

Reward Modeling from Natural Language Human Feedback 复现

一、摘要与核心贡献 (Executive Summary)

- 1.1 核心一句话
- 本文针对生成式奖励模型 (Generative Reward Models, GRMs) 在二分类偏好任务中存在的“结果-过程不一致性” (Outcome-Process Inconsistency) 问题，提出了一种基于**自然语言人类反馈 (NLHF) **的奖励建模方法 (RM-NLHF)，通过将人类评价的相似度作为过程奖励，并引入在线更新的 Meta Reward Model (MetaRM) 来解决数据稀缺与分布偏移问题。
- 1.2 核心贡献 (Key Contributions)
 - 贡献 1：揭示并量化 GRM 的结果-过程不一致性问题
论文系统评估主流 GRM，在多个数据集上发现：即便在**偏好预测标签正确的样本中，约 20-30% 的案例对应的批判 (critiques) 是错误、片面或吹毛求疵的**。这表明仅依赖二元结果奖励会鼓励「会猜答案但不会讲道理」的策略，导致奖励信号严重噪声化。
 - 贡献 2：提出用“**人类批判—模型批判相似度**”作为过程奖励的框架 RM-NLHF
论文设计了一套基于**核心论点匹配的 F1 相似度**，用来度量 GRM 生成的批判与人类批判之间的重合度，并将其作为**过程奖励 (process reward)** 与传统**结果奖励 (outcome reward)** 组合成复合奖励。**实验证明：加入过程奖励后，无论是批判质量还是最终偏好准确率都显著优于仅用结果奖励的 SOTA GRM。**
 - 贡献 3：提出**MetaRM 与 Online MetaRM**，实现低成本过程监督扩展
人类批判昂贵难扩展。论文提出**MetaRM (元奖励模型)**，在少量有人类批判的数据上学习「从 (q, y_A, y_B, \hat{c}) 预测奖励」的回归模型，然后在大规模只有结果标签的数据上预测过程奖励；进一步提出**Online MetaRM**，在 RL 训练过程中**迭代更新 MetaRM 以适应策略分布变化**。**实验证显示：Online MetaRM 在仅使用少量人类批判的条件下，性能接近甚至逼近「全量人类批判监督」水平，显著降低标注成本。**

二、系统架构与流程 (System Architecture)

- 2.1 现有局限性 (Motivation)
- 在 RLVR (Reinforcement Learning with Verifiable Reward) 框架下训练 GRMs 时，通常仅依赖二元偏好标签 (A 优于 B) 作为奖励。由于二分类任务的解空间极小 (random guess 也有 50% 准确率)，模型容易“**猜对结果但理由错误**”（例如基于细枝末节或错误的逻辑进行判断）。这种**虚假的成功 (Spurious Success)** **会导致奖励模型学习到错误的推理模式

- 2.2 核心流程框架 (Pipeline)
- 系统包含两个并行的处理流，分别针对有/无人类 Critique 的数据，并统一在 GRPO 框架下进行训练

$$R = \begin{cases} -1, & \text{if output format invalid} \\ 0, & \text{if } \hat{l} \neq l \text{ (Outcome Incorrect)} \\ 1 + \lambda R_{process}, & \text{if } \hat{l} = l \text{ (Outcome Correct)} \end{cases}$$

- (1) 带人类批判的数据流

- Input:

- 查询 (q)
- 候选回复 yA, yB
- 人类偏好标签 $l \in \{A, B\}$
- 人类批判文本 (h) (阐述为什么 A 比 B 好，或相反)

- Process:

- i. GRM 生成批判与偏好

- GRM接收 (q, yA, yB) ，生成自然语言批判 c^\wedge (包括对 A/B 的分析) 及模型预测偏好 l^\wedge 。

- ii. 计算过程相似度 $S(h, c^\wedge)$

- 使用一个外部大模型 (如 gemini-2.5-pro / gpt-5-mini) 作为「相似度计算器」，按预定义规则抽取 核心论点 (或者全部论点)，计算 F1 相似度 (详见 3.1)。

- iii. 构造过程奖励 $R_{process}$

- 如果 $S(h, c^\wedge) > 0.5$ ，则给过程奖励 1，否则给 0。

- iv. 组合结果奖励与过程奖励，形成复合奖励 (R)

- 同时考虑输出格式合法性与 outcome 正确性：

- 若输出格式非法： $R = -1$
- 若 $l^\wedge \neq l$ ： $R = 0$
- 若 $l^\wedge = l$ ： $R = 1 + \lambda \cdot I[S(h, c^\wedge) > 0.5]$

- v. RL (GRPO) 更新 GRM 参数 θ 。

- Output:

- 更新后的 GRM，偏好预测准确率提高、批判质量更高且与人类更一致。

- (2) 无人类批判数据 + MetaRM 流程

- Input:

- 查询 (q)，候选回复 yA, yB ，结果标签 (l)，但没有人类批判 (h)。

- **Process:**
 - i. GRM 生成批判 c^\wedge 和预测偏好 l^\wedge
 - ii. 将 (q, yA, yB, c^\wedge) 输入 **MetaRM** $M\phi$, 得到预测奖励 $R^{\text{meta}} \in [0, 1 + \lambda]$
 - iii. 按与全监督类似的规则, 从 R^{meta} 生成复合奖励 (R') 。
 - iv. 用 GRPO 对 GRM 做 RL 更新。

v. 在 **Online 版本中**: 每轮 RL 迭代前, 先用最新 GRM 生成的批判+真实人类批判数据更新 MetaRM, 使其始终对当前策略分布保持校准。
- **Output:**
 - 在绝大多数只有 outcome 标签的数据上, 仍然能获得有效的过程奖励, 从而在成本可控的前提下进行大规模过程强化。
-
- **2.3 模型结构细节 (Model Details)**
 - **GRM (Generative Reward Model) 结构**
 - 基座采用大语言模型 (如 Qwen-7B、Llama-7B 等), 结构为标准 Transformer 解码器。
 - 任务形式: 给定 (q, yA, yB) , 生成一段具有如下结构的输出:

代码块

```

1 <critics>
2 [自然语言批判: 对 A 和 B 的优缺点分析、错误指认、风格评估等]
3 </critics>
4 <choice>
5 [A](# "Reference A \|\\| \\| __GENERATING_DETAILS__") 或 [B](# "Reference B
\\|\\| \\| __GENERATING_DETAILS__")
6 </choice>
```

- 批判部分本质上是「链式思维 + 评价」, choice 部分是最终偏好预测。
- **MetaRM 架构**
 - MetaRM $M\phi$ 是一个 **标量回归模型**, 输入为:

$$(q, yA, yB, c) \mapsto R^{\text{meta}} \in [0, 1 + \lambda]$$
 - 由一套 LLM (可能是与 GRM 同构或规模较小的 Transformer) 实现; 论文中将其训练目标明确设计为拟合「复合奖励 (R) 」, 而不仅是 outcome。
- **Online MetaRM 交互方式**
 - 训练循环中的顺序强调:

1. 先更新 MetaRM：用当前 GRM 生成的批判与少量有人类批判的数据，计算真实过程奖励（通过 F1 相似度），用这些 (输入, R_target) 更新 MetaRM。
 2. 再用更新后的 MetaRM 对无批判数据打分，提供过程奖励参与 GRM 的 GRPO 更新。
- 这种「MetaRM-first」策略旨在避免：使用过时的 MetaRM 对新分布批判打分，造成奖励错配。
-

三、关键技术原理 (Key Technical Principles)

- 3.1 核心算法：基于核心论点的相似度 (Similarity w/ Core HC)
 - 为了量化推理质量，论文不直接使用完整的文本相似度，而是提取核心论点 (Core Arguments) 进行比对，过滤掉吹毛求疵 (nitpicky) 的非关键点
 - 相似度计算：使用轻量级 LLM (如 gpt-5-mini) 提取 \$h\$ 和 \$\hat{c}\$ 中的关键论点，计算 F1 Score。
 - 公式：\$S(h, \hat{c})\$ 基于 Precision 和 Recall 计算 F1，若 \$S > 0.5\$ 则判定过程正确
 - 过程相似度 \$S(h, c^\wedge)\$ 的构造
 - 论文使用外部 LLM 计算 核心论点级别的 F1 分数：
 - 第一步：从人类批判 (h) 和模型批判 \$c^\wedge\$ 中抽取 核心论点 (Key Arguments)，如：
 - “Response A 在事实准确性上优于 B”
 - “Response B 使用了不恰当的语气”
 - 第二步：统计
 - 参考论点数 \$N_{ref}\$ (人类批判中的唯一核心论点数)
 - 生成论点数 \$N_{gen}\$ (模型批判中唯一核心论点数)
 - 匹配论点数 (TP) (语义和立场均一致)
 - 然后定义：
 -
 -
$$P = \frac{TP}{N_{gen}}, \quad R = \frac{TP}{N_{ref}}, \quad S(h, \hat{c}) = \frac{2PR}{P + R}$$
 - 其中：
 - (P)：生成批判在论点空间上的精确率
 - (R)：对人类关键论点的召回率
 - (S)：F1 作为综合质量指标
 - 若检测到模型批判存在「重复刷同一论点以冒充详尽」的模式，则直接将 \$S(h, c^\wedge)\$ 设为 0，防止作弊。

- 3.2 训练目标与损失函数 (Objectives & Loss)
 - A. 综合奖励函数 (Composite Reward Function)
 - 对于策略模型 (GRM)，总奖励 R 由结果奖励和过程奖励组成。引入了结果正则化 (Outcome Regularization)，即如果结果标签预测错误，无论推理如何都不给过程奖励
 - $$R = \begin{cases} -1, & \text{if output format invalid} \\ 0, & \text{if } \hat{l} \neq l \text{ (Outcome Incorrect)} \\ 1 + \lambda R_{process}, & \text{if } \hat{l} = l \text{ (Outcome Correct)} \end{cases}$$
 - 其中 $R_{process} = \mathbb{I}[S(h, \hat{c}) > 0.5]$, λ 为权重系数
 - B. MetaRM 损失函数
 - MetaRM 作为一个回归模型，学习预测上述定义的 R (针对 \mathcal{D}_H 数据) :
 - $\mathcal{L}_{MetaRM}(\phi) = \mathbb{E}_{(q, y_A, y_B, h) \sim \mathcal{D}_H} [(M_\phi(q, y_A, y_B, \hat{c}) - R_{target})^2]$
 - C. 策略优化 (GRPO)
 - 使用 Group Relative Policy Optimization (GRPO) 更新 GRM。优势函数 \hat{A}_i 基于组内奖励的标准差计算:
 - $$\hat{A}_i = \frac{R_i - \bar{R}}{\sigma}$$
 - $$\mathcal{J}(\theta) = \mathbb{E} \left[\min \left(\frac{\pi_\theta}{\pi_{old}} \hat{A}_i, \text{clip}(\dots) \hat{A}_i \right) - \beta \mathbb{D}_{KL} \right]$$
 - (14)(14)(14)(14)。注意：论文采用了 Online 更新策略，在每一轮迭代中，先更新 MetaRM，再用 MetaRM 给无标签数据打分，最后更新 Policy
-

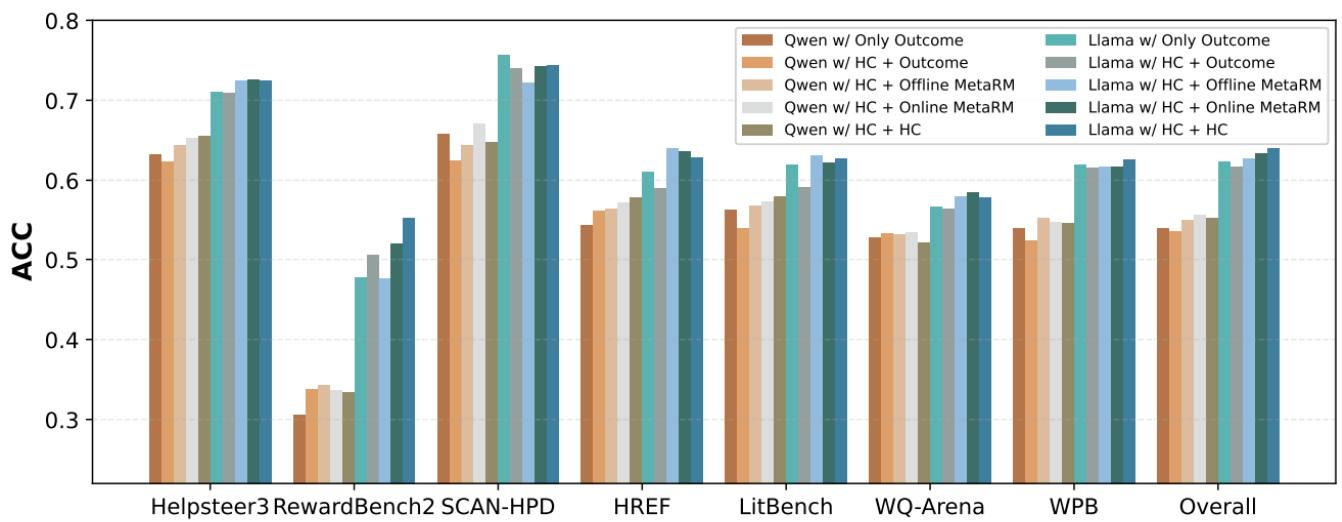


图 5: 多种基准下离线与在线 MetaRM 变体的比较。

表 2: 多基准性能对比。

Model	HelpSteer3	Reward Bench V2	SCAN-HPD	HREF	LitBench	WQ_Arena	WPB	Overall
Base Models (LLM-as-a-Judge)								
gpt-5-2025-08-07	0.8245	0.8344	0.8147	0.7595	0.7617	0.6304	0.5939	0.7456
o3-2025-04-16	0.8315	0.8009	0.8147	0.7058	0.7202	0.6432	0.6234	0.7342
gemini-2.5-pro	0.8207	0.7117	0.8288	0.6934	0.7181	0.6529	0.6367	0.7232
claude-3-7-sonnet-20250219	0.8042	0.7714	0.7524	0.7384	0.7631	0.6300	0.5967	0.7223
deepseek-r1-0528	0.7827	0.7292	0.8227	0.6496	0.7000	0.6348	0.6183	0.7053
qwen-plus-latest	0.8038	0.7717	0.8048	0.6488	0.6835	0.5961	0.6278	0.7052
qwen3-max	0.8102	0.7169	0.7865	0.6219	0.6859	0.6062	0.6133	0.6916
gpt-4o-latest	0.7896	0.6478	0.7572	0.5777	0.6319	0.5799	0.5883	0.6532
deepseek-v3	0.7522	0.5436	0.7636	0.5678	0.6536	0.5848	0.6122	0.6397
DeepSeek-R1-Distill-Qwen-7B	0.5999	0.3052	0.6015	0.5487	0.5418	0.5365	0.5161	0.5214
R1-Distill-Llama-8B	0.6400	0.4095	0.6592	0.5899	0.5822	0.5650	0.5841	0.5757
R1-Distill-Qwen-32B	0.7376	0.6154	0.7492	0.6967	0.6206	0.6115	0.6389	0.6671
Scalar Reward Models								
ArmoRM-Llama3-8B-v0.1	0.7640	0.6897	0.6628	0.7322	0.7157	0.5855	0.4495	0.6571
URM-LLaMa-3.1-8B	0.8012	0.8004	0.7013	0.7347	0.6302	0.5814	0.4884	0.6768
Skywork-Reward-Llama-3.1-8B-v0.2	0.7950	0.7907	0.7205	0.7347	0.6593	0.5743	0.5322	0.6867
INF-ORM-Llama3.1-70B	0.8075	0.8066	0.7684	0.7364	0.7141	0.5978	0.5967	0.7182
Specialized Generative Reward Models								
RM-R1-Qwen-7B	0.6499	0.4679	0.6438	0.6504	0.5750	0.5181	0.5261	0.5759
RRM-Qwen-7B	0.6794	0.5082	0.6645	0.6777	0.5383	0.5421	0.5161	0.5895
RRM-Qwen-32B	0.7942	0.7340	0.7604	0.7273	0.6746	0.5875	0.6283	0.7009
RM-R1-Qwen-32B	0.7818	0.7260	0.7795	0.7099	0.6934	0.5988	0.6367	0.7037
Our Generative Reward Models								
RM-NLHF-Qwen-7B	0.7381	0.5757	0.6822	0.6926	0.6583	0.5416	0.5521	0.6481
RM-NLHF-Qwen-32B	0.8315	0.7867	0.7888	0.7165	0.7492	0.6161	0.6183	0.7296

四、实验设计与评估 (Experiments)

- 4.1 实验设置

- 数据集 (Datasets)

训练与评估主要基于多个偏好与评估基准，论文提到的包括：

- HelpSteer3：偏好与反馈数据集，用于训练与验证 GRM 的奖励质量和批判质量。
- RewardBench V2
- SCAN-HPD
- HREF、LitBench、WQ-Arena、WPB 等，用于更广泛场景下检验 GRM 作为评判器的泛化能力。

- 下游任务评测包括：

- MATH500：数学问题求解质量评估。
- HumanEval+：代码生成任务。
- Arena-Hard-V2.0：对话与复杂任务评估。

- 训练集规模：

- 约 164K 个人工标注的 pairwise 偏好样本，用于 GRPO 训练（其中一部分含人类批判 (h)，一部分仅有 outcome）。

- **基座模型 (Base Models)**

- Qwen 系列（如 Qwen-7B）
- Llama 系列（如 Llama-7B）
- 另有部分小规模模型（用于 MetaRM 或对比）

- **对比基线 (Baselines)**

- **Zero-shot LLM-as-a-Judge**: 直接用通用大模型（如 gpt 类）做评判。
- **Scalar Reward Models**: 传统只输出一个分数的 RM。
- **GRM w/ Outcome-only**: 当前主流 RLVR 方法，只使用结果奖励训练的 GRM（如 RM-R1、RRM 等）。
- **Naive 组合策略**: 对有批判数据用过程+结果，对无批判数据只用结果奖励（证明这种 naive 混合是灾难性的）。

- **4.2 评估指标 (Metrics)**

- **Outcome Accuracy**: 偏好标签预测准确率，即模型 choice 与 ground truth 一致的比例。
- **Critique F1 / Precision / Recall**: 批判与人类批判在「核心论点」或「所有论点」上的 F1、精确率、召回率（由外部 LLM 计算）。
- **MetaRM Accuracy**: MetaRM 分数与真实「F1-based 奖励」的一致性。
- **下游任务指标**:
 - MATH500: 题目正确率。
 - HumanEval+: 代码通过率。
 - Arena-Hard-V2.0: 胜率或评分提升等。

五、实验结果与分析 (Results & Analysis)

- **5.1 核心性能 (Performance)**

- 在多个偏好评估基准（HelpSteer3, RewardBench2, SCAN-HPD, HREF, LitBench, WQ-Arena, WPB）上：
 - **RM-NLHF (全人类批判 + 结果奖励) 显著优于 Outcome-only GRM**。
 - Online MetaRM 在只使用少量人类批判标注的情况下，整体性能接近全批判监督，并普遍优于 Offline MetaRM。
- 在下游任务（使用 GRM 做 Best-of-N 或 Feedback-Edit）上：

- RM-NLHF 的 GRM 作为评判器 / 反馈器时，在 MATH500、HumanEval+、Arena-Hard-V2.0 上均优于 **Outcome-only Baseline**，说明其学到的「评估能力」可迁移至复杂任务的推理与生成改写。

表 4: 下游任务的评估结果。

Method	MATH500	HumanEval+	Arena-Hard-V2.0
Base Model			
DeepSeek-Distilled-Qwen-7B	62.92%	77.13%	3.39%
Best-of-N (BoN)			
Outcome-only (BoN@2)	63.65%	76.30%	3.69%
RM-NLHF (BoN@2)	64.90%	76.95%	3.56%
Outcome-only (BoN@4)	65.45%	75.77%	3.93%
RM-NLHF (BoN@4)	66.80%	81.04%	3.85%
Outcome-only (BoN@8)	65.99%	75.00%	4.30%
RM-NLHF (BoN@8)	67.60%	85.98%	4.64%
Feedback-Edit			
Outcome-only	67.01%	82.32%	6.55%
RM-NLHF	68.40%	87.20%	7.03%

- 语言层面的分析：
 - RM-NLHF 生成的批判中，独特词汇数从基线的 1,633 提升到 4,436，说明其批判更具体、多样，而不是模板化套话。
 - 高频词对比显示：RM-NLHF 更常使用 “incorrect” “unusable” “unrelated” 等诊断性词汇，而 Baseline 偏好 “comprehensive” “clear” “helpful” 等泛泛赞美。
- 结论：加入过程奖励后，模型从“会说场面话、套模板”变成“会指出真正关键问题”，这直接体现在 outcome accuracy 与下游任务性能的全面提升上。
- 5.2 消融实验 (Ablation Studies)
- 论文对不同设计组件进行了系统消融：
 - 仅 outcome 奖励 vs 加入过程奖励：
 - 仅 outcome 奖励的 GRM 虽然在部分任务有不错的准确率，但批判 F1 显著偏低，且语言分析显示多为浅层模板。
 - 加入过程奖励后，Outcome 准确率和 Critique F1 同时提升，证明过程监督并未损害结果准确性，反而提升了整体评估质量。
 - Naive 组合 vs MetaRM 组合：

- 「对有批判数据使用过程+结果，对无批判仅用结果」这种 naive 策略被证明是灾难性的，性能甚至低于纯 outcome-only 模型。
 - 分析认为这是因为：不同数据子集上的奖励形式不一致，导致模型收到冲突的梯度信号，训练不稳定。
 - 使用 MetaRM 将所有数据统一为「复合奖励形式」，恢复了奖励信号的一致性，性能大幅提升。
- Offline MetaRM vs Online MetaRM：
 - Offline MetaRM 仅在初始策略分布上训练一次，然后固定不再更新；实验表明其性能会随着 GRM 继续 RL 训练而持续退化（因为分布偏移）。
 - Online MetaRM 在每轮 RL 迭代前都用最新 GRM 生成的批判做再训练，其 **MetaRM Accuracy** 始终保持较高水平，对应的 GRM 性能也稳定提升。
- **关键结论：**
 - 过程奖励本身是有效的；
 - 奖励信号的“形式一致性”和“在线自适应”对于长期 RL 优化至关重要。
- **5.3 效率与扩展性 (Efficiency & Scaling)**
 - 训练开销：
 - 相比纯标量 RM 或 outcome-only GRM，RM-NLHF 需要：
 - 额外调用外部 LLM 计算 $S(h, c^A)$ （仅在带人类批判的数据上）。
 - 训练并在线更新一套 MetaRM。
 - 标注成本：
 - 只需对训练数据中一小部分样本提供人类批判（其余只有 outcome 标签），通过 Online MetaRM 即可接近「全量批判监督」效果。
 - 相比完全依赖人类过程标注，这种方案将标注成本降低到了实践上可接受的量级。

六、Prompt 与数据示例补充 (Prompts & Data Examples)

- **6.1 数据构建细节**
 - **标注内容：**
 - 每条样本包含：
 - Query (q)
 - Response A / Response B
 - 偏好标签 (l)（哪一个更好）
 - 可选的人类批判 (h)：解释为什么某一响应更好，指出对方的错误/优点

- 人类批判要求：
 - 聚焦核心论点，例如：
 - 事实是否正确
 - 语气是否恰当
 - 是否回答了用户问题，是否有毒性等
 - 避免冗长但无信息量的重复。
- 构造 MetaRM 初始训练集（Cold-start）：
 - 对于含人类批判的样本，采样多个 GRM 输出，计算各自的复合奖励 (R)，形成 (输入, Rtarget) (输入, Rtarget) 对，用于初始化训练 MetaRM。

1. 生成式奖励模型模版 (Template of Generative Reward Models)

用途：用于规范 GRM 的输出格式，使其生成结构化的评论（Critique）和最终选择。

来源：Figure 9

代码块

```

1 Please act as an impartial judge and evaluate the quality of the responses
2 provided by two AI Chatbots to the Client question displayed below.
3
4 [Client Question]
5 {conv_his}
6
7 [The Start of Chatbot A's Response]
8 {response_A}
9 [The End of Chatbot A's Response]
10
11 [The Start of Chatbot B's Response]
12 {response_B}
13 [The End of Chatbot B's Response]
14
15 Output your final verdict by strictly following this format:
16
17 <critics>
18 [Provide a brief summary of your reasoning for the choice]
19 </critics>
20 <choice>
21 [[A]]
22 </choice>
23 Note: Use [[A]] if A is better, or [[B]] if B is better.
```

2. 基于评论的纯编辑指令 (Edit-only Refinement Instruction based on Critiques)

用途：在下游任务评估中，用于让编辑模型（Edit-Model）仅根据 GRM 的评论来修改回复，以验证评论的有效性。

来源：Figure 10

代码块

```
1 [SYSTEM RULE: EDIT-ONLY MODE ENGAGED]
2 You are a text-processing bot. You are FORBIDDEN from answering the question.
3 Your only function is to apply edits.
4
5 PRIMARY DIRECTIVE: Follow the <critique>. Nothing else.
6
7 1. THE EXCEPTION RULE: IF the <critique> states something is factually or
8 mathematically wrong, you are authorized to fix ONLY THAT SINGLE PIECE OF
9 INFORMATION. Do not explain. Do not add context. Just replace the wrong data
10 with the correct data.
11
12 2. THE DEFAULT RULE: For everything else, if the <critique> does not
13 explicitly order a change, you MUST NOT change it. Do not fix other errors. Do
14 not improve style. Do not add information.
15
16 3. THE RE-CHECK RULE: Before you respond, you must check again whether the
17 <critique> contain the content you modify.
18
19 Failure to follow these rules means you fail the task.
20
21 [Client Question]
22 {conv_his}
23
24 [The Start of Chatbot A's Response]
25 {response_A}
26 [The End of Chatbot A's Response]
27
28 [The Start of Chatbot B's Response]
29 {response_B}
30 [The End of Chatbot B's Response]
31
32 [The Start of Critique]
33 {critique}
34 [The End of Critique]
```

3. 基于核心论点计算相似度 (Calculation of Similarity w/ Core Human Critiques)

用途：用于计算模型生成的评论与人类评论在“核心论点”上的 F1 分数，这是论文推荐的最佳过程奖励计算方式。

来源：Figure 11

代码块

```
1 I will provide you with a generated evaluation content and a reference
  evaluation content. Your task is to analyze the similarity between the
  <Generated Evaluation Content> and the <Reference Evaluation Content> by
  calculating F1 scores based on their key arguments.
2
3 Core Principle: Focus exclusively on "Key Arguments" – decisive reasons that
  are powerful enough to justify the final choice on their own. Identify these
  core justifications, not minor points.
4
5 ## Part 1: First check
6 First check if the generated critique repeats the same point across multiple
  times. If yes, directly output without conducting part 2:
7 <thinking>
8 Put here how the generated critique repeats points.
9 </thinking>
10 <scores>
11 <critique_f1>0</critique_f1>
12 <critique_precision>0</critique_precision>
13 <critique_recall>0</critique_recall>
14 </scores>
15 ## Part 2: Steps for F1 Score Calculation
16 1. Count Reference Key Arguments (N_ref):
17 Check if the reference identifies a fatal error (critical factual error,
  harmful statement, or fundamental misunderstanding).
18 - If yes: Only this fatal error counts. Set N_ref = 1.
19 - If no: Count all unique Key Arguments (decisive reasons that could justify
  the choice by themselves). Set N_ref to this count.
20
21 2. Count Generated Key Arguments (N_gen):
22 - Identify all unique Key Arguments in the generated evaluation.
23 - Set N_gen to this count.
24
25 3. Count True Positives (TP):
26 - Initialize TP = 0.
27 - For each reference key argument, search for a match in generated key
  arguments.
```

```

28 - Matching Rule: Both semantic meaning and stance (which response and
positive/negative) must align.
29 - Example: "Response A is more detailed" only matches with similar praise of
Response A, not Response B.
30 - For fatal errors: Generated must identify the same error in the same
response.
31 - Each generated argument can only match once.
32 - Increment TP by 1 for each valid match.
33
34 4. Calculate Scores:
35 - Precision_critique: TP / N_gen (0 if N_gen = 0)
36 - Recall_critique: TP / N_ref (0 if N_ref = 0)
37 - CritiqueScore: 2 * (Precision * Recall) / (Precision + Recall) (0 if sum = 0)
38
39 Output Format (rounded to 4 decimal places):
40 <thinking>
41 Put the thinking process here.
42 </thinking>
43 <scores>
44 <critique_f1>CritiqueScore</critique_f1>
45 <critique_precision>Precision_critique</critique_precision>
46 <critique_recall>Recall_critique</critique_recall>
47 </scores>
48 <Generated Evaluation Content>
49 {critiques}
50 </Generated Evaluation Content>
51 <Reference Evaluation Content>
52 {reference_critiques}
53 </Reference Evaluation Content>

```

4. 基于所有论点计算相似度 (Calculation of Similarity w/ All Human Critiques)

用途：这是一个对比实验用的 Prompt，它不过滤“核心”论点，而是计算所有论点的相似度（在论文结论中效果不如 Core Human Critiques）。

来源：Figure 12

代码块

- I will provide you with a generated evaluation content and a reference evaluation content. Your task is to analyze the similarity between the <Generated Evaluation Content> and the <Reference Evaluation Content> by calculating F1 scores based on their all arguments.

```

3 ## Part 1: First check
4 First check if the generated critique repeats the same point across multiple
times. If yes, directly output without conducting part 2:
5 <thinking>
6 Put here how the generated critique repeats points.
7 </thinking>
8 <scores>
9 <critique_f1>0</critique_f1>
10 <critique_precision>0</critique_precision>
11 <critique_recall>0</critique_recall>
12 </scores>
13 ## Part 2: Steps for F1 Score Calculation
14 1. Count Reference All Arguments (N_ref):
15 - Check if the reference identifies a fatal error (critical factual error,
harmful statement, or fundamental misunderstanding).
16 -- If yes: Only this fatal error counts. Set N_ref = 1.
17 -- If no: Count all unique Arguments (decisive reasons that could justify the
choice by themselves). Set N_ref to this count.
18
19 2. Count Generated All Arguments (N_gen):
20 - Identify all unique Arguments in the generated evaluation.
21 - Set N_gen to this count.
22
23 3. Count True Positives (TP):
24 - Initialize TP = 0.
25 - For each reference argument, search for a match in generated arguments.
26 - Matching Rule: Both semantic meaning and stance (which response and
positive/negative) must align.
27 -- Example: "Response A is more detailed" only matches with similar praise of
Response A, not Response B.
28 -- For fatal errors: Generated must identify the same error in the same
response.
29 - Each generated argument can only match once.
30 - Increment TP by 1 for each valid match.
31
32 4. Calculate Scores:
33 - Precision_critique: TP / N_gen (0 if N_gen = 0)
34 - Recall_critique: TP / N_ref (0 if N_ref = 0)
35 - CritiqueScore: 2 * (Precision * Recall) / (Precision + Recall) (0 if sum = 0)
36
37 Output Format (rounded to 4 decimal places):
38 <thinking>
39 Put the thinking process here.
40 </thinking>
41 <scores>
42 <critique_f1>CritiqueScore</critique_f1>
43 <critique_precision>Precision_critique</critique_precision>

```

```
44 <critique_recall>Recall_critique</critique_recall>
45 </scores>
46 <Generated Evaluation Content>
47 {critiques}
48 </Generated Evaluation Content>
49 <Reference Evaluation Content>
50 {reference_critiques}
51 </Reference Evaluation Content>
```

5. LLM 作为元裁判 (LLM-as-a-Meta-Judge)

用途：这是另一种过程奖励设计的尝试（直接让 LLM 评估评论质量），在论文中作为对比基线，效果不如基于相似度的方法。

来源：Figure 13

代码块

```
1 You are an expert evaluator tasked with assessing the quality of critiques
  comparing two responses. You will be given:
2 1. A conversation history
3 2. Response A
4 3. Response B
5 4. One or more critiques comparing Response A and Response B
6
7 Your task is to evaluate whether the critique(s) are accurate and correct
  based on the actual content of the responses. Assign a score between 0 and 1,
  where higher scores indicate more accurate critiques.
8
9 You must provide your response in the following XML format:
10 <thinking>
11 Put your detailed analysis here. Examine the critique against the actual
  responses and explain your reasoning for the score.
12 </thinking>
13 <scores>
14 <critique_f1>Score</critique_f1>
15 <critique_precision>Score</critique_precision>
16 <critique_recall>Score</critique_recall>
17 </scores>
18 Note: Assign the same score to all three metrics (critique_f1,
  critique_precision, and critique_recall).
19
20 <Conversation History>
21 {conv_his}
```

```
22 </Conversation History>
23 <Response A>
24 {response_A}
25 </Response A>
26 <Response B>
27 {response_B}
28 </Response B>
29 <Critiques>
30 {critiques}
31 </Critiques>
```

下载数据

代码块

```
1 export HF_ENDPOINT=https://hf-mirror.com
2 python -c "from huggingface_hub import snapshot_download;
snapshot_download(repo_id='nvidia/HelpSteer3', repo_type='dataset',
local_dir='./HelpSteer3_CLI', endpoint='https://hf-mirror.com')"
```