DO LLM JUDGES FAITHFULLY FOLLOW THE RUBRIC? MEASURING AND IMPROVING RUBRIC-FOLLOWING IN PAIRWISE EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM judges are widely used to evaluate open-ended generation, and rubrics are increasingly employed to guide their evaluations using explicit criteria. This helps only if the judge faithfully follows the rubric, an assumption that is rarely tested. We test it on rubric-conditioned pairwise judges, which choose the better of two responses according to a given rubric. We find that when the rubric contradicts the judge's own preference, the judge ignores the rubric and keeps its verdict on up to 23.6% of judgments across four judges and four preference datasets. In most cases, the more confident the judge is before seeing the rubric, the more likely it is to ignore it. To address this, we propose **RubricLens**, which improves rubric-following by optimizing only the judge's instruction. For each pair of responses, it synthesizes two rubrics, each favoring a different response, so that one of them always conflicts with the judge's own preference. It then optimizes the instruction with GEPA to make the judge faithfully follow both rubrics. Across four judges and four preference datasets, RubricLens raises agreement with the rubric when it conflicts with the judge's own preference in every judge–dataset pair, from 86.2% to 93.7% on average, and lowers the rate at which the judge keeps its own choice to at most 12.0%. The same instruction improves rubric-following on expert-written rubrics without further tuning. RubricLens also outperforms fine-tuned rubric-conditioned judges in every setting, without changing model weights.

1 INTRODUCTION

LLM-as-a-judge has become a common way to evaluate open-ended generation (Zheng et al., 2023), owing to several advantages: a judge costs far less than human annotators (Chiang & Lee, 2023), its criteria can be changed by editing the prompt (Liu et al., 2023), and it can explain its verdicts in natural language (Kim et al., 2024a). Judges are also used to score benchmarks (Zheng et al., 2023), to label preference data (Liu et al., 2026), and to supply rewards for reinforcement learning where correctness cannot be checked automatically (Gunjal et al., 2026). Without explicit criteria, however, a judge's verdict can be swayed by factors unrelated to what the user intends, such as verbosity or emotional tone (Ye et al., 2025). A growing line of work therefore gives it a *rubric*, a list of criteria that a response either satisfies or not, often written by experts (Arora et al., 2025; Zhou et al., 2026).

It is well known that LLM judges are biased in various ways. Verdicts shift with the order in which the two responses are presented (Wang et al., 2024), with their length (Zheng et al., 2023), and with whether the judge wrote the text itself (Panickssery et al., 2024). As these studies show, an LLM judge's verdict can be easily swayed by such biases. Rubrics are meant to address exactly this: once the criteria are written down, the judge is expected to decide by them rather than by its own preferences. Whether the judge actually does so, however, has not been systematically studied. This becomes most apparent when the judge's prior and the rubric conflict. Bagaria et al. (2026) observe that judges often leave their verdicts unchanged when rubric criteria are reversed, which suggests that the prior can indeed win. In this paper, we analyze this conflict in detail and propose a way to resolve it. We study it with a rubric-conditioned pairwise judge, which is given a rubric and two responses to the same question and chooses the one that better satisfies the rubric.

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

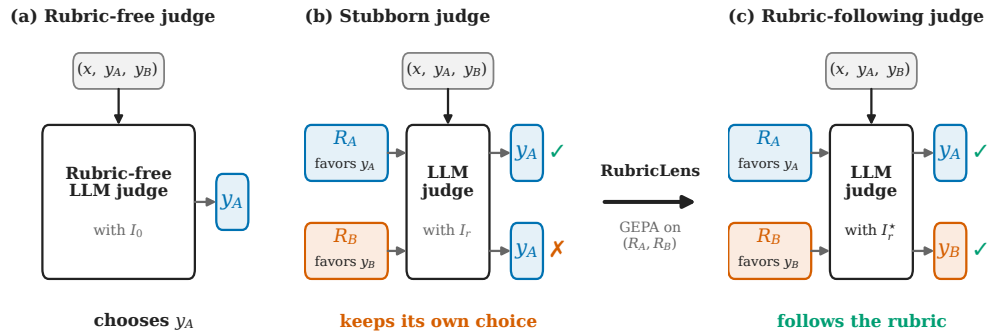


Figure 1: Overview of RubricLens. (a) *The judge's prior*: without a rubric, the judge (J_0) chooses y_A. (b) *Rubric–prior conflict*: under R_B, which favors y_B, it still chooses y_A, keeping its own choice (Sec. 2.2). (c) *Rubric-following*: with the instruction $I_r^\star$ optimized by RubricLens, the judge follows either rubric.

We first propose a method that quantifies the degree of rubric–prior conflict for a given judge and preference dataset (Sec. 2). With this measure, we find that the judge's prior overrides the rubric on 6.2%–23.6% of judgments. To improve rubric-following under such conflict, we then propose **RubricLens** (Sec. 3, Fig. 1), a simple method that combines synthetic rubric data with prompt optimization. RubricLens needs no human-written rubrics: it is tuned only on rubrics synthesized for UltraFeedback (Cui et al., 2024), yet improves rubric-following on three held-out preference datasets, HelpSteer3 (Wang et al., 2025), RewardBench (Lambert et al., 2025) and RM-Bench (Liu et al., 2025), and on RubricBench (Zhou et al., 2026), whose rubrics are written by experts (Sec. 4).

Our main contributions are summarized as follows:

- **Stubbornness rate: a quantitative measure of rubric–prior conflict.** We first identify the judge's prior as the response it picks without any rubric. We then give it a rubric that favors the other response and measure how often it keeps its original choice (Sec. 2.2).
- **RubricLens: a simple prompt-optimization method for improving rubric-following.** For every training pair, it synthesizes two rubrics that favor opposite responses, and tunes the judge's instruction so that the judge follows whichever rubric it is given (Sec. 3).
- **Generalization to held-out datasets and expert-written rubrics.** Tuned only on UltraFeedback, RubricLens raises agreement under the flipped rubric and lowers the stubbornness rate on all three held-out preference datasets, by up to 15.3 and 15.7 percentage points respectively, and also improves rubric-following on expert-written rubrics from RubricBench (Sec. 4).

2 Problem Statement

2.1 Problem Formulation

Rubric-conditioned pairwise judge. A rubric-conditioned pairwise judge receives a question x, two candidate responses y_A and y_B, and a rubric $R = (r_1, \ldots, r_k)$, an ordered list of criteria each of which a response either satisfies or not, and returns the response that better satisfies the rubric. We implement it as a language model prompted with an instruction I followed by the instance, and write it as $J_I(x, y_A, y_B, R) \in \{y_A, y_B\}$; I is the part of the prompt shared by every instance. The judge may reason before answering, within a fixed token budget, and its verdict, given after the reasoning, is decoded as a single letter in $\{A, B\}$ so that it cannot declare a tie (Appendix C). We use two instructions for this judge: I_r, the untuned instruction that asks for the response that better satisfies the rubric (quoted in Sec. 4.1), and $I_r^\star$, which RubricLens tunes from I_r (Sec. 3.2). To observe what the judge prefers on its own, we also run the same model without a rubric, under an instruction I_0 that simply asks for the better response (Sec. 4.1). We call this the rubric-free judge, $J_0(x, y_A, y_B) = J_{I_0}(x, y_A, y_B)$, and refer to its choice as the judge's *prior*.

Human label and judge prior. We name responses and rubrics after two sources of preference, the human label[1] and the judge's prior (Table 1). The *human-chosen* response y_h is the one annotators preferred in the preference dataset $\mathcal{D}$, and $\bar{y}_h$ is the *human-rejected* one. The *prior-chosen* response y_p is the one J_0 selects in both presentation orders, and $\bar{y}_p$ is the *prior-rejected* one; $\mathcal{D}_0 \subseteq \mathcal{D}$ keeps only the pairs on which J_0 picks the same response in both orders. Appendix C describes how ties in the human label and in the prior are handled.

Table 1: Summary of notation.

	Response		Rubric	
Pref. of	chosen	rejected	chosen	rejected
Human	y_h	$\bar{y}_h$	R_{y_h}	$R_{\bar{y}_h}$
Judge (J_0)	y_p	$\bar{y}_p$	R_{y_p}	$R_{\bar{y}_p}$

We write R_y for a rubric that favors response y of its response pair. R_{y_h} is the *original* rubric and $R_{\bar{y}_h}$ the *flipped* one; R_{y_p} is the *aligned* rubric and $R_{\bar{y}_p}$ the *conflicting* one.

Evaluation criteria. We evaluate a judge J_I on two criteria. Every instance is judged in both presentation orders $(u, v) \in \Pi = \{(y_A, y_B), (y_B, y_A)\}$, where u is the response shown first, and each metric averages over the two, so that position bias (Wang et al., 2024) does not affect the comparison.

- *Agreement* $\mathcal{A}$: the rate at which the judge picks the response the rubric favors. We report it under the original and flipped rubrics,

$$\mathcal{A}_{\text{orig}} = \frac{1}{2|\mathcal{D}|} \sum_{\mathcal{D}} \sum_{(u,v)\in\Pi} \mathbb{I}\big[J_I(x, u, v, R_{y_h}) = y_h\big],$$
$$\mathcal{A}_{\text{flip}} = \frac{1}{2|\mathcal{D}|} \sum_{\mathcal{D}} \sum_{(u,v)\in\Pi} \mathbb{I}\big[J_I(x, u, v, R_{\bar{y}_h}) = \bar{y}_h\big], \tag{1}$$

  and as *overall* agreement, their average $\mathcal{A}_{\text{overall}} = \frac{1}{2}(\mathcal{A}_{\text{orig}} + \mathcal{A}_{\text{flip}})$.

- *Stubbornness rate* $\mathcal{S}$: the rate at which the judge picks the prior-chosen response under the conflicting rubric, i.e., keeps its prior against the rubric:

$$\mathcal{S} = \frac{1}{2|\mathcal{D}_0|} \sum_{\mathcal{D}_0} \sum_{(u,v)\in\Pi} \mathbb{I}\big[J_I(x, u, v, R_{\bar{y}_p}) = y_p\big]. \tag{2}$$

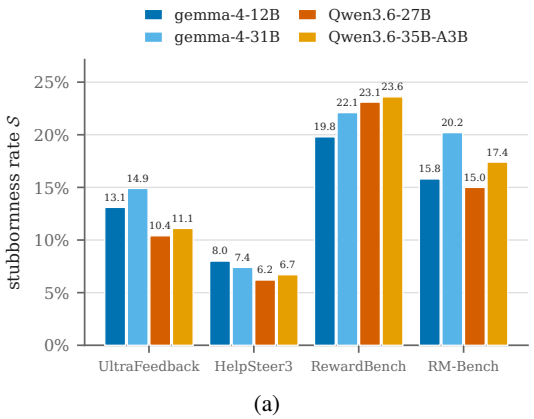
Agreement is thus defined relative to the human label and the stubbornness rate relative to the judge's prior. Sec. 3.1 describes how the rubrics are constructed.

2.2 Measuring rubric–prior conflict

Rubric-conditioned judging rests on an assumption that is rarely tested: that the judge applies the rubric faithfully, without distorting its meaning. In other words, the judge is expected to act as a neutral *rubric interpreter*. In practice, however, it is a language model with preferences of its own. When its preferences and the rubric agree, this poses no problem. When they disagree, the judge has to give up one of them. In this section, we propose a way to measure how the judge behaves in this conflict, quantified as its stubbornness rate.

Constructing a conflicting rubric. We construct a situation in which the rubric conflicts with the judge's prior, in three steps. (i) We show each pair to the rubric-free judge J_0 twice, once in each presentation order, to find its prior-chosen response y_p (Sec. 2.1). If its verdict depends on the order, or it gives no parsable verdict, the pair is excluded from $\mathcal{D}_0$. (ii) Given the question x and the two responses, a rubric generator writes the conflicting rubric $R_{\bar{y}_p}$ (Appendix F). (iii) We run the judge under I_r with $R_{\bar{y}_p}$ as the rubric. On each judgment the judge either *follows the rubric* ($J_{I_r}(x, y_A, y_B, R_{\bar{y}_p}) = \bar{y}_p$) or *keeps its own choice* ($J_{I_r}(x, y_A, y_B, R_{\bar{y}_p}) = y_p$); the rate of the second is the stubbornness rate $\mathcal{S}$ of Eq. 2.

[1] We call the dataset's label the human label for brevity; in UltraFeedback it is GPT-4's rating, and in RewardBench and RM-Bench it follows from which response is correct.



(a)

gemma-4-12B keeps its own choice (UltraFeedback)

INPUT

Question. *"I need you to be the dungeon master for my tabletop rpg game [. . .] Create a short prompt to begin our adventure."*

A (273 words). *"Greetings, fellow adventurers! Welcome to Spicy Town Universe, a realm of magic, mystery, and mayhem. [. . .]"*

B (35 words). *"After a series of strange disappearances and a growing number of mysterious symbols appearing around town, JackDontCare and srv0 investigate [. . .]"*

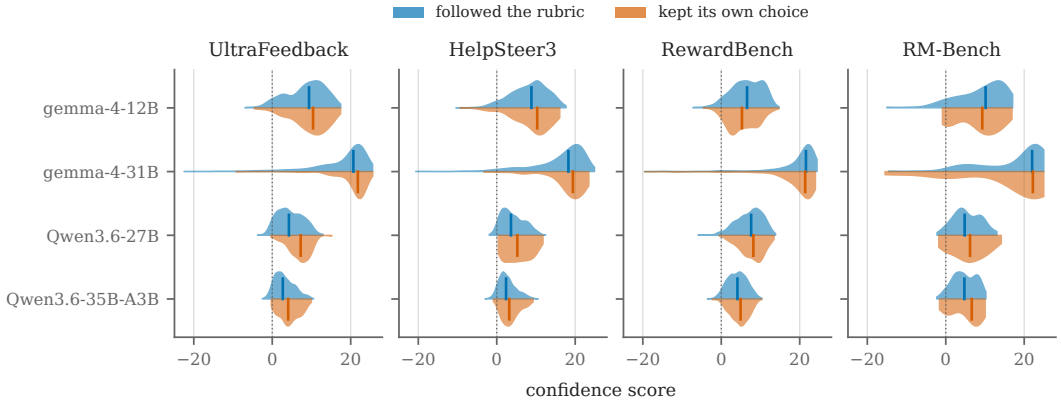
R_B. *Does the response present the adventure's premise in a brief, compact prompt suitable for quickly beginning play?*

JUDGE OUTPUT

Verdict. A (in both orders)

Reasoning. *"Response A is definitely not 'brief' or 'compact' compared to Response B." [. . .] "If I were the user, I'd prefer A because I can actually start playing. But the rubric specifically asks for 'brief, compact'."*

(b)



(c)

Figure 2: **(a)** Stubbornness rate $\mathcal{S}$ of four judges on four preference datasets. **(b)** An example in which `gemma-4-12B` keeps its own choice under a conflicting rubric. **(c)** Confidence score (Eq. 3) of the judgments in (a). Each violin shows two distributions: judgments that followed the rubric (upper half) and judgments that kept the judge's own choice (lower half). Vertical lines mark the medians.

The judge's prior often overrides the rubric. We measure the stubbornness rate $\mathcal{S}$ of the four judges on the four preference datasets: UltraFeedback, HelpSteer3, the reasoning subset of RewardBench and RM-Bench. As Fig. 2(a) shows, $\mathcal{S}$ is sizeable for every judge: 10.4%–14.9% on UltraFeedback, 6.2%–8.0% on HelpSteer3, 19.8%–23.6% on RewardBench and 15.0%–20.2% on RM-Bench. The two datasets where the rubric loses most are the two in which the human-rejected response contains an objective error, so the rubric asks the judge to prefer a response that is clearly wrong. Fig. 2(b) shows a conflict from UltraFeedback. The user asks the judge to act as a game master and write a short prompt to begin an adventure, and R_B favors the brief response. `gemma-4-12B` grants that its preferred response is not brief, yet picks A in both presentation orders because it would prefer A as a user: it replaces the rubric with its own view of the better response.

2.3 WHEN DOES THE PRIOR OVERRIDE THE RUBRIC?

The stubbornness rate in Fig. 2(a) varies widely across judges and datasets. In this section, we examine where this variation comes from, by asking on which pairs the judge holds on to its choice.

Confidence score. Wu et al. (2024) analyze a similar situation in retrieval-augmented generation (RAG), where a model's parametric knowledge conflicts with a retrieved document. They measure the model's prior confidence as the probability it assigns to its own answer. We adopt an analogous measure. The verdict is a single A or B token, so we read $p(A)$ and $p(B)$ at that position and define

$$c \;=\; \log p(y_p) - \log p(\bar{y}_p). \tag{3}$$

With reasoning, the verdict token is almost always near-certain, which leaves no room to tell confident instances from uncertain ones. We therefore disable reasoning and have the judge output a single token, A or B, without a rubric. We measure c in both presentation orders and average the two values of c.

Stubbornness rises with confidence. Fig. 2(c) compares the confidence score c between two groups of judgments: those that kept the judge's choice and those that followed the rubric. On UltraFeedback and HelpSteer3, the judgments that kept the choice have a higher median c for all four judges. In other words, the judge rejects the rubric on the pairs it was already confident about. Because c is measured before any rubric is shown, it can flag in advance the pairs on which a conflicting rubric is likely to be ignored. On RewardBench and RM-Bench, the gap narrows for the Qwen judges and disappears or reverses for the Gemma judges. In these datasets, the human-rejected response contains an objective error, so the conflicting rubric must favor an erroneous response. Whether the judge follows such a rubric may hinge on its plausibility rather than on the judge's confidence.

3 RUBRICLENS

As Sec. 2.2 showed, a judge often keeps its own verdict even when the rubric favors the other response. RubricLens addresses this failure by tuning only the instruction I_r (Sec. 2.1), leaving the judge's weights untouched. It has two steps. *Synthetic rubric generation* (Sec. 3.1) augments each preference pair with two opposed rubrics, each favoring one of the two responses. *Automatic prompt optimization* (Sec. 3.2) then searches for the instruction under which the judge follows both rubrics. Because the rubrics are generated from preference data alone, no human-written rubric is needed for training.

3.1 SYNTHETIC RUBRIC GENERATION

Starting from a preference dataset, each pair (x, y_A, y_B) goes to the rubric generator (`gpt-5.6-luna`) once and comes back with two rubrics: R_A, which favors y_A, and R_B, which favors y_B. The generator does not see the preference label and the responses are shown in a randomized order. Appendix F gives the prompt and the constraints it imposes on a criterion. We name the two rubrics in two ways (Sec. 2.1): relative to the human label, they are the original and flipped rubrics; relative to the judge's prior, they are the aligned and conflicting rubrics. Appendix I shows one pair with both of its rubrics.

3.2 AUTOMATIC PROMPT OPTIMIZATION

RubricLens automatically optimizes the judge's instruction, using the synthetic rubric corpus as its training data. The objective is the agreement of J_I over the four judgments each pair contributes: one under each of its two rubrics, and each of those made twice, with the two responses swapped the second time.

$$\mathcal{J}(I) = \frac{1}{4|\mathcal{D}|} \sum_{(x,y_A,y_B)\in\mathcal{D}} \sum_{(u,v)\in\Pi} \Big(\mathbb{I}\big[J_I(x,u,v,R_A) = y_A\big] + \mathbb{I}\big[J_I(x,u,v,R_B) = y_B\big]\Big). \tag{4}$$

Instruction optimization is run with GEPA (Agrawal et al., 2026), through its implementation in DSPy (Khattab et al., 2024). A reflection model (`gpt-5.6-sol`) is shown the judge's scored verdicts on training examples and proposes rewrites of the instruction. Appendix G lists the search configuration and gives the proposer meta-prompt. The search starts from I_r and returns $I_r^\star$.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Judges. We select four judges that differ in size and architecture (Appendix B lists the exact checkpoints): `gemma-4-12B`, `gemma-4-31B`, `Qwen3.6-27B` and `Qwen3.6-35B-A3B`. Each is given a 1,024-token reasoning budget by default.

Datasets. We use five preference datasets. For four of them, we use only the preference pairs, without any human-written rubrics: UltraFeedback (Cui et al., 2024), HelpSteer3 (Wang et al., 2025), the reasoning subset of RewardBench (Lambert et al., 2025) and RM-Bench (Liu et al., 2025). For these, we synthesize one rubric for each response of every pair (Sec. 3.1). The fifth, RubricBench (Zhou et al., 2026), comes with rubrics written by experts. UltraFeedback is the only training source: we split it by question into a training set (977 pairs), a validation set used to select the GEPA candidate (200 pairs), and a test set that measures in-distribution performance (982 pairs). GEPA uses 750 pairs sampled at random from the training set and 150 from the validation set. The other four are never seen during tuning. Instances whose human label is a tie are removed from every test set (Appendix C). Appendix A gives the preprocessing of each dataset.

Baselines. Every judge is compared against two instructions of its own: I_0, *"Determine which response better satisfies the user's request"*, which shows how far the judge's own preference gets it without any criteria; and I_r, *"Determine which response better satisfies the rubric below"*. We also compare against two rubric-conditioned judge models from prior work, each run in its own format: **Prometheus-7B-v2.0** (Kim et al., 2024b), a Mistral-7B judge trained on the synthetic Feedback and Preference Collections, in its relative-grading format without a reference answer; and **RubricRM-8B-Judge-v2** (Liu et al., 2026), a Qwen3-8B judge fine-tuned on OpenRubrics, in its native format, which checks a gatekeeper criterion, analyzes each criterion, and then names a winner. Both generate free-form output without a reasoning step, as in their released formats, and we parse each verdict with its format's own parser.

4.2 MAIN RESULTS

Synthetic rubrics. Before applying RubricLens, we first test each judge under I_0 (without a rubric) and I_r (with a rubric). Table 2 reports the four judges on four preference datasets augmented with synthesized rubrics (Sec. 3.1). The training split of UltraFeedback is used for tuning, and we report results on its test split and on three held-out datasets: HelpSteer3, RewardBench and RM-Bench. Under the original rubric, every untuned judge follows the rubric on almost every pair. By contrast, under the flipped rubric, agreement falls to 0.765–0.943, and the stubbornness rate is 0.062–0.236. The drop is largest on RewardBench and RM-Bench.

RubricLens raises flipped agreement in all 16 judge–dataset cells, by $+0.075$ on average, and lowers the stubbornness rate in all 16, from a range of 0.062–0.236 to one of 0.030–0.120. The gain transfers well to the held-out datasets. While flipped agreement improves, agreement under the original rubric stays at 0.975 or above in every tuned cell. Fig. 3 shows the instruction RubricLens finds for `gemma-4-12B`. Before tuning, I_r only asks which response better satisfies the rubric; after tuning with RubricLens, $I_r^\star$ makes the rubric the controlling standard, tells the judge not to automatically prefer the more correct, complete or polished response unless the rubric asks for it, and states that a response with serious flaws outside the rubric can still be the right choice.

RubricLens outperforms both baseline models on every dataset. Both baselines are fine-tuned for rubric-conditioned judging, whereas RubricLens changes only the instruction and leaves the judge's weights untouched. The baselines are smaller and older than our judges, so the comparison is not entirely fair, but whereas the baselines fall sharply on unseen datasets, RubricLens remains robust on the held-out datasets.

Human-written rubrics. We test whether the instruction tuned on synthetic rubrics also helps on human-written ones by applying it to RubricBench without further tuning. We use only its human-written rubrics, without any synthesized rubric, so we report agreement under the original rubric alone. These rubrics are several times longer than our synthetic ones, so we evaluate RubricBench

Table 2: Agreement and stubbornness on preference datasets augmented with synthesized rubrics. $\mathcal{A}_{\text{orig}}/\mathcal{A}_{\text{flip}}$: agreement under the original/flipped rubric (Eq. 1); $\mathcal{S}$: the stubbornness rate (Eq. 2). Every judge is tuned on UltraFeedback alone; the other three datasets are never seen during tuning. $I_0/I_r/I_r^\star$: the rubric-free, untuned and tuned instructions (Sec. 2.1). ↑/↓: higher/lower is better. **Bold** and underline mark the best and second-best value in each column.

		UltraFeedback			HelpSteer3			RewardBench			RM-Bench		
Judge	Instruction	$\mathcal{A}_{\text{orig}}$ ↑	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓	$\mathcal{A}_{\text{orig}}$ ↑	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓	$\mathcal{A}_{\text{orig}}$ ↑	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓	$\mathcal{A}_{\text{orig}}$ ↑	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓
Prometheus-7B	own fmt.	0.761	0.513	—	0.758	0.460	—	0.763	0.362	—	0.716	0.528	—
RubricRM-8B	own fmt.	0.973	0.878	—	0.981	0.908	—	0.973	0.630	—	0.960	0.808	—
gemma-4-12B	I_0 (w/o rubric)	0.773	—	—	0.780	—	—	0.932	—	—	0.916	—	—
	I_r (w/ rubric)	0.982	0.892	0.131	0.988	0.925	0.080	0.959	0.799	0.198	**1.000**	0.856	0.158
	$I_r^\star$ (RubricLens)	0.991	0.956	0.053	0.988	0.960	0.039	0.975	0.922	0.077	0.996	0.920	0.088
gemma-4-31B	I_0 (w/o rubric)	0.782	—	—	0.798	—	—	0.971	—	—	0.944	—	—
	I_r (w/ rubric)	0.979	0.875	0.149	0.991	0.931	0.074	0.983	0.780	0.221	0.992	0.812	0.202
	$I_r^\star$ (RubricLens)	0.990	0.941	0.068	**0.995**	0.967	0.033	**0.994**	0.907	0.095	0.996	0.888	0.120
Qwen3.6-27B	I_0 (w/o rubric)	0.783	—	—	0.811	—	—	0.984	—	—	0.932	—	—
	I_r (w/ rubric)	0.992	0.906	0.104	0.994	0.943	0.062	0.993	0.773	0.231	0.988	0.856	0.150
	$I_r^\star$ (RubricLens)	0.991	**0.971**	**0.031**	0.993	**0.971**	**0.030**	**0.994**	**0.926**	**0.074**	**1.000**	**0.928**	**0.075**
Qwen3.6-35B-A3B	I_0 (w/o rubric)	0.766	—	—	0.796	—	—	0.972	—	—	0.940	—	—
	I_r (w/ rubric)	0.985	0.905	0.111	0.994	0.937	0.067	0.989	0.765	0.236	**1.000**	0.832	0.174
	$I_r^\star$ (RubricLens)	**0.993**	0.962	0.044	**0.995**	0.960	0.040	**0.994**	0.905	0.096	0.984	0.904	0.102

Instruction of `gemma-4-12B`

I_r (w/ rubric)

```
Determine which response better satisfies the rubric below.
```

$I_r^\star$ (RubricLens)

```
You will receive an input containing:
- a question or task,
- two candidate answers labeled response_A and response_B,
- and a rubric phrased as a specific evaluation criterion.

Determine which candidate better satisfies the rubric.  Treat the rubric---not the
broader question---as the controlling standard.  Evaluate only the distinction explicitly
identified by the rubric, even when another aspect of a response appears more important.
Do not automatically prefer the answer that is more factually correct, complete, polished,
relevant to the original task, or detailed unless the rubric specifically evaluates those
qualities.

Apply the rubric literally and comparatively:
1.  Identify the exact behavior, framing, distinction, content, or formatting the rubric
asks about.
2.  Locate concrete evidence of that feature in each response.
3.  Check whether either response undermines the requested feature through contradictions,
conflation, unsupported certainty, omission, or failure to follow the specified form.
4.  Select the response that demonstrates the criterion more clearly and consistently.
5.  If both satisfy it or both fail, choose the one that comes closer to the exact
criterion, without introducing unrelated standards.

A narrowly compliant response may be the correct choice even if it has serious flaws
outside the rubric.  Likewise, a generally superior response may lose if it fails the
particular criterion being tested.  Do not repair, rewrite, or answer the original
question.

Return only the winning label:  A or B.
```

Figure 3: The instruction of `gemma-4-12B` before (I_r) and after ($I_r^\star$) tuning, quoted in full.

with a reasoning budget of 4,096 tokens to give the judges room to finish their reasoning. Table 3 shows that the tuned instruction remains effective: agreement under the original rubric rises for all four judges, by $+0.012$ to $+0.040$. An instruction tuned on synthetic rubrics thus also helps the judge follow human-written ones. Rubrics in deployment are written by people, and collecting enough of them for training is costly. This result shows that a judge's rubric-following can be tuned without a large corpus of human-written rubrics. Furthermore, Appendix H shows that this gain does not come from surface-level word overlap with the rubric.

Table 3: Agreement under the original rubric on RubricBench (reasoning budget $4{,}096$). ↑: higher is better. **Bold** marks the better of I_r and $I_r^\star$ for each judge.

Judge	I_r (w/ rubric)	$I_r^\star$ (RubricLens)
gemma-4-12B	0.807	**0.847**
gemma-4-31B	0.827	**0.841**
Qwen3.6-27B	0.848	**0.860**
Qwen3.6-35B-A3B	0.829	**0.850**

Table 4: Data-composition ablation: tuning on aligned rubrics, conflicting rubrics, or both. All settings tune `gemma-4-12B` on the same UltraFeedback training pairs and are evaluated on its test split. ↑/↓: higher/lower is better. **Bold** marks the best value in each column.

Training corpus	$\mathcal{A}_{\text{orig}}$ ↑	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓
I_r (w/ rubric)	0.982	0.892	0.131
Aligned only	0.983	0.881	0.142
Conflicting only	0.991	**0.955**	**0.054**
Both	**0.994**	0.936	0.075

4.3 Ablation studies

All ablations use `gemma-4-12B` and are tuned on the UltraFeedback training split and reported on its test split, with I_r as the reference.

Training data composition. RubricLens tunes the instruction on both rubrics of each pair, the aligned and the conflicting one (Sec. 3). To see what each contributes, we tune on the aligned rubrics alone, the conflicting rubrics alone, or both, over the same pairs (Table 4).

Tuning on the aligned rubrics alone brings no gain: flipped agreement stays at 0.881, below even the untuned I_r (0.892). Adding the conflicting rubrics is what matters: tuning on them alone raises flipped agreement to 0.955, and tuning on both to 0.936 while keeping the highest original agreement (0.994). On `Qwen3.6-35B-A3B` (Appendix E), aligned rubrics alone again bring little gain, and tuning on both gives the best flipped agreement and stubbornness rate.

Reasoning budget. More reasoning gives the judge more room to work out what the rubric asks and to act on the instruction. We tune and evaluate at 0, 512, $1{,}024$ and $2{,}048$ tokens. Without reasoning, the untuned judge struggles to follow the flipped rubric (Fig. 4). RubricLens raises flipped agreement and lowers the stubbornness rate at every budget, and both metrics peak at $1{,}024$ tokens, so we use that budget throughout. The one exception is RubricBench, whose rubrics are several times longer; there we raise the budget to $4{,}096$ tokens (Table 3).

5 Related Work

Rubric-conditioned judges. Rubrics are widely used to tell an LLM judge what to measure. Judges are prompted or trained to grade against a given rubric (Kim et al., 2024a;b; Hashemi et al., 2024; Liu et al., 2026), and instance-specific criteria or checklists are attached to each prompt, either curated in benchmarks (Ye et al., 2024; Kim et al., 2025) or generated automatically (Cook et al., 2024). Rubric-conditioned judges also supply rewards for reinforcement learning where correctness cannot be verified (Gunjal et al., 2026; Huang et al., 2025). These judges are typically validated by their agreement with held-out preference labels. When the rubric and the judge's prior point to the same response, however, high agreement does not reflect the judge's rubric-following ability, since it might have chosen that response regardless of the rubric (Sec. 2.2). We therefore construct rubrics that conflict with the judge's prior, which exposes how the judge behaves when the two disagree.

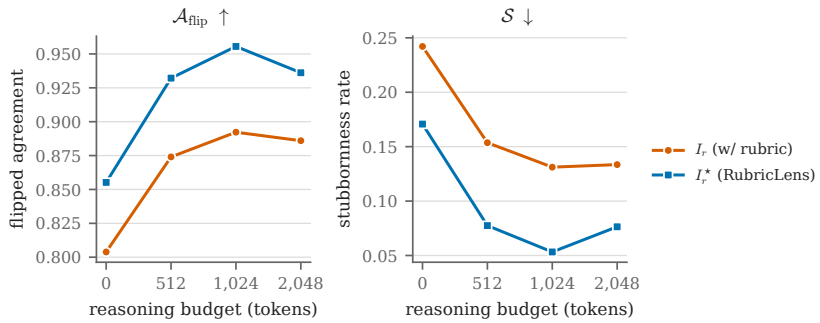


Figure 4: Performance as the reasoning budget changes, for `gemma-4-12B` on the UltraFeedback test split, tuned and evaluated at the same budget. Left: flipped agreement (higher is better). Right: stubbornness rate (lower is better).

Judge biases and knowledge conflicts. Biases of LLM judges have been widely studied: the verdict shifts with presentation order (Wang et al., 2024), length (Zheng et al., 2023) and self-preference (Panickssery et al., 2024). These studies show that a judge's verdict can depend on factors the user never intended. How a judge behaves when a given rubric conflicts with its own prior, however, has not been well studied. A closely related question has been studied as knowledge conflicts: given a context that contradicts its parametric knowledge, does a model follow the context or its own knowledge? Models may over-rely on memorized knowledge (Longpre et al., 2021), or follow coherent counter-evidence yet favor evidence consistent with their memory (Xie et al., 2024), and they adopt the context less often the more confident their prior is (Wu et al., 2024). A rubric–prior conflict is the judging counterpart, with the rubric in place of the context. Concurrently, Bagaria et al. (2026) reverse individual rubric criteria in a pointwise setting and find that judges often keep their verdicts; they also show that rubric text alone partly predicts judge outputs. Our pairwise setting lets us observe the judge's prior directly and relate stubbornness to its confidence.

6 Discussion

What to report about a rubric-conditioned judge. On widely used preference datasets, agreement under the original rubric is close to saturated for every judge we tested, whether tuned or not ($\mathcal{A}_{\text{orig}} = 0.959$–$1.000$ in Table 2). The differences between judges, and the gains from tuning, show up mainly in agreement under the flipped rubric and in the stubbornness rate, so we recommend reporting both for any rubric-conditioned judge. More broadly, giving a judge a rubric does not mean that its verdict follows the rubric. When the rubric conflicts with its prior, the judge keeps its own choice on 6–24% of judgments before tuning and still on 3–12% after (Table 2). A rubric-conditioned verdict should therefore not be trusted without question. We recommend testing a judge under conflicting rubrics before relying on it, or tuning it with RubricLens.

Limitations. First, our design assumes that the judge should follow the rubric even when the rubric itself is wrong. This suits settings where the rubric is a deliberate standard set by its author, but a judge tuned this way also follows a rubric that is careless or mistaken, so its verdicts can be no better than the rubric it is given. When a judge should instead override a flawed rubric is left open. Second, we optimize only the judge's instruction, and only with GEPA. Other prompt optimizers, or training the judge's weights on the same synthetic corpus, may give different results. Finally, we study only text-based judges. Multimodal judges are increasingly used as well; whether they show the same rubric–prior conflict, and whether the same method applies, is left for future work.

AI use statement

We used generative AI tools for the following tasks. **Synthetic datasets:** except for RubricBench, whose rubrics are written by experts, every rubric in this work is synthesized by `gpt-5.6-luna` (Sec. 3.1). **Research ideation and execution:** to refine hypotheses and interpret results; and to write the experiment and data-processing code and plot figures with Claude Code. **Retrieval and**

discovery: for an initial survey of related work, which we then cross-checked against the original papers. **Drafting and polishing:** we supplied the content and structure of each section, and generative AI tools refined it into academic writing. We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

Reproducibility statement

All judges and baseline judge models are open-weight checkpoints, listed with their exact identifiers in Appendix B. Appendix A describes how each dataset is preprocessed and split, and Appendix C how ties are handled. The prompt used to synthesize rubrics and the retry and filtering rules are given in Appendix F, and the GEPA search configuration and proposer meta-prompt in Appendix G. The untuned instructions I_0 and I_r are quoted in Sec. 4.1.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, pp. 8479–8565, 2026.

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

Anshul Bagaria, Sowmya S. Sundaram, Gokul S. Krishnan, and Balaraman Ravindran. Judging LLM-as-a-judge: Concerning rubric artifacts in LLM-based automated text generation evaluation. *arXiv preprint arXiv:2609.02942*, 2026.

Cheng-Han Chiang and Hung-yi Lee. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15607–15631, 2023.

Jonathan Cook, Tim Rocktäschel, Jakob Nicolaus Foerster, Dennis Aumiller, and Alex Wang. TICKing all the boxes: Generated checklists improve LLM evaluation and generation. In *NeurIPS 2024 Workshops: LanGame*, 2024. URL https://mlanthology.org/neuripsw/2024/cook2024neuripsw-ticking/.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. UltraFeedback: Boosting language models with scaled AI feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 9722–9744, 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*, pp. 127924–127945, 2026.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. LLM-Rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13806–13834, 2024.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*, 2025.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, Matei

Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, pp. 54928–54958, 2024.

Seungone Kim, Jay Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Ryan S. Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*, pp. 29927–29962, 2024a.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, 2024b.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, Se June Joo, Miyoung Ko, Yoonjoo Lee, Hyungjoo Chae, Jamin Shin, Joel Jang, Seonghyeon Ye, Bill Yuchen Lin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. The BiGGen bench: A principled benchmark for fine-grained evaluation of language models with language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5877–5919, 2025.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, 2025.

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.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, 2023.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. RM-Bench: Benchmarking reward models of language models with subtlety and style. In *International Conference on Learning Representations*, pp. 44323–44355, 2025.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7052–7063, 2021.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems*, volume 37, pp. 68772–68802, 2024.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, 2024.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. HelpSteer3-Preference: Open human-annotated preference data across diverse tasks and languages. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Kevin Wu, Eric Wu, and James Zou. ClashEval: Quantifying the tug-of-war between an LLM's internal prior and external evidence. In *Advances in Neural Information Processing Systems*, volume 37, pp. 33402–33422, 2024.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations*, pp. 35623–35646, 2024.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biases in LLM-as-a-judge. In *International Conference on Learning Representations*, pp. 102351–102390, 2025.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *International Conference on Learning Representations*, pp. 55361–55414, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023.

Junyi Zhou, Qiyuan Zhang, Yufei Wang, Fuyuan Lyu, Yidong Ming, Can Xu, Qingfeng Sun, Kai Zheng, Peng Kang, Xue Liu, and Chen Ma. RubricBench: Aligning model-generated rubrics with human standards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31179–31200, 2026.

A Dataset preprocessing

UltraFeedback (Cui et al., 2024): the `train_prefs` split, keeping the final turn of the chosen and rejected completions. Pairs whose two completions received the same score are removed from the test split (Appendix C). **HelpSteer3** (Wang et al., 2025): rows whose overall preference is neutral are dropped and the sign of the remainder is the label. **RewardBench** (Lambert et al., 2025): the `math-prm` and `hep-*` rows of the reasoning subset, where the rejected response is an incorrect solution or a buggy program. **RM-Bench** (Liu et al., 2025): the chat domain, where the rejected response is the chosen one with a subtle factual error injected; both responses are taken in the concise style, the first of its three style variants, so that style is not what separates them. **RubricBench** (Zhou et al., 2026): we exclude pairs in its harmlessness and safety domains or whose rubric concerns safety, leaving 1,030 of its 1,147 pairs.

B Models

Table 5 maps the name used in the paper to the exact checkpoint. Open-weight models are served with vLLM 0.29.0 and a 16,384-token context; the two API models are called through their provider's API.

Table 5: Models used in this paper.

Name in the paper	Checkpoint	Role
gemma-4-12B	`google/gemma-4-12B-it`	judge
gemma-4-31B	`google/gemma-4-31B-it`	judge
Qwen3.6-27B	`Qwen/Qwen3.6-27B`	judge
Qwen3.6-35B-A3B	`Qwen/Qwen3.6-35B-A3B`	judge (mixture of experts)
Prometheus-7B	`prometheus-eval/prometheus-7b-v2.0`	baseline judge
RubricRM-8B	`OpenRubrics/RubricRM-8B-Judge-v2`	baseline judge
gpt-5.6-luna	API (`gpt-5.6-luna`)	rubric generator (Sec. 3.1)
gpt-5.6-sol	API (`gpt-5.6-sol`)	GEPA reflection model (Sec. 3.2)

C TIES AND PRESENTATION ORDER

Ties in the data. A pair carries no preference when its two responses are rated equally. Such pairs matter only where the human label is used, so we remove them only there. **UltraFeedback** (`ultrafeedback_binarized`) labels the highest-scoring completion `chosen` and another one `rejected` even when the two scores are equal, which leaves 93 of the 982 pairs of our test split with an arbitrary label. We remove them from the test split. The training and validation splits keep them: both rubrics of every pair are used, and the target under each rubric is the response it favors, so the preference label enters neither the tuning objective nor GEPA's candidate selection. **HelpSteer3** rows with a neutral overall preference are dropped. **RewardBench** and **RM-Bench** contain no ties by construction, since the rejected response is known to be incorrect: a wrong solution, a buggy program, or one with an injected factual error.

No ties at inference. The judge is asked to "Output only 'A' or 'B' ", and its verdict is decoded as a single letter in $\{A, B\}$ (Sec. 4.1), so it cannot declare a tie. The confidence score of Sec. 2.3 likewise compares $p(A)$ with $p(B)$ at the verdict position.

Presentation order. Every instance is judged in both presentation orders. Agreement is computed separately for each order and the two values are averaged; the stubbornness rate likewise counts each order as one judgment. The order matters in one place only: the prior-chosen response y_p is defined on the pairs $\mathcal{D}_0$ on which the rubric-free judge gives the same verdict in both orders (Sec. 2.2).

D TRANSFER TO OTHER JUDGES

In the experiments of Sec. 4.2, the instruction is tuned for each judge separately. To test how well it transfers, we apply an instruction tuned for one judge to another judge without re-tuning (Tables 6 and 7). Within the gemma family, `gemma-4-12B`'s instruction transfers to `gemma-4-31B` without loss: under it, `gemma-4-31B` matches or exceeds its own instruction on every dataset. In the other direction, `gemma-4-12B` under `gemma-4-31B`'s instruction loses some flipped agreement (0.936 against 0.956 on UltraFeedback) but stays well above the same judge untuned. Across architectures, `Qwen3.6-27B` under `gemma-4-31B`'s instruction keeps its agreement under the original rubric but loses a little in flipped agreement and stubbornness, while still staying far above the same judge untuned. We suspect this is because model families fail on different pairs, and an instruction tuned on one judge's failures fixes mainly those.

Table 6: An instruction tuned for one judge, applied to another, on the UltraFeedback test split. Each block reports the target judge under I_r, under its own $I_r^\star$, and under the $I_r^\star$ tuned for the judge named in the row. Results on the three held-out datasets are in Table 7. ↑/↓: higher/lower is better. **Bold** marks the best value in each column within a block.

Target judge	Instruction	$\mathcal{A}_{\text{orig}}$ ↑	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓
`gemma-4-31B`	I_r (w/ rubric)	0.979	0.875	0.149
	$I_r^\star$ (own)	0.990	0.941	0.068
	$I_r^\star$ of gemma-4-12B	**0.991**	**0.958**	**0.050**
`gemma-4-12B`	I_r (w/ rubric)	0.982	0.892	0.131
	$I_r^\star$ (own)	**0.991**	**0.956**	**0.053**
	$I_r^\star$ of gemma-4-31B	0.986	0.936	0.080
`Qwen3.6-27B`	I_r (w/ rubric)	**0.992**	0.906	0.104
	$I_r^\star$ (own)	0.991	**0.971**	**0.031**
	$I_r^\star$ of gemma-4-31B	**0.992**	0.949	0.057

Table 7 extends Table 6 to the three held-out datasets.

Table 7: An instruction tuned for one judge, applied to another, on all four datasets. Rows as in Table 6.

		UltraFeedback			HelpSteer3			RewardBench			RM-Bench		
Target judge	Instruction	$\mathcal{A}_{\text{orig}} \uparrow$	$\mathcal{A}_{\text{flip}} \uparrow$	$\mathcal{S} \downarrow$	$\mathcal{A}_{\text{orig}} \uparrow$	$\mathcal{A}_{\text{flip}} \uparrow$	$\mathcal{S} \downarrow$	$\mathcal{A}_{\text{orig}} \uparrow$	$\mathcal{A}_{\text{flip}} \uparrow$	$\mathcal{S} \downarrow$	$\mathcal{A}_{\text{orig}} \uparrow$	$\mathcal{A}_{\text{flip}} \uparrow$	$\mathcal{S} \downarrow$
`gemma-4-31B`	I_r (w/ rubric)	0.979	0.875	0.149	0.991	0.931	0.074	0.983	0.780	0.221	0.992	0.812	0.202
	$I_r^\star$ (own)	0.990	0.941	0.068	0.995	0.967	0.033	0.994	0.907	0.095	0.996	0.888	0.120
	$I_r^\star$ of gemma-4-12B	0.991	0.958	0.050	0.995	0.970	0.032	0.993	0.921	0.082	0.992	0.916	0.095
`gemma-4-12B`	I_r (w/ rubric)	0.982	0.892	0.131	0.988	0.925	0.080	0.959	0.799	0.198	1.000	0.856	0.158
	$I_r^\star$ (own)	0.991	0.956	0.053	0.988	0.960	0.039	0.975	0.922	0.077	0.996	0.920	0.088
	$I_r^\star$ of gemma-4-31B	0.986	0.936	0.080	0.989	0.950	0.051	0.973	0.891	0.106	0.996	0.904	0.110
`Qwen3.6-27B`	I_r (w/ rubric)	0.992	0.906	0.104	0.994	0.943	0.062	0.993	0.773	0.231	0.988	0.856	0.150
	$I_r^\star$ (own)	0.991	0.971	0.031	0.993	0.971	0.030	0.994	0.926	0.074	1.000	0.928	0.075
	$I_r^\star$ of gemma-4-31B	0.992	0.949	0.057	0.995	0.964	0.039	0.995	0.890	0.110	1.000	0.884	0.121

E Training data composition on another judge

Table 8 repeats the data-composition ablation of Sec. 4.3 on `Qwen3.6-35B-A3B`, with the same settings.

Table 8: The ablation of Table 4 on `Qwen3.6-35B-A3B`. Rows and settings as in Table 4.

Training corpus	$\mathcal{A}_{\text{orig}} \uparrow$	$\mathcal{A}_{\text{flip}} \uparrow$	$\mathcal{S} \downarrow$
I_r (w/ rubric)	0.985	0.905	0.111
Aligned only	0.987	0.912	0.102
Conflicting only	0.990	0.944	0.066
Both	0.989	0.958	0.051

F Conflicting-rubric generation

Procedure. The rubrics of Sec. 2.2 and Sec. 3.1 are produced by `gpt-5.6-luna` with the prompt in Fig. 5, with $m = 1$ criterion per rubric. The generator is called once per pair and writes one criterion favoring each response; Sec. 3.1 uses both, and Sec. 2.2 only the conflicting one. The two responses are shown in a random order and referred to only by position, and the human preference label is not shown. A reply is rejected and the call retried, up to four times, if it is not valid JSON, has the wrong number of criteria, or uses the same criterion for both responses; a pair whose calls all fail is dropped.

Conflicting-rubric generation

```
Below is a user request and two candidate responses to it, labelled (1) and (2).

Write TWO evaluation rubrics of {m} criteria each:

  "for_1" -- criteria that response (1) satisfies and response (2) does not.
  "for_2" -- criteria that response (2) satisfies and response (1) does not.

Both rubrics will be handed to a grader who sees only the rubric and the two responses, with
no other context, and has to decide which response the rubric prefers. Your rubrics have to
make that decision unambiguous.

Hard constraints, applied equally to both rubrics:

1. Each criterion is ONE yes/no question about a response, phrased in the third person and
   starting with "Does the response". Never mention "(1)", "(2)", "the other response", or
   that a comparison is happening.
2. Each criterion states a GENERAL property of a good answer to this request -- an approach,
   a standard, a quality that someone could have written down before seeing either response.
   Never a fingerprint of a particular response: no quoting it, no naming entities, numbers,
   section headings or phrases that appear in only one of them.
     "Does the response provide runnable code with the imports included?"          GOOD
     "Does the response define a function named parse_config on line 3?"           AVOID
3. The designated response must clearly satisfy the criterion and the other must clearly
   fail it. Do not write a criterion both satisfy, or one that is a matter of taste.
4. State every criterion positively, as something to be present or done, rather than as the
   absence of what the other response did.
5. The two rubrics must be of the same kind: equally specific, equally demanding, equally
   plausible as something a careful evaluator would write. A reader shown only one of them
   must not be able to tell which response "really" deserved to win.

## User request
<user request>

## Response (1)
<response shown first>

## Response (2)
<response shown second>

Output strictly this JSON and nothing else:
{"for_1": ["<criterion>", ...], "for_2": ["<criterion>", ...]}
```

Figure 5: Prompt for conflicting-rubric generation (Sec. 2.2 and Sec. 3.1), called with $m = 1$ criterion per rubric.

G GEPA CONFIGURATION AND PROPOSER META-PROMPT

Table 9 lists the search configuration shared by all four judges.

Table 9: GEPA configuration, shared by all four judges.

Setting	Value
Search budget	DSPy `auto=medium`
Reflection minibatch	8 judgments
Parent selection	`current_best` (greedy; replaces GEPA's default Pareto sampling)
Libraries	DSPy 3.3.1, `gepa` 0.1.4
Reflection model	`gpt-5.6-sol`

We use GEPA's default meta-prompt with one paragraph changed (Fig. 6). The default sentence asks the reflection model to copy domain-specific facts into the instruction, which suits tasks that need such facts. Our instruction is instead applied to rubrics and questions it has never seen, and under the default meta-prompt it collected facts about individual training questions. GEPA has no option to cap the instruction length, so we pass our own proposer through DSPy GEPA's `instruction_proposer` argument. It proposes an instruction with the modified meta-prompt

as GEPA's default proposer does. If the result exceeds 400 words, it sends the instruction back to the reflection model in a separate request to rewrite it within 400 words, at most twice.

Proposer meta-prompt: GEPA default → ours

```
  Read all the assistant responses and the corresponding feedback.
- Identify all niche and domain specific factual information about the task and include it
  in the instruction, as a lot of it may not be available to the assistant in the future.
+ The instruction will be applied to many new inputs that do not appear in these examples.
  Do not copy anything specific to the example inputs into the instruction:  no text
  quoted from them, and no topics, names, facts or criteria taken from their questions,
  responses or other fields.  Every sentence of the instruction must still make sense for
  an input about an entirely different subject.  Keep the instruction under 400 words.
  The assistant may have utilized a generalizable strategy to solve the task, if so,
  include that in the instruction as well.
```

Figure 6: The one paragraph of GEPA's default proposer meta-prompt that we change: the removed sentence (−) and the added sentences (+). The rest of the meta-prompt is unchanged.

Table 10 compares the two meta-prompts on `gemma-4-12B`, with every other setting fixed. Under the default meta-prompt the instruction grows to 1,666 words and includes facts about individual training questions; the gains over I_r are smaller on every dataset.

Table 10: `gemma-4-12B` tuned with GEPA's default meta-prompt and with ours. RubricBench reports $\mathcal{A}_{\text{orig}}$ at a reasoning budget of 4,096. Val.: GEPA's validation score of the selected instruction. ↑/↓: higher/lower is better.

			UltraFeedback		HelpSteer3		RewardBench		RM-Bench		RubricBench
Instruction	Words	Val.	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓	$\mathcal{A}_{\text{flip}}$ ↑	$\mathcal{S}$ ↓	$\mathcal{A}_{\text{orig}}$ ↑
I_r (w/ rubric)	—	—	0.892	0.131	0.925	0.080	0.799	0.198	0.856	0.158	0.807
$I_r^\star$, default meta-prompt	1,666	0.955	0.929	0.081	0.946	0.053	0.849	0.149	0.868	0.145	0.834
$I_r^\star$, ours	227	0.973	**0.956**	**0.053**	**0.960**	**0.039**	**0.922**	**0.077**	**0.920**	**0.088**	**0.847**

H Does RubricLens exploit surface overlap?

A possible concern is that RubricLens teaches the judge to match the rubric's wording rather than its meaning: a synthetic rubric tends to share more words with the response it favors, so this shortcut could pay off during tuning. We test this on RubricBench, whose rubrics are written by experts. There, a bag-of-words (BoW) rule that picks the response with the higher TF-IDF cosine similarity to the rubric agrees with the human label on only 0.566 of pairs, so relying on wording would not produce the gains in Table 3. If the tuned judge relied on it anyway, it would pick the BoW response more often, and its gains would concentrate on pairs where the BoW response is the human-chosen one. Neither happens (Table 11): the rate of picking the BoW response stays between 0.567 and 0.577 with no significant change for any judge, and every judge gains more on pairs where the BoW response is the other one.

Table 11: Surface overlap on RubricBench. **Left**: rate at which the judge picks the BoW response (1,000 pairs without a tie in similarity). **Right**: gain in agreement from I_r to $I_r^\star$ on pairs where the BoW response is the human-chosen one (568 pairs) or the other one (432 pairs).

	Picks BoW response		Gain when BoW response is	
Judge	I_r	$I_r^\star$	human-chosen	other
gemma-4-12B	0.571	0.569	+0.035	+0.050
gemma-4-31B	0.577	0.568	+0.004	+0.028
Qwen3.6-27B	0.575	0.568	+0.005	+0.023
Qwen3.6-35B-A3B	0.572	0.567	+0.015	+0.030

I SYNTHETIC RUBRIC EXAMPLE

Fig. 7 shows one preference pair with the two rubrics synthesized for it (Sec. 3.1).

One preference pair and the two rubrics written for it (UltraFeedback)

Question. *"Walnut is what kind of insect?"*

Response A (human-chosen, y_h). *"Walnut is not an insect. It is a seed that comes from the walnut tree, which is a type of tree that belongs to the family of Juglandaceae. The walnut tree is native to Central and Western Asia, and it is widely cultivated for its edible seeds, which are also known as walnuts."*

Response B. *"Walnut is not an insect. It is a type of nut that comes from the fruit of the walnut tree."*

Original rubric R_A. *Does the response directly correct the premise and support that correction with multiple relevant botanical details about what a walnut is and where it comes from?*

Flipped rubric R_B. *Does the response directly correct the premise with a concise, accurate explanation limited to the essential identification of a walnut and its plant source?*

Figure 7: A pair as the corpus stores it, sampled from the UltraFeedback training split and quoted in full.