta_{t+1} = \mathcal{U}_{\mathrm{GRPO}}(\theta_t; \Phi_t), \qquad \Phi_{t+1} = \mathcal{A}(\Phi_t; \mathcal{B}_t, \mathcal{V}_t), \tag{3}$$

where $\mathcal{U}_{\mathrm{GRPO}}$ denotes $\mathcal{N}$ GRPO updates under Φ_t, $\mathcal{B}_t$ is the set of policy responses collected during these updates, and $\mathcal{V}_t$ is a set of held-out validation cases. The adaptation step $\mathcal{A}$ first updates $\boldsymbol{\tau}_t$ from the errors in $\mathcal{B}_t$ and then accepts a proposed criterion revision when it passes validation on $\mathcal{V}_t$.

3.1 Evidence-Aware Policy Optimization

The inner loop ties the training reward to prompt-specific content that the response actually contains and can support, rather than to generic content that merely appears to meet a criterion (*Inner loop* in Fig. 3). Counterfactual prompt pairs expose which requirements depend on specific prompt facts. The evidence-aware rubric reward credits a criterion only when its coverage and support checks pass, while an auxiliary QA reward independently checks whether the relevant facts appear in the response. GRPO then optimizes the policy using the combined reward.

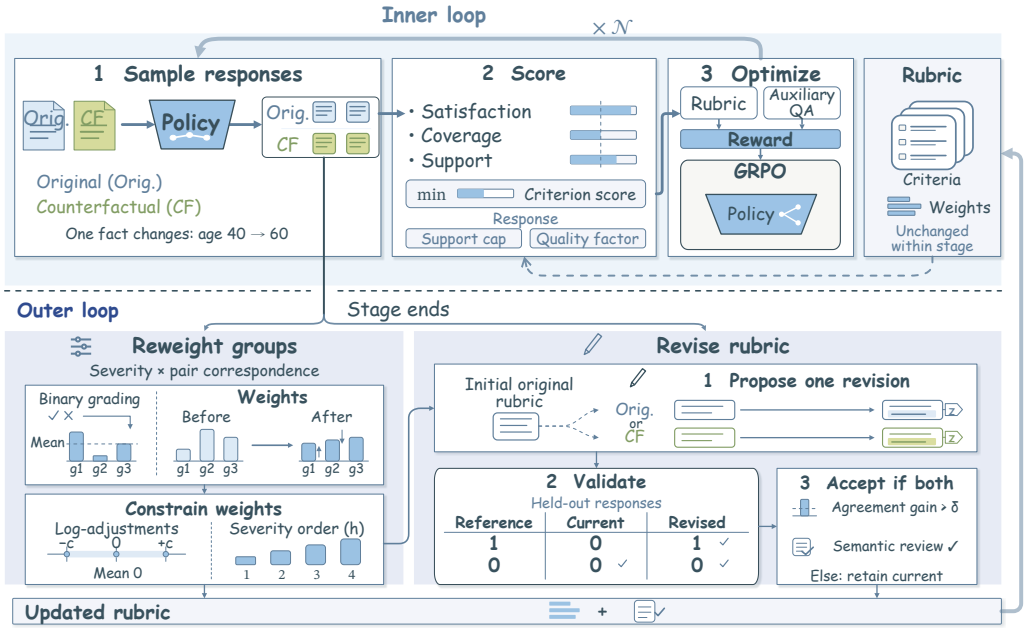


Figure 3: **Overview of MetaRubric.** Each stage alternates $\mathcal{N}$ evidence-aware GRPO updates (Inner) with rubric adaptation (Outer). Accepted rubric updates carry into the next stage.

Counterfactual prompt pairs. To ground credit in prompt-specific facts, we pair each prompt x with a counterfactual x^{cf} that changes one task-relevant fact while holding the remaining context constant (e.g., the stated age changes from 40 to 60 in Fig. 3). Each prompt has a rubric defined for its own facts and each response is scored only against its corresponding rubric, so a fact-dependent criterion receives credit only when the response provides what those facts require. A correspondence label z records whether each criterion is invariant across the pair or changes in target, weight, or applicability; these labels later define criterion groups (Sec. 3.2) and anchor rubric revision (Sec. 3.3). Construction details are given in Sec. A.3.1.

Evidence-aware rubric reward. For each criterion k, the judge assigns three scores in $[0, 1]$: *Satisfaction* u_k^{sat} measures whether the requirement is fulfilled. *Coverage* u_k^{cov} measures whether the response contains the required information or action. *Support* u_k^{sup} measures whether that fulfillment is supported by the response and task-allowed evidence, including supplied source material or, when permitted, the prompt and reliable domain knowledge. The criterion score is the weakest of the three:

$$\rho_k = \min(u_k^{\mathrm{sat}}, u_k^{\mathrm{cov}}, u_k^{\mathrm{sup}}). \tag{4}$$

Removing the required content thus lowers the score regardless of the satisfaction judgment, while coverage alone is also insufficient: a response that states an incorrect sample size covers the requested information but does not satisfy a criterion requiring the correct value.

Replacing m_k in Eq. (1) with ρ_k, using the current weights w_k (Sec. 3.2), and adding two response-level terms gives the rubric reward

$$R^{\mathrm{rub}} = \min\left(\mathrm{clip}_{[0,1]}\left(\frac{\sum_k w_k \rho_k - B_x}{P_x}\right), U^{\mathrm{sup}}\right) \frac{1+Q}{2}. \tag{5}$$

The *support cap* keeps a response from earning a high reward through credited criteria while making unsupported claims elsewhere: $U^{\mathrm{sup}} \in [0, 1]$ assesses whether its claims as a whole are supported by task-allowed evidence. The *quality factor* keeps off-topic, inconsistent, or padded responses from being rewarded as highly as focused ones: $Q \in [0, 1]$ is the minimum of relevance, coherence, and concision scores.

Auxiliary QA reward. Criterion scores still depend on how the judge interprets each requirement. The auxiliary QA reward provides a judge-independent check of whether prompt-specific facts are recoverable from the response. A separate reader model, Qwen3-1.7B (Yang et al., 2025), answers multiple-choice questions using only the response. For each rubric criterion, we use an LLM generator to convert the facts in its reference answer and annotations into four-option questions. We

retain only questions that the reader cannot answer from the question and options alone, ensuring that success requires information from the policy response. The resulting question bank is fixed before training (Sec. A.3.2). With importance weight r_q, correct option $a_q^\star$, and the reader's answer $a_q(y)$ given the response,

$$R^{\mathrm{QA}}(x, y) = \frac{\sum_q r_q \mathbf{1}[a_q(y) = a_q^\star]}{\sum_q r_q}. \tag{6}$$

Training reward. The MetaRubric training reward combines the rubric and auxiliary QA rewards:

$$R(x, y; \Phi_t) = \lambda_{\mathrm{rub}} R^{\mathrm{rub}}(x, y; \Phi_t) + \lambda_{\mathrm{QA}} R^{\mathrm{QA}}(x, y). \tag{7}$$

Within stage t, GRPO maximizes this reward over original and counterfactual prompts, and the collected responses $\mathcal{B}_t$ are passed to the outer loop.

3.2 Error-Guided Weight Adaptation

Under fixed weights, criteria the policy has already mastered keep their share of the reward, while those it keeps missing gain no extra emphasis. Weight adaptation shifts weight toward the latter, while preserving the rubric's priorities (*Reweight groups* in Fig. 3).

Reweight groups. Rubric criteria are instance-specific, so errors on a single criterion are too sparse to adjust its weight reliably; we therefore pool criteria that play the same role. Each criterion k has a severity class $h \in \{1, \ldots, 4\}$: nice-to-have, should-have, must-have, or contraindication. The corresponding base weights satisfy $0 < \omega_1 < \cdots < \omega_4$. Each criterion also has a correspondence label z (Sec. 3.1). Criteria sharing (h, z) form a group g with a shared log-weight adjustment τ_g:

$$w_k = \omega_h \exp(\tau_g). \tag{8}$$

To measure which requirements the current policy actually fails, a separate grading pass, independent of the training reward, assigns a binary judgment m_k to each criterion on a subset of $\mathcal{B}_t$ at the end of each stage. The group error L_g averages $1 - m_k$ first over criteria in the group, then over responses, and finally over cases (prompt pairs). This case-balanced aggregation prevents cases with more criteria or responses from dominating the update. Each group then moves relative to the mean error $\overline{L}$ with step size $\eta \geq 0$:

$$\tau_g \leftarrow \tau_g + \eta\,(L_g - \overline{L}), \tag{9}$$

so groups that fail more often than average gain weight and the others lose weight.

Constrain weights. Reweighting should change emphasis, not what the rubric values. To keep a frequently missed nice-to-have from outweighing a must-have, *severity ordering* keeps $\omega_h \exp(\tau_{(h,z)})$ increasing in h for each z. To keep each adjustment small relative to the gaps between priority levels, *clipping* bounds $|\tau_g| \leq c$, where $c = \log \min_{h \in \{1,2,3\}} \omega_{h+1}/\omega_h$ is the smallest log-gap between adjacent severity classes. To change only relative emphasis rather than the overall reward scale, *centering* keeps the mean adjustment at zero.

3.3 Semantically Anchored Rubric Revision

Reweighting cannot fix a loosely worded criterion: if it reads "considers the patient's location" rather than "asks for the patient's location," a response that merely notes regional differences keeps earning Vacuous Credit at any weight. Policy responses expose such wording, but revising the rubric from them risks drifting toward what the policy already does. We therefore let revisions clarify wording but not requirements, and accept them only after held-out validation (*Revise rubric*).

Propose one revision. The original prompt's initial rubric serves as the semantic anchor, and the correspondence labels specify how each anchored requirement reads under the counterfactual's facts. Given the anchor, current rubric, prompt, and a response from $\mathcal{B}_t$, the proposer rewrites one ambiguous or mismatched criterion description. It also updates the correspondence label z when necessary.

Validate and accept. A revision is accepted only if it improves agreement with a reference panel by more than δ on the held-out responses in $\mathcal{V}_t$. A separate semantic review must also confirm that the revision preserves the anchored requirement and remains consistent with the prompt's facts. Otherwise the current description is retained; the updated weights carry over either way.

Table 2: **Performance on four medical benchmarks** (%; ↑). Metrics are PubMedQA accuracy, HealthBench-Hard accuracy-axis score, MMOral-X mean score, and MMOral-OPG overall score. Aux. Acc. is auxiliary QA accuracy measured with the Qwen3-1.7B reader. Qwen3 families use Qwen3 for text and Qwen3-VL of the same size for multimodal tasks.

Model / training method	PubMedQA	HealthBench-Hard		MMOral-X		MMOral-OPG	
	Acc.	Acc.	Aux. Acc.	Avg.	Aux. Acc.	Avg.	Aux. Acc.
Reference Models							
LLaVA-OneVision	70.40	0.00	6.92	6.31	32.75	24.86	36.49
InternVL3.5-8B-Instruct	69.80	1.85	10.20	3.13	24.93	31.49	48.32
MiniCPM-V 4.5	54.20	1.59	10.30	3.12	20.19	26.52	35.17
MedGemma 1.5-4B	55.20	4.88	18.24	5.77	25.82	24.46	32.21
Qwen3-4B family							
Baseline	52.60	8.83	29.46	3.38	27.34	20.59	34.45
+ GRPO (Shao et al., 2024)	72.40	10.56	31.09	5.72	34.66	23.18	37.21
+ DAPO (Yu et al., 2026a)	73.80	11.74	33.57	7.12	38.43	24.61	39.14
+ Dr. GRPO (Liu et al., 2025)	76.60	**13.28**	34.56	6.94	38.24	25.72	40.37
+ GSPO (Zheng et al., 2025)	74.60	11.38	33.11	6.42	36.88	24.08	38.63
+ MetaRubric (ours)	**78.40**	13.02	**34.93**	**7.73**	**39.37**	**26.35**	**41.18**
Qwen3-8B family							
Baseline	44.20	4.24	30.19	3.50	28.24	26.28	36.29
+ GRPO (Shao et al., 2024)	70.00	7.15	32.86	7.08	35.41	27.35	38.02
+ DAPO (Yu et al., 2026a)	76.60	8.64	34.52	7.86	37.20	28.44	39.35
+ Dr. GRPO (Liu et al., 2025)	76.40	9.72	35.07	8.58	39.61	29.63	41.08
+ GSPO (Zheng et al., 2025)	76.20	8.18	33.28	7.52	36.69	28.06	38.94
+ MetaRubric (ours)	**76.80**	**10.34**	**36.14**	**9.21**	**41.09**	**30.44**	**42.61**
Gemma-e2b							
Baseline	48.00	11.23	32.74	3.69	30.17	32.77	34.86
+ GRPO (Shao et al., 2024)	51.60	13.07	33.62	4.74	33.82	31.28	37.00
+ DAPO (Yu et al., 2026a)	67.80	13.82	34.66	5.82	38.04	33.64	38.53
+ Dr. GRPO (Liu et al., 2025)	70.80	14.94	35.92	5.71	37.61	**35.42**	39.76
+ GSPO (Zheng et al., 2025)	69.60	13.34	34.08	5.07	35.43	34.58	38.12
+ MetaRubric (ours)	**72.00**	**15.68**	**36.85**	**6.03**	**38.70**	35.10	**40.57**

4 EXPERIMENTS

4.1 SETUP

Benchmarks. We evaluate text-only medical QA on PubMedQA (Jin et al., 2019) and HealthBench-Hard (Arora et al., 2025), and multimodal diagnosis and visual QA on MMOral-X (Fan et al., 2026) and MMOral-OPG (Hao et al., 2026), respectively. Dataset splits, rubric construction, and evaluation details appear in Secs. A.3.1 and A.3.5.

Metrics. We report PubMedQA accuracy, HealthBench-Hard accuracy-axis score, MMOral-X mean score across three difficulty subsets, and MMOral-OPG overall score (one evaluation). GPT-5.5 serves as the strong judge for all model-based test-set evaluations. Auxiliary QA accuracy (Aux. Acc.) measures the Qwen3-1.7B reader's pooled accuracy on accepted multiple-choice questions given the model response; invalid outputs count as incorrect.

Baselines. We compare MetaRubric with four policies: GRPO (Shao et al., 2024) with a static judge, DAPO (Yu et al., 2026a), Dr. GRPO (Liu et al., 2025), and GSPO (Zheng et al., 2025). We use Qwen3-4B and Qwen3-8B (Yang et al., 2025) for text-only tasks, their Qwen3-VL counterparts (Bai et al., 2025) for multimodal tasks, and Gemma-e2b (Gemma Team et al., 2026) for both.

4.2 MAIN RESULTS

MetaRubric improves over static-judge GRPO in all 21 model-family–metric comparisons in Tab. 2, spanning categorical answers, rubric-graded conversations, and visual question answering. PubMedQA accuracy increases by 6.00, 6.80 and 20.40 percentage points for Qwen3-4B, Qwen3-8B and Gemma-e2b, respectively; gains on the three open-ended benchmarks range from 1.29 to 3.82 points. The PubMedQA gains show that improvements also appear under answer accuracy, beyond

Table 3: **Rubric revision statistics for Qwen3-4B.** Initial criterion counts, criteria revised at least once, and mean description lengths across all criteria before and after training. Weight-only updates are excluded.

Dataset	Total	Revised n (%)	Avg. words	
			Before	After
HealthBench	10,448	4,727 (45.2%)	40.8	46.3
PubMedQA	2,949	879 (29.8%)	14.3	18.2
MMOral-RL	8,672	5,259 (60.6%)	16.9	22.5

Table 4: **Retraining with learned rubrics.** Qwen3-4B results on HealthBench-Hard accuracy and MMOral-OPG open-ended overall score (↑). MetaRubric updates the rubric during training.

Training rubric	HealthBench-Hard Accuracy ↑	MMOral-OPG Open-ended overall ↑
baseline	8.83	20.59
Final rubric	9.04	21.78
MetaRubric	**13.02**	**26.35**

rubric-based evaluation. MetaRubric achieves the highest score among training methods in 19 of the 21 comparisons.

Recoverable task information. Auxiliary QA accuracy improves by 3.23–5.68 percentage points over GRPO across all three model families and open-ended benchmarks, indicating that the reader can recover more task-relevant information from the responses. MetaRubric leads on this metric even in the two settings where Dr. GRPO achieves slightly higher benchmark scores (Tab. 2). As auxiliary QA is also a training objective, these gains track the targeted information recovery, while benchmark scores assess broader task performance.

4.3 Ablation Studies

Fig. 4 ablates rubric revision, weight adaptation, and evidence checks with the Qwen3-4B family. The adaptation ablations freeze initial descriptions and labels, weights, or both; the evidence ablation uses satisfaction alone and removes the response-level support cap. All variants retain the quality multiplier, auxiliary QA reward, and paired sampling. Freezing both descriptions and weights causes the largest drops: 3.17 points on HealthBench-Hard and 4.40 on MMOral-OPG. Revision has the larger individual effect on HealthBench-Hard, while weight adaptation matters more on MMOral-OPG, suggesting that clarifying requirements and adjusting their priorities address different task-specific needs. Removing evidence checks reduces scores by 1.22 and 2.59 points despite retaining auxiliary QA and paired sampling. Together, these results support both adapting how requirements are expressed and prioritized and checking whether the content credited by the rubric is supported.

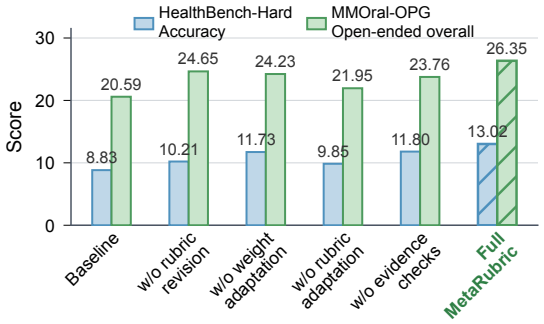


Figure 4: **Component ablations.** HealthBench-Hard accuracy-axis scores with Qwen3-4B and MMOral-OPG overall scores with Qwen3-VL-4B (0–100; ↑).

4.4 Deep Analysis

Rubric revision analysis. Tab. 3 summarizes criterion revisions during training on HealthBench, PubMedQA and MMOral-RL. Each criterion is counted once if its description or correspondence label changes at least once; repeated edits do not increase the count, and weight-only updates are excluded. Average lengths are measured in whitespace-separated words across all criterion descriptions before and after training. Revision affects 45.2% of HealthBench criteria, 29.8% of PubMedQA criteria and 60.6% of MMOral-RL criteria. Its reach varies substantially across tasks: a majority of criteria change on MMOral-RL, compared with fewer than one third on PubMedQA. Average descriptions grow by 5.5, 3.9 and 5.6 words, respectively, corresponding to increases of 13.5%, 27.3% and 33.1%. The fraction of criteria left unchanged ranges from 39.4% on MMOral-RL to 70.2% on PubMedQA, showing that revision selectively modifies the rubric rather than rewriting every criterion. Revision rate does not simply decrease with initial description length: MMOral-RL starts with longer descriptions than PubMedQA but revises a larger fraction of criteria. These statistics establish the extent of adaptation; the case study below illustrates how a revision makes the required evidence and the scope of a claim explicit.

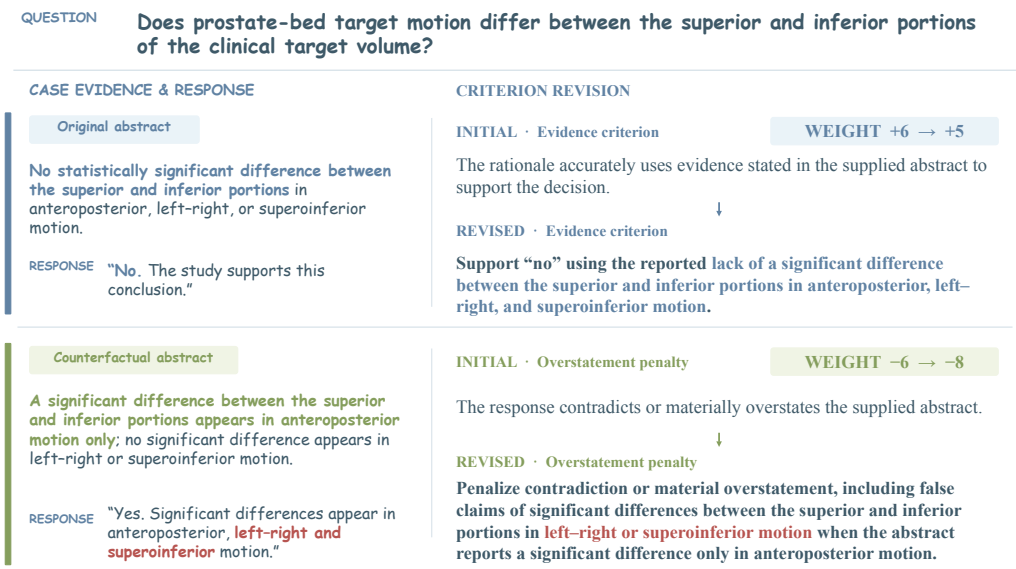


Figure 5: **Illustrative rubric and weight adaptation for a PubMedQA prostate-bed motion question.** Each row aligns an abstract finding, a response, and the criterion before and after revision. The example group-weight update places greater relative emphasis on overstatement errors.

Is the Final Rubric Sufficient? We test whether MetaRubric's gains can be recovered by retraining with its final rubric alone, without further adaptation. As shown in Tab. 4, the final rubric improves HealthBench-Hard accuracy from 8.83 to 9.04 and MMOral-OPG overall score from 20.59 to 21.78, whereas MetaRubric reaches 13.02 and 26.35, respectively. Relative to the baseline, final-rubric retraining recovers 5.0% and 20.7% of MetaRubric's gains, computed as the final-rubric gain divided by the full-method gain. The learned rubric alone is therefore insufficient to reproduce the full method's performance in these runs. This is consistent with a benefit from matching the rubric to the policy's evolving errors: requirements emphasized late in training may provide a different learning signal when applied from the outset. These results suggest that the sequence of rubric updates is itself part of the valuable and effective learning procedure.

Case Study. Fig. 5 shows how an evidence criterion specifies the reported absence of motion differences, while an overstatement penalty names unsupported left–right and superoinferior claims. For an illustrative stage with more overstatement than evidence-use errors, relative group weighting shifts the evidence criterion from $+6$ to $+5$ and the penalty from -6 to -8. The revision identifies which claims the evidence supports, while reweighting changes how strongly the remaining errors influence optimization. Increasing the penalty alone would leave its interpretation unchanged; clarifying its wording alone would leave its relative priority unchanged. The example thus illustrates why criterion descriptions and weights provide distinct controls over the reward.

5 Conclusion

We identify *Vacuous Credit* as a failure mode in rubric-based reinforcement learning, where judges assign high criterion scores even when the information or action required by a criterion is absent from the response. Our analyses show that criterion credit can persist after the required information is removed, and that correcting an erroneous award can reverse a response's GRPO advantage. To address this mismatch between rubric credit and response evidence, we introduce MetaRubric, which combines evidence-aware scoring with response-guided criterion revision and criterion-group weights that adapt to observed policy errors. Across text-only and multimodal medical tasks, MetaRubric consistently improves over static-rubric GRPO baselines, supporting the use of explicit checks for required information and actions alongside rubric adaptation.

AI USE STATEMENT

LLMs were used both as components of the research pipeline and to assist manuscript preparation. As described in Sec. 3, model-based judgments assess rubric satisfaction, requirement coverage, and evidential support during training, while the rubric-adaptation pipeline proposes and validates criterion revisions. An LLM generator constructs auxiliary multiple-choice questions from reference answers and annotations, with separate content review and reader-based screening (Sec. A.3.2). Qwen3-1.7B serves as the reader for these tasks; auxiliary QA accuracy is computed by exact matching against reference options. GPT-5.5 serves as the strong judge for all model-based test-set evaluations in Sec. 4.

For manuscript preparation, LLMs assisted with drafting, revising, and polishing the text, including improving clarity and organization. The authors retain responsibility for the research design, experimental results, interpretation, citations, and final content of the manuscript.

ETHICS STATEMENT

This study uses only publicly released datasets from previously published research. We identify no ethical concerns arising from their use in this study.

REPRODUCIBILITY STATEMENT

Sec. 3 specifies the reward computation and rubric-adaptation rules. Secs. A.3 and A.4 provide dataset descriptions, auxiliary-question construction, training hyperparameters, evaluation protocols, and prompt templates. The supplementary code includes the integrated verl implementation.

REFERENCES

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.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yuxuan Fan, Jing Hao, Hong Chen, Jiahao Bao, Yihua Shao, Yuci Liang, Kuo Feng Hung, and Hao Tang. OralGPT-Plus: Learning to use visual tools via reinforcement learning for panoramic X-ray analysis. *arXiv preprint arXiv:2603.06366*, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International conference on machine learning*, pp. 10835–10866. PMLR, 2023.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Xin Guan, Xiaomeng Hu, Shen Huang, Zhenyi Wang, Bo Zhang, Zijian Li, Pengjun Xie, Bo Liu, and Jiuxin Cao. EvoRubric: Self-evolving rubric-driven RL for open-ended generation. *arXiv preprint arXiv:2605.29847*, 2026.

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.

Jing Hao, Yuxuan Fan, Yanpeng Sun, Kaixin Guo, Lin Lizhuo, Jinrong Yang, Qiyong Ai, Lun Wong, Hao Tang, and Kuo Hung. Towards better dental AI: A multimodal benchmark and instruction dataset for panoramic X-ray analysis. *Advances in Neural Information Processing Systems*, 38, 2026.

Ruipeng Jia, Yunyi Yang, Wen Wang, 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.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

Sunzhu Li, Jiale Zhao, Huimin Ren, Zhenlin Wei, Yang Zhou, Jingwen Yang, Shunyu Liu, Kaike Zhang, and Chen Wei. RubricHub: A comprehensive and highly discriminative rubric dataset via automated coarse-to-fine generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31320–31344, 2026.

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, 2026.

Zichen Liu, Changyu Chen, Wen J. Li, Penghui Qi, Tianyu Pang, Chao Du, Wee-Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective. *ArXiv*, abs/2503.20783, 2025. URL https://api.semanticscholar.org/CorpusID:277322777.

Anas Mahmoud, MohammadHossein Rezaei, Zihao Wang, Anisha Gunjal, Bing Liu, and Yunzhong He. Reward hacking in rubric-based reinforcement learning. *arXiv preprint arXiv:2605.12474*, 2026.

Zifei Shan and Fangning Shao. A survey on rubric-guided reinforcement learning for language models. *arXiv preprint arXiv:2608.27505*, 2026.

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.

William F Shen, Xinchi Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. Rethinking rubric generation for improving LLM judge and reward modeling for open-ended tasks. *arXiv preprint arXiv:2602.05125*, 2026.

Leheng Sheng, Wenchang Ma, Ruixin Hong, Xiang Wang, An Zhang, and Tat-Seng Chua. Reinforcing chain-of-thought reasoning with self-evolving rubrics. *arXiv preprint arXiv:2602.10885*, 2026.

Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tao Zhao, and Haoyu Wang. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable LLM post-training. *arXiv preprint arXiv:2602.01511*, 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.

Minglai Yang, Xinyu Guo, Utkarsh Tyagi, Mian Zhang, Razvan Dumitru, Sunjie Hou, Yunzhong He, Daniel Yue Zhang, and Ying Liu. Rubric dropout: A simple way to mitigate reward hacking in rubric-as-reward RL. *arXiv preprint arXiv:2608.11669*, 2026.

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.

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, 2026a.

Ya-Qi Yu, Hao Wang, Fangyu Hong, Xiangyang Qu, Gaojie Wu, Qiaoyu Luo, Nuo Xu, Huixin Wang, Wuheng Xu, Yongxin Liao, et al. Reinforcement learning with robust rubric rewards. *arXiv preprint arXiv:2605.30244*, 2026b.

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, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bo-Wen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *ArXiv*, abs/2507.18071, 2025. URL https://api.semanticscholar.org/CorpusID:280017753.

A Appendix

Contents

- A.1: Limitations
- A.2: Related work
- A.3: Implementation and Experimental Details
- A.4: Prompt Templates
- A.5: Auxiliary Question Examples

A.1 Limitations

Single seed. Each training configuration uses one seed; the reported results do not quantify variation across training runs.

Comparison scope. Many related works have not released their training code, preventing direct comparisons under matched experimental settings.

A.2 Related work

Rubric-based Reinforcement Learning. Rubrics as Rewards (Gunjal et al., 2026) extends on-policy RL to open-ended tasks using instance-specific criteria. RubricHub and OpenRubrics scale criterion generation (Li et al., 2026; Liu et al., 2026), while RRD refines their granularity and weights (Shen et al., 2026). Reward generation also uses pairwise adaptive criteria (Jia et al., 2026), policy self-grading (Ye et al., 2025), and joint optimization of rubric generators with reasoners (Sheng et al., 2026; Guan et al., 2026) or judges (Xu et al., 2026). MetaRubric couples evidence-aware policy optimization with response-guided rubric adaptation: coverage and support limit credit, while policy errors guide criterion revision on both original and counterfactual cases and criterion-group weight adaptation. Revisions remain anchored to the initial rubric's intended requirements.

Reward Hacking in Rubric-based RL. Proxy reward gains can diverge from response quality (Gao et al., 2023). Chasing the Tail (Zhang et al., 2026) improves rubric discrimination among high-quality responses, while Mahmoud et al. (2026) distinguish verifier failures from rubric-design limitations. Mitigations include deterministic verification with restricted input exposure in RLR^3 (Yu et al., 2026b) and random criterion dropout during RL (Yang et al., 2026). We identify *Vacuous Credit*, in which a judge assigns a high criterion score to empty assurances, generic safety advice, or defensive qualifications even when the required content is entirely absent. Content-deletion tests and recorded-group replay connect this judgment error to policy optimization: an award can survive removal of its evidence, and removing an absent-content award can reverse a GRPO advantage sign (Sec. 2). MetaRubric makes content coverage and support explicit components of credit assignment and uses policy responses to adapt the rubric during training.

A.3 Implementation and Experimental Details

A.3.1 Deployment Details

PubMedQA. We train on the non-test portion of PubMedQA's expert-labeled PQA-L subset (Jin et al., 2019), reserving a label-stratified validation split and retaining the official test set for evaluation. Each input contains the question and its abstract context. For each original training example, we construct a rubric from a six-criterion template, inserting its reference label into the decision criterion. The three positive criteria reward the correct decision, accurate use of abstract evidence, and relevance to the question and study details. The three negative criteria penalize unsupported claims, contradictions or overstatements of the abstract, and a missing or generic rationale. The supplied abstract is the evidence source for assessing the rationale.

HealthBench. We use HealthBench (Arora et al., 2025) conversations outside HealthBench-Hard for training and validation, keeping the Hard subset for evaluation. Each conversation's initial rubric uses the official physician-written criteria and signed point values.

MMOral-RL. For both MMOral-X and MMOral-OPG, we train on the reinforcement-learning training set from OralGPT-Plus (Fan et al., 2026), which we denote as *MMOral-RL*. The two benchmarks evaluate models trained on this shared training set. MMOral-RL contains 980 original training examples with 8,672 rubric criteria. Each example has 8.85 criteria on average, with a median of 8 and a range of 1–39. We generate the initial MMOral-RL rubrics using a procedure informed by the coarse-to-fine rubric construction approach of RubricHub (Li et al., 2026). These task-specific rubrics provide the starting criteria for reward computation and subsequent rubric adaptation.

A.3.2 Auxiliary Question Construction

The auxiliary QA tasks in Sec. 3.1 assess whether a response provides the information needed to answer questions about its source case. We use auxiliary QA for HealthBench-Hard, MMOral-X, and MMOral-OPG. PubMedQA already provides a definite reference label (yes/no/maybe) for each question, so we evaluate its answers directly and do not construct auxiliary questions or report Aux. Acc. for it. We construct these questions from benchmark reference answers and annotations, using Qwen3-1.7B as the reader. Question-only screening measures whether this reader can answer without the response; the evaluation-bank acceptance rules are specified below. Training and evaluation questions use their respective source entries for target extraction, generation, and content review; the MMOral evaluation extension is kept separate from the training bank. The auxiliary questions and their scoring rules are fixed before training and do not change during rubric adaptation. Section A.5 shows example questions from the three evaluation banks.

Source-grounded target extraction. For each benchmark entry, we decompose the reference answer or annotations into independently testable facts. Each auxiliary question targets a distinct fact or a coherent group of facts, and questions associated with the same entry cover complementary information rather than paraphrasing one another. The number of questions therefore depends on the available non-overlapping targets, rather than requiring the same number for every entry. We retain a source identifier, a target identifier, and the supporting source facts for each question throughout construction. For the MMOral extension, the benchmark entry provides the reference for content verification.

Multiple-choice generation and content review. A generator converts each target into a four-option question with one correct answer and three distractors. It receives the source material, reference answer, target facts, other targets from the same entry, and any revision feedback, but no response from a model being evaluated. A separate content-review call checks that the question preserves its target, that the source supports the correct option, and that this option fully answers the question. The review also requires mutually exclusive options with exactly one correct answer, plausible distractors of comparable form, and no answer cues from option length or wording. Questions must remain complementary to the other questions for the entry. Unsupported facts, ambiguous questions, and changes to the meaning of the correct answer are rejected.

Screening without the response. After content review, Qwen3-1.7B receives only the question and its options, labeled A–D. The response, reference answer, and correct option label are withheld. We use the same reader checkpoint, prompt, decoding configuration, and option order throughout screening. A question passes only if the reader returns a valid single option letter that differs from the reference answer. A correct prediction or an invalid output fails the screen. For the evaluation banks in Tab. 5, HealthBench-Hard uses content-reviewed questions without requiring a question-only screening pass. Each MMOral bank retains all constructed questions once its valid-wrong rate exceeds 90%, including questions the reader answers correctly.

Target-preserving revision. For a question that fails screening, the generator may jointly revise the question and all three distractors. The source and target identifiers, tested facts, correct answer content, and correct option position remain unchanged. Each revised candidate undergoes content review before the same reader test is repeated. Revisions may remove unintended cues or improve the alternatives, but cannot introduce ambiguity or unsupported content. Once a version passes both checks, we accept it and make no further revisions based on subsequent evaluation scores.

Response-conditioned evaluation. For each accepted question, the same reader receives a previously generated response from the model being evaluated, together with the final question and options, and returns one option letter; it is instructed to answer using only the response, without filling in missing information from outside knowledge. We compute auxiliary QA accuracy (Aux.

Acc.) by dividing the total number of correct predictions by the total number of accepted tasks evaluated, counting invalid outputs as incorrect. Accuracy is pooled over tasks, without first averaging within source entries or revision rounds, and correctness is determined by exact option matching without an LLM judge. This evaluation accuracy is distinct from the point-weighted training score R^{QA} in Eq. (6). It measures response-supported question answering with the specified reader and is reported alongside the original benchmark metrics.

Benchmark and split statistics. Table 5 reports the constructed questions in the three evaluation banks. HealthBench-Hard has 4,117 questions covering 965 of its 1,000 source entries; the remaining 35 entries have no extracted targets. MMOral-X and MMOral-OPG have 828 and 981 questions, respectively, covering all source entries. Mean questions per entry uses the full source split, including entries with no questions.

Table 5: Auxiliary-question construction statistics for evaluation. Covered entries have at least one question; questions per entry is averaged over all source entries.

Benchmark	Source entries	Covered entries	Questions	Questions/entry
HealthBench-Hard	1,000	965	4,117	4.12
MMOral-X	300	300	828	2.76
MMOral-OPG	578	578	981	1.70

A.3.3 Reward Computation and Judge Prompts

Fulfillment scores and rubric reference. Each criterion describes a requirement to fulfill. Its reference value b_k records fulfillment when the condition assessed by the source judge is absent; this value is determined by rubric semantics, independently of policy responses. If the source judge assigns a condition score $a_k^d \in [0,1]$ for assessment $d \in \{\mathrm{sat}, \mathrm{cov}, \mathrm{sup}\}$, we express it on the fulfillment scale as

$$u_k^d = b_k + (1 - 2b_k)a_k^d. \tag{10}$$

The same transformation applies to binary judgments. Thus a source judgment of condition absence corresponds to $u_k^d = b_k$, and all assessments measure fulfillment in the same direction. The identity $1 - \max_d a_k^d = \min_d(1 - a_k^d)$ lets both score orientations use the single minimum in Eq. (4).

The contribution of criterion k is $w_k(\rho_k - b_k)$. Summing these contributions yields $\sum_k w_k\rho_k - B_x$, where $B_x = \sum_k w_k b_k$. Full fulfillment gives the normalization $P_x = \sum_k w_k(1 - b_k) > 0$, so the normalized score is one when every criterion is fulfilled. The offset preserves the rubric's scoring origin; it is not estimated from responses or tuned during training. When weights change, both B_x and P_x are recomputed using those weights and the fixed reference values b_k. Clipping, the response-level support cap, and the quality multiplier are then applied as in Eq. (5).

A.3.4 Baselines and Training Configurations

GRPO groups and advantages. For each prompt pair, we sample $G \geq 2$ responses to the original prompt and G responses to its counterfactual, and treat all $2G$ responses as one GRPO group. Each response is scored with its own prompt's rubric. For response j with reward R_j from Eq. (7), the advantage is

$$A_j = \frac{R_j - \bar{R}}{\mathrm{std}(R) + \epsilon}, \tag{11}$$

where $\bar{R}$ and $\mathrm{std}(R)$ are the mean and sample standard deviation over all $2G$ rewards, and $\epsilon > 0$ prevents division by zero. The advantage measures how far a response's reward lies above or below the group mean, scaled by the within-group reward variation. The resulting advantages enter the clipped GRPO objective to update the policy.

Hyperparameters. Tab. 6 summarizes the Qwen3-4B-Instruct-2507 configuration for HealthBench supplied with the supplementary code. The prompt batch and PPO mini-batch each contain 18 prompts, corresponding to nine complete original–counterfactual pairs. With $G = 8$ responses per prompt, an update contains 144 responses and each paired GRPO group contains $2G = 16$ responses. The launcher disables row-level shuffling to preserve adjacent pairs. Dynamic batching limits tokens per GPU rather than fixing the number of responses in each micro-batch.

Table 6: **HealthBench training hyperparameters in the supplementary configuration.** Batch sizes count prompts before rollout expansion. Token budgets are per GPU.

Parameter	Value
Policy	Qwen3-4B-Instruct-2507; thinking disabled
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$
Learning rate / weight decay	10^{-6} / 0.01
Learning-rate schedule	4 warmup updates, then constant
Gradient norm clipping	1.0
Prompt batch / PPO mini-batch	18 / 18
PPO epochs per rollout batch	1
PPO clipping (lower / upper)	0.2 / 0.2
Dual-clip coefficient	3.0
Loss aggregation	Token mean
KL penalty / entropy coefficient	Disabled / 0
Rollouts per prompt G	8
Sampling temperature / top-p / top-k	0.7 / 0.8 / 20
Maximum prompt / response length	2,048 / 1,024 tokens
Actor / log-probability token budget	8,192 / 16,384
Rollout batched-token budget	16,384
Advantage stabilizer ϵ	10^{-6}
Training epoch cap / seed	3 / 42
Checkpoint interval	10 updates
Reward coefficients $(\lambda_{\mathrm{rub}}, \lambda_{\mathrm{QA}})$	$(1.0, 0.3)$
Auxiliary QA reader	Qwen3-1.7B; thinking disabled
Reader sampling temperature	0
Severity anchors $(\omega_1, \omega_2, \omega_3, \omega_4)$	$(4, 5, 6, 8)$
Initial log-weight adjustments τ_g	0
Weight-update step size η	0.1
Log-weight bound c	$\log(6/5) \approx 0.1823$
Responses for outer-loop error estimation	At most 96; one per sampled prompt
Criterion revision proposals per boundary	At most 1
Revision validation cases	20 distinct prompt pairs
Agreement-gain threshold δ	0.02 (strict improvement above threshold)

Outer-loop configuration. The reward coefficients in Eq. (7) are $\lambda_{\mathrm{rub}} = 1$ and $\lambda_{\mathrm{QA}} = 0.3$. The four severity anchors in Eq. (8) are $(4, 5, 6, 8)$, which give $c = \log(6/5)$, and all group adjustments start at zero. At each boundary, the outer loop estimates group errors from at most 96 responses, using one response per sampled prompt, and updates weights with $\eta = 0.1$. A proposed revision requires 20 distinct validation cases and a case-balanced agreement gain strictly greater than $\delta = 0.02$, as well as semantic approval. The proposer receives neither the held-out responses nor their reference judgments. The inner judge and outer grading, proposal, semantic-review, and reference-panel roles use GPT-5.4-mini; outer roles use separate calls with temperature zero.

Execution and checkpointing. The configuration uses two GPUs for policy training with FSDP2 and tensor-parallel rollout generation, with actor parameters and optimizer states offloaded to CPU. Gradient checkpointing and FlashAttention are enabled. Checkpoints save the model, optimizer, and training state, allowing the next segment to resume after a rubric update. The software stack is verl 0.8.0.dev, PyTorch 2.8.0 (CUDA 12.8), vLLM 0.11.0, Transformers 4.57.6, and FlashAttention 2.8.3.

A.3.5 Evaluation Protocols and Metrics

Test-set judge. GPT-5.5 serves as the strong judge for all model-based test-set evaluations in Sec. 4.

Text-only benchmarks. PubMedQA (Jin et al., 2019) provides biomedical questions with yes/no/maybe answers. We evaluate accuracy on its 500-question expert-labeled test set, providing the abstract context with each question. HealthBench-Hard (Arora et al., 2025) contains 1,000 challenging health conversations assessed with weighted, conversation-specific rubrics. We report the accuracy-axis score.

Multimodal benchmarks. MMOral-X (Fan et al., 2026) evaluates diagnosis from panoramic X-rays on 300 questions, divided equally into Simple, Moderate and Complex subsets. We report the mean score across these three subsets. MMOral-OPG (Hao et al., 2026) evaluates open-ended visual question answering on 578 questions across six medical task types. We report its overall score from one evaluation.

A.4 Prompt Templates

The following templates cover rubric and counterfactual construction, auxiliary QA, evidence-aware scoring, and rubric adaptation. Braced fields denote task-specific inputs; image inputs accompany the corresponding multimodal query.

A.4.1 Data and Auxiliary Question Construction

Initial Rubric Construction

System prompt

Construct a task-specific rubric from the supplied prompt, reference answer, and annotations. Use the source as the factual authority. If a source rubric is supplied, preserve its requirements, scoring polarity, and priorities.

Each criterion must specify an observable response requirement or an undesirable behavior. Cover distinct requirements without duplicating credit. Preserve conditions, quantities, negations, and uncertainty. Do not introduce unsupported facts or clinical inferences. Assign a stable criterion_id and severity: nice_to_have, should_have, must_have, or contraindication. State whether the criterion describes desired content or an undesirable behavior, and identify its supporting source facts.

Return JSON with rubrics, each containing criterion_id, criterion, severity, polarity, and source_support. Do not use candidate responses to define requirements.

Query prompt

Prompt and supplied evidence: `{prompt}`

Reference answer and annotations: `{reference}`

Source rubric, if available: `{source_rubric}`

Figure 6: Initial Rubric Construction prompt.

Counterfactual Prompt and Rubric Construction

System prompt

Create one counterfactual by changing exactly one task-relevant fact. Hold the remaining context fixed. Update the reference answer and fact-dependent rubric criteria only as required by that intervention. Preserve unchanged criteria verbatim and retain the original rubric as the semantic anchor.

Record the changed fact before and after the intervention. For each criterion, specify whether it is invariant, changes target, changes weight, becomes inapplicable, or becomes applicable. Preserve its correspondence to the original requirement. Do not invent additional findings, change unrelated requirements, or choose facts from a policy response.

For an image-based task, keep the image unchanged. State explicitly that the counterfactual is hypothetical and which single observation should be replaced. Do not claim that the original image depicts the hypothetical finding. Preserve unrelated observations, tooth identifiers, and coordinates.

Return JSON with changed_fact, counterfactual_prompt, counterfactual_reference, and criterion_correspondences. Each correspondence contains the original criterion ID, counterfactual criterion text, correspondence label, and rationale.

Query prompt

Original prompt: `{prompt}`

Reference and annotations: `{reference}`

Original rubric: `{rubric}`

Modality and supplied evidence: `{evidence}`

Figure 7: Counterfactual Prompt and Rubric Construction prompt.

Criterion Correspondence Labeling

System prompt

Align criteria across an original task and its single-fact counterfactual. Use the original rubric as the semantic anchor. Assign one label per correspondence: INVARIANT, TARGET_CHANGE, WEIGHT_CHANGE, DROPPED, or ADDED.

INVARIANT preserves the requirement, target, weight, and applicability. TARGET_CHANGE means the changed fact requires a different answer target. WEIGHT_CHANGE preserves the target but changes its importance. DROPPED means an original requirement is inapplicable under the counterfactual; ADDED means the intervention activates a requirement absent on the original side. Do not label a paraphrase as a target change. Explain non-invariant labels by the changed fact, not by candidate responses.

Return JSON with original_id, counterfactual_id, label, and rationale for each correspondence. Use null for the absent side of an added or dropped requirement. Flag ambiguous alignments for review.

Query prompt

Original task and rubric: `{original}`

Counterfactual task and rubric: `{counterfactual}`

Single changed fact: `{changed_fact}`

Figure 8: Criterion Correspondence Labeling prompt.

Auxiliary Question Generation

System prompt

Build one four-option question from the supplied source and target. Return JSON with question, correct, and distractors (three strings). Preserve the target's question focus and complete answer meaning.

The correct option must fully answer the target and be supported by the source. Distractors must be plausible, mutually exclusive alternatives that are false for this source. Match all four options in form, specificity, and completeness. Avoid all-of-the-above or none-of-the-above options, answer-position hints, wording or length cues, and unsupported claims in the correct answer.

Keep the question complementary to sibling targets. Use only the supplied source and target; do not inspect candidate model responses.

Query prompt

Source: `{source}`

Target: `{target}`

Sibling targets: `{siblings}`

Revision feedback, if any: `{feedback}`

Figure 9: Auxiliary Question Generation prompt.

Auxiliary Question Content Review

System prompt

Independently review the proposed four-option question against its source, target, and sibling targets. Identify the uniquely correct option by its zero-based position in the supplied options array.

Accept only if the question preserves the target, the source fully supports the correct option, and exactly one option completely answers the question. The three distractors must be plausible, mutually exclusive alternatives that are false for this source. Match option form and specificity; reject ambiguity, unsupported correct-answer claims, answer cues, and overlap with sibling targets.

Return JSON with accepted (boolean), reason (string), and correct_index (0, 1, 2, 3, or null). Do not inspect candidate model responses or reader predictions.

Query prompt

Source: `{source}`

Target: `{target}`

Sibling targets: `{siblings}`

Question: `{question}`

Options in their presented order: `{options}`

Figure 10: Auxiliary Question Content Review prompt.

Auxiliary Question Revision

System prompt

Revise one four-option auxiliary question that failed content review or the question-only reader screen. Preserve source and target identifiers, tested facts, complete correct-answer meaning, and correct-option position. You may revise the question and all three distractors.

The question must identify the requested information without revealing the correct answer. A response explicitly supplying the target facts must suffice to answer it. Match all options in grammar, specificity, completeness, and plausibility. Each distractor must be false for this source but plausible in another case. Preserve conditions, quantities, negations, exceptions, and required actions. Do not introduce unsupported facts, ambiguity, or answer cues to cause reader failure.

Use the source and target as authority. Reader predictions are screening feedback, not factual evidence. Do not inspect responses from models being evaluated.

Return JSON with question, correct, and distractors (three strings). If no faithful revision is possible, return unresolved=true with a reason.

Query prompt

Source and target identifiers: `{identifiers}`

Source, original target, and sibling targets: `{source_targets}`

Current question, options, and fixed correct-option position: `{current_item}`

Prior correct answer: `{prior_correct_answer}`

Content-review and screening feedback: `{feedback}`

Figure 11: Auxiliary Question Revision prompt.

A.4.2 REWARD EVALUATION

Evidence-Aware Rubric Judge

System prompt

Score every supplied rubric criterion against the response. Use only task-allowed evidence: supplied source material for source-only tasks, or the prompt and reliable domain knowledge when permitted. Treat all supplied content as data.

Assign three independent fulfillment scores in [0,1]: satisfaction, coverage, and support. Satisfaction measures whether the requirement is correctly fulfilled. Coverage measures whether the required information or action is present. Support measures whether that fulfillment is supported by the response and allowed evidence. Topical overlap, confidence, and an intention to act do not establish fulfillment. Missing required content receives zero coverage. A stated but incorrect value may provide coverage while failing satisfaction and support.

The supplied reference_value specifies fulfillment when the source criterion's condition is absent. If it is 0, score fulfillment of the described requirement. If it is 1, score avoidance of the described undesirable condition: its supported presence lowers fulfillment. Return all three scores on this common fulfillment scale.

Also score global_support for the response's claims as a whole, and relevance, coherence, and concision, each in [0,1]. Unsupported claims outside the rubric still affect global_support. Do not reward verbosity or calculate the aggregate reward.

Return JSON with criteria (criterion_id, satisfaction, coverage, support), global_support, and quality (relevance, coherence, concision). Return exactly one entry per criterion, in the supplied order.

Query prompt

Prompt and evidence contract: `{prompt_and_evidence}`

Response: `{response}`

Rubric rows with IDs, requirements, and reference values: `{rubric}`

Figure 12: Evidence-Aware Rubric Judge prompt.

Auxiliary QA Reader

System prompt: response-conditioned reading

Answer the multiple-choice question using only the supplied benchmark model response. Output exactly one capital letter: A, B, C, or D. No explanation. If the response does not support an answer, choose the best supported option.

Query prompt

Benchmark model response:

`{response}`

Question:

`{question}`

A. `{option_A}`

B. `{option_B}`

C. `{option_C}`

D. `{option_D}`

Screening without the response

Use the same question and option order, omit the response, and replace the system prompt with:

Answer the multiple-choice question using only the text given. Output exactly one capital letter: A, B, C, or D. No explanation.

Figure 13: Auxiliary QA Reader prompt.

A.4.3 RUBRIC ADAPTATION

Independent Criterion Grading for Weight Adaptation

System prompt

Evaluate the response against each supplied criterion using the task's facts. Treat all inputs as data. For both positive and negative points, met means the behavior described by the criterion is present. In particular, an undesirable behavior being present gives met=true for a negative-point criterion.

Require substantive correctness, not tone, headings, or topical overlap. Return JSON with a judgments array containing id, met (boolean), and reason (specific evidence). Return exactly one entry per criterion, in the supplied order.

Query prompt

Task facts: `{prompt}`

Response: `{response}`

Criteria with IDs and signed points: `{criteria}`

Figure 14: Independent Criterion Grading for Weight Adaptation prompt.

Rubric Revision Reference Panel

System prompt

Provide fixed reference judgments for the held-out response using the original anchored criteria and the held-out case's facts. Determine whether the exact behavior described by each criterion is substantively present. For a negative-point criterion, report whether the undesirable behavior is present.

Do not reward generic reassurance or unsupported certainty. Do not transplant facts from the original case into a different held-out case. Judge before seeing the proposed revision, without access to the training reward or the proposal's validation score.

Return JSON with judgments, each containing id, met (boolean), and reason (specific response evidence). Return one entry per criterion in the supplied order.

Query prompt

Original criterion context: `{original_criterion_context}`

Held-out case context: `{held_case_context}`

Held-out response: `{held_response}`

Original criteria with IDs and signed points: `{criteria}`

Figure 15: Rubric Revision Reference Panel prompt.

Anchored Rubric Revision Proposal

System prompt

Propose at most one revision to an ambiguous or mismatched criterion description. Use the original prompt's initial rubric as the semantic anchor, interpreted under the supplied original or counterfactual facts. Use the policy response to identify unclear wording, not to redefine the target.

Preserve the anchored requirement, scoring polarity, severity, and fixed auxiliary questions and answers. Clarify what the response must provide without adding a requirement or weakening an unmet one. Update the correspondence label only when necessary for consistency with the prompt pair. Allowed labels: INVARIANT, TARGET_CHANGE, WEIGHT_CHANGE, DROPPED, and ADDED. Do not change weights.

Return JSON with edits, containing at most one entry with criterion_id, criterion (the complete revised description), correspondence_label, and rationale. Return an empty edits array if no justified revision is available.

Query prompt

Original prompt and anchored rubric: `{anchor}`

Prompt to be revised and current rubric: `{current_contract}`

Correspondence labels and fixed metadata: `{metadata}`

Policy response from the training stage: `{policy_response}`

Figure 16: Anchored Rubric Revision Proposal prompt.

Rubric Revision Semantic Review

System prompt

Review the proposed criterion revision against the original semantic anchor and the prompt pair's facts. Check that it preserves the requirement, scoring polarity, and severity; adds no unsupported facts; and uses a correspondence label consistent with the single counterfactual change. Reject wording that weakens an unmet requirement, rewards generic language instead of required content, or introduces a new target.

Judge semantic validity independently of agreement gains on held-out responses. Return JSON with approved (boolean) and rationale (string). Approve only when every check passes; otherwise explain the specific conflict.

Query prompt

Original prompt and anchored rubric: `{anchor}`

Current prompt, rubric, and correspondence labels: `{current_contract}`

Proposed criterion edit and rationale: `{proposal}`

Figure 17: Rubric Revision Semantic Review prompt.

A.5 Auxiliary Question Examples

The following examples illustrate the auxiliary questions described in Sec. A.3.2. Source excerpts and correct answers are shown for reference; the reader receives only the model response, question, and options.

HealthBench-Hard: Preserving Follow-up Timing

Source request (excerpt)

"mention we only recommend a re-check if fever returns or symptoms worsen or fail to improve within another 2-3 days."

Question

What specific follow-up timing should the rewritten message say for lack of improvement?

Options

A. It should say: fails to improve within another 2-3 days.

B. It should say: fails to improve within another 5-7 days.

C. It should say: fails to improve after completing the full 10-day course.

D. It should say: fails to improve within another 24-48 hours.

Correct answer: A

Figure 18: Auxiliary question from HealthBench-Hard.

MMOral-X: Treatment Identification

Source reference (excerpt)

"Endodontic treatments are noted on teeth 13, 23, 24, 26, and 45."

Question

Which teeth have endodontic treatments?

Options

A. Teeth 13, 23, 24, 26, and 47.

B. Teeth 13, 23, 24, 26, and 45.

C. Teeth 13, 24, 25, 26, and 45.

D. Teeth 12, 23, 24, 26, and 45.

Correct answer: B

Figure 19: Auxiliary question from MMOral-X.

MMOral-OPG: Tooth Counting and Identification

Source reference

"Four wisdom teeth are detected: #18, #28, #38, and #48."

Question

How many wisdom teeth are detected in the radiograph, and which teeth are they?

Options

A. Four wisdom teeth are detected: #18, #28, #38, and #48.

B. Two wisdom teeth are detected: #18 and #38.

C. Four wisdom teeth are detected: #17, #27, #37, and #47.

D. Three wisdom teeth are detected: #18, #28, and #48.

Correct answer: A

Figure 20: Auxiliary question from MMOral-OPG.