

当验证器只标注部分样本时： GRPO 中冻结、重新校准与从头训练的评判输出头

DC-Jev-RL 项目，第三部分 B

2026 年 10 月

摘要

当验证器只能给 GRPO run 的部分样本打标签时，可以由评判模型为其余样本补全奖励。我们沿用第二部分的设置，在 MATH 上用 Qwen3.5-4B-Base 研究这一情形：验证器每组 8 个解答只标注 2 个，另外 6 个或者被丢掉 (`rlvr_sub`)，或者用冻结的 Kev-4B 的判断打分 (`sub_kev`)，或者用指针头在迄今收集的标签上在线重新拟合后的 Kev-4B 打分 (`sub_kev_recal`)，或者用一个形状相同、在相同特征和标签上从头训练的输出头打分 (`sub_probe`)；每个条件两个种子，各 200 步。冻结的评判模型对未标注解答排序良好 (AUROC 0.91)，但过于慷慨，它的两个策略都学会了在答完之后重复自己，而评判模型会为此付费；它们在 MATH-500 上最终相对丢掉这些解答为 -2.3 $[-4.4, -0.1]$ 个百分点。重新拟合输出头修正了它的校准（期望校准误差 0.10 对 0.19），但两个策略中仍有一个重复自己。从头训练的输出头在 MATH-500 上最终相对丢掉为 $+4.1$ $[1.6, 6.6]$ 个百分点，在 AIME24 上高 $+12.2$ ，也高于满标注的基线。它的两个策略都改成了以纯文本的非正式草稿开头；在本项目的其他 GRPO run 中，这一切换无论奖励为何都标志着最好的 run，因此两个种子无法说明增益中有多少属于探针。代码、日志、每个训练解答以及本文的每个数字都在仓库中。

1 引言

RLVR 假设每个采样出的解答都能被检验。当检验代价高昂时，例如需要人工评分、长时间的测试套件或缓慢的仿真器，一次训练可能只负担得起部分样本的标签，其余的可以由评判模型补上。然而，一个昨天在样本上判断正确的评判模型，今天可能就判断错了，因为它所评判的策略在不断变化。

本文用 GRPO 在 MATH 上研究这一设定，沿用第二部分的设置，其中验证器只给每组 8 个解答中的 2 个打标签。我们比较利用另外 6 个解答的四种方式：丢掉它们；用冻结的 Kev-4B 的判断作为奖励；用在线重新拟合输出头（基于迄今收集的标签）之后的 Kev-4B 作为奖励；或者用一个形状相同、在同样标签上从头训练的输出头作为奖励。由于验证器实际上是可用的，我们还能逐步测量每种补全奖励在它所打分的解答上有多准确。

2 相关工作

半监督奖励学习和主动标注可以减少 RLHF 中人工标签的数量 [4]。训练一次后就冻结的奖励模型，会随着策略漂移而被利用 [1]。在训练中用新标签重新拟合奖励模型是 RLHF 中常用的补救办法，但要消耗标签。在新数据上重新校准分类器的输出层是应对分布偏移的标准做法 [2]，这里把它在线地用在评判模型的指针头上。GRPO 和我们的更新方式沿用第二部分 [3, 5]。

3 设定：只给部分解答打标签的验证器

第二部分中每个解答都得到验证器的判定。这里每组解答中只有比例 $f = 0.25$ 被打标签，即 8 个中随机选 2 个。打标签的比例模拟运行代价高昂的验证器，例如人工评分或缓慢的测试。问题是如何利用另外 75% 未打标签的解答：

- **rlvr_sub**: 把它们排除在外。优势只来自每组中两个打了标签的解答，其余解答没有梯度。
- **sub_kev**: 用发布版、冻结的 Kev-4B 的 P_c (“所给解答的最终答案是否正确?") 作为它们的奖励。
- **sub_kev_recal**: 用 Kev-4B 指针头的一个副本给出的 P_c 作为奖励，该副本在迄今见过的所有已标注解答上在线重新拟合。
- **sub_probe**: 用一个形状相同、基于相同特征、在相同标签上从头训练的输出头作为奖励。它区分了“Kev-4B 发布版输出头的贡献”与“任何拟合这些标签的小输出头都能做到的”。

已标注的解答总是得到验证器的奖励。上限参照是第二部分的 **rlvr**，其中每个解答都有标签。其余一切都沿用第二部分的设置：相同的模型、数据、学习率、批大小、步数和评估（第二部分第 2 节）。仅为记录之用，验证器也检查每个未标注的解答，从而测量每种补全奖励在它所用于的解答上有多准确。

4 在线重新校准

Kev-4B 的指针头把骨干的隐藏状态变成各选项的 logit z ，其 $p(\text{yes})$ 为 $\text{softmax}(z/T)_{\text{yes}}$ ， T 为发布版的温度。**sub_kev_recal** 保持骨干和 Kev-4B 的适配器冻结，每一步之后重新拟合输出头的一个 float32 副本 θ 。损失为 $p(\text{yes})$ 对验证器标签的二元交叉熵，加上把 θ 锚定到发布版输出头 θ_0 的项 $\alpha \|\theta - \theta_0\|^2$ ($\alpha = 10^{-3}$)。拟合在最近 8192 个已标注解答的缓冲区上做 50 步 Adam (学习率 10^{-4} ，批大小 256)。特征（输出头的输入）在 Kev-4B 给解答打分时只计算一次，所以重新拟合不需要再跑骨干。骨干上的发布版输出头始终不变。**sub_probe** 采用同样的流程，但从输出头投影层的全新初始化开始， $T = 1$ ，不加锚定项。

之所以预期重新校准会起作用，是因为策略的解答在训练中会漂移：变长或变短，风格改变，于是一个在自身训练分布上校准过的评判模型，在这些解答上可能失去校准。已标注的解答直接测量了这种漂移，而在线重新拟合是跟上它的最便宜的办法。

condition	seeds	MATH-500	GSM8K	AMC23	AIME24	AIME25
base model	–	76.5	87.9	50.6	10.0	10.9
rlvr	3	82.5 $\pm$ 0.6	93.0 $\pm$ 0.3	60.9 $\pm$ 2.4	21.5 $\pm$ 0.4	16.7 $\pm$ 2.1
rlvr_sub	2	81.8 $\pm$ 1.7	92.4 $\pm$ 0.4	60.8 $\pm$ 1.1	18.3 $\pm$ 1.8	15.4 $\pm$ 0.3
sub_kev	2	79.5 $\pm$ 1.3	91.2 $\pm$ 1.3	56.1 $\pm$ 3.5	15.9 $\pm$ 1.6	14.5 $\pm$ 1.6
sub_kev_recal	2	81.2 $\pm$ 2.5	92.3 $\pm$ 0.9	59.4 $\pm$ 1.8	21.1 $\pm$ 4.6	14.7 $\pm$ 3.4
sub_probe	2	85.9 $\pm$ 1.8	93.7 $\pm$ 0.5	67.7 $\pm$ 2.3	30.5 $\pm$ 0.1	23.9 $\pm$ 1.3

表 1: 最终策略: 准确率 (%), 种子均值 $\pm$ 种子间标准差。MATH-500 和 GSM8K: 贪心解码。AMC23、AIME24 和 AIME25: 温度 0.6 下每题 16 个样本的平均。rlvr: 每个解答都打标签 (第二部分); 其他条件每组 8 个解答只标注 2 个。

condition	seed	MATH-500	GSM8K	AMC23	AIME24	AIME25
rlvr	1	82.8	93.0	63.6	21.0	18.8
	2	83.0	93.3	59.8	21.9	16.9
	3	81.8	92.6	59.2	21.7	14.6
rlvr_sub	1	80.6	92.6	61.6	19.6	15.6
	2	83.0	92.1	60.0	17.1	15.2
sub_kev	1	78.6	90.3	53.6	14.8	13.3
	2	80.4	92.1	58.6	17.1	15.6
sub_kev_recal	1	79.4	91.7	58.1	17.9	12.3
	2	83.0	92.9	60.6	24.4	17.1
sub_probe	1	87.2	93.3	69.4	30.6	22.9
	2	84.6	94.1	66.1	30.4	24.8

表 2: 每个 run 的最终策略: 准确率 (%), 口径同表 1。

5 结果

每个条件跑了 2 个种子, 各 200 步; 第二部分给每个解答都打标签的 rlvr 是满标注预算的参照 (3 个种子)。表 1 给出最终策略的准确率, 表 2 给出每个 run 的准确率, 表 3 给出相对 rlvr_sub (直接丢掉未标注解答的条件) 的配对差。表 4 给出每种补全奖励在它所打分的解答上有多准, 表 5 给出每个最终策略写了什么, 图 1 给出训练曲线。区间接按题目重采样, 不包含种子之间的差异; 在本项目的 GRPO run 中, 种子之间的差异可能大于条件之间的差异 (第二部分 §8)。

丢掉四分之三的标签, 代价不大。 每组只用两个已标注解答时, rlvr_sub 在 MATH-500 上达到 81.8% (两个种子为 80.6% 和 83.0%)。八个都标注 (rlvr) 多出 +0.7 [−1.1, 2.7] 个百分点, AIME24 上多出 +3.2 [1.0, 5.7]。补全奖励能挽回的就是这么多——前提是它不教给策略别的东西。

minus rlvr_sub	MATH-500	GSM8K	AMC23	AIME24	AIME25
rlvr	+0.7 [-1.1, 2.7]	+0.6 [-0.2, 1.4]	+0.1 [-3.2, 3.3]	+3.2 [1.0, 5.7]	+1.3 [-0.7, 3.3]
sub_kev	-2.3 [-4.4, -0.1]	-1.2 [-2.1, -0.2]	-4.7 [-8.8, -0.3]	-2.4 [-5.3, 0.1]	-0.9 [-4.2, 2.2]
sub_kev_recal	-0.6 [-2.8, 1.6]	+0.0 [-0.9, 0.8]	-1.4 [-5.5, 3.2]	+2.8 [0.8, 5.2]	-0.7 [-3.1, 1.4]
sub_probe	+4.1 [1.6, 6.6]	+1.3 [0.4, 2.3]	+7.0 [2.7, 11.2]	+12.2 [5.2, 20.3]	+8.4 [4.0, 13.5]

表 3: 相对 rlvr_sub 的配对差 (百分点): 逐题取各条件的种子均值之差, 再对题目取平均, 附按题目自助法得到的 95% 区间。

condition	seeds	fill-in AUROC			frozen Kev-4B		mean	
		steps 0–19	first half	second half	AUROC	ECE	fill-in	label
sub_kev	2	0.89	0.92	0.90	0.91	0.19	0.75	0.60
sub_kev_recal	2	0.92	0.94	0.92	0.90	0.10	0.63	0.62
sub_probe	2	0.93	0.95	0.92	0.91	0.11	0.66	0.65

表 4: 补全奖励在未标注解答上对照验证器被隐去的标签 (逐步计算, 再对步数和种子取平均): 前 20 步与 run 前后两半的 AUROC、冻结 Kev-4B 在同样解答上的 AUROC、期望校准误差, 以及平均补全奖励与其正确解答所占比例的对照。

冻结的评判模型让结果更差。 用发布版 Kev-4B 的 P_c 给未标注解答打分 (sub_kev), 两个种子都比直接丢掉它们差 (MATH-500 上 78.6% 和 80.4%): MATH-500 上差 -2.3 $[-4.4, -0.1]$ 个百分点, AMC23 上差 -4.7 $[-8.8, -0.3]$, AIME24 上差 -2.4 $[-5.3, 0.1]$ 。然而评判模型对未标注解答的排序一直很好: 它对被隐去标签的 AUROC 在整个 run 上为 0.91 (表 4)。问题在于它太慷慨: 前 20 步它在未标注解答上的平均 P_c 为 0.65, 而其中只有 47.6% 正确; 最后 20 步为 0.79, 正确的有 63.3%。哪些解答被标注是随机的, 所以一个常数偏移会让每种行为的期望奖励提高同样的量, 组基线在期望上会把它消掉。但一个为某种行为付出超出其正确性所值的回报的评判模型, 会改变策略学到的东西。两个策略都学会了重复自己: 它们的 MATH-500 解答中分别有 48.6% 和 24.4% 的行与前面某行相同, rlvr_sub 为 1.3% 和 1.1%; 种子 1 有 69.0% 的解答写到 2048 词元的上限 (表 5)。在第一个答案之后, 种子 1 一遍又一遍地重述结论, 常常一直写到长度上限; 种子 2 则把题目再解一遍。两者都不改变验证器读到的答案, 所以已标注的解答不会惩罚这种行为, 而在未标注的解答上评判模型会为它付费 (§5.2)。

重新校准修正了校准, 没有修正偏好。 在迄今收集的标签上重新拟合 Kev-4B 的输出头 (sub_kev_recal), 在它被拟合的那些数上如预期般起作用: 它在未标注解答上的校准误差为

condition	seed	tokens	truncated	repeated lines		several answers	draft opening	Kev-4B P_c	
				share	after answer			right	wrong
rlvr	1	541	2%	1%	0%	2%	1%	0.84	0.60
	2	555	4%	1%	0%	6%	0%	0.85	0.59
	3	909	8%	28%	90%	95%	1%	0.88	0.64
rlvr_sub	1	630	6%	1%	0%	3%	1%	0.84	0.54
	2	588	3%	1%	0%	3%	2%	0.83	0.56
sub_kev	1	1723	69%	49%	49%	99%	1%	0.91	0.72
	2	1072	12%	24%	85%	97%	2%	0.91	0.71
sub_kev_recal	1	904	3%	22%	85%	98%	7%	0.90	0.71
	2	538	3%	1%	0%	11%	1%	0.85	0.62
sub_probe	1	546	6%	0%	0%	75%	90%	0.90	0.61
	2	694	6%	0%	0%	71%	90%	0.91	0.66

表 5: 每个最终策略在 MATH-500 上写了什么 (贪心): 解答词元数、在 2048 词元处被截断的比例、(至少 20 个字符的) 行与同一解答前面某行相同的比例、在第一个框出答案之后重复这样一行的解答比例、含有多个框出答案的解答比例 (验证器读最后一个)、以非正式草稿开头的解答比例, 以及 Kev-4B 对正确与错误答案的 P_c 。

0.10 (冻结输出头为 0.19), 平均补全奖励 0.63 与其中 61.6% 的正确率相符, 在同样的解答上排序也略好于冻结输出头 (AUROC 0.93 对 0.90)。种子 1 仍然学会了重复自己 (22.1% 的行), MATH-500 最终为 79.4%; 种子 2 没有 (0.8%), 最终为 83.0%。两个种子合起来, 这个条件并不比丢掉未标注解答更好 (MATH-500 上 $-0.6 [-2.8, 1.6]$)。

从头训练的输出头: 两个种子都改成了先写草稿。 探针 (sub_probe) 的排序和校准与重新校准的输出头大致相当 (AUROC 0.93, 校准误差 0.11), 它的两个策略都不重复自己 (分别 0.2% 和 0.4% 的行)。两者最终都远高于本文其他所有 run: MATH-500 上 87.2% 和 84.6%, AIME24 上 30.6% 和 30.4%。两个种子合起来, 在 MATH-500 上比 rlvr_sub 高 +4.1 [1.6, 6.6] 个百分点, 在 AIME24 上高 +12.2 [5.2, 20.3], 也高于满标注的 rlvr (MATH-500 上 82.5%)。两个策略的写法都与众不同: 先用纯文本把题目做一遍, 其中分别有 50% 和 72% 的草稿之后再写格式化的解答 (§5.1)。

评估噪声。 在本比较的 11 个 run 中, 基座模型第 0 步的 MATH-500 准确率在 75.2% 到 78.2% 之间; 一个 run 在第 150 步与第 200 步的评估之间平均变化 1.5 个百分点。每个 run 的最终权重在 MATH-500 上被贪心评估两次: 一次是第 200 步的定期评估, 一次是最终评估。两次结果平均相差 1.0 个百分点, 最多相差 2.2 (11 个 run)。

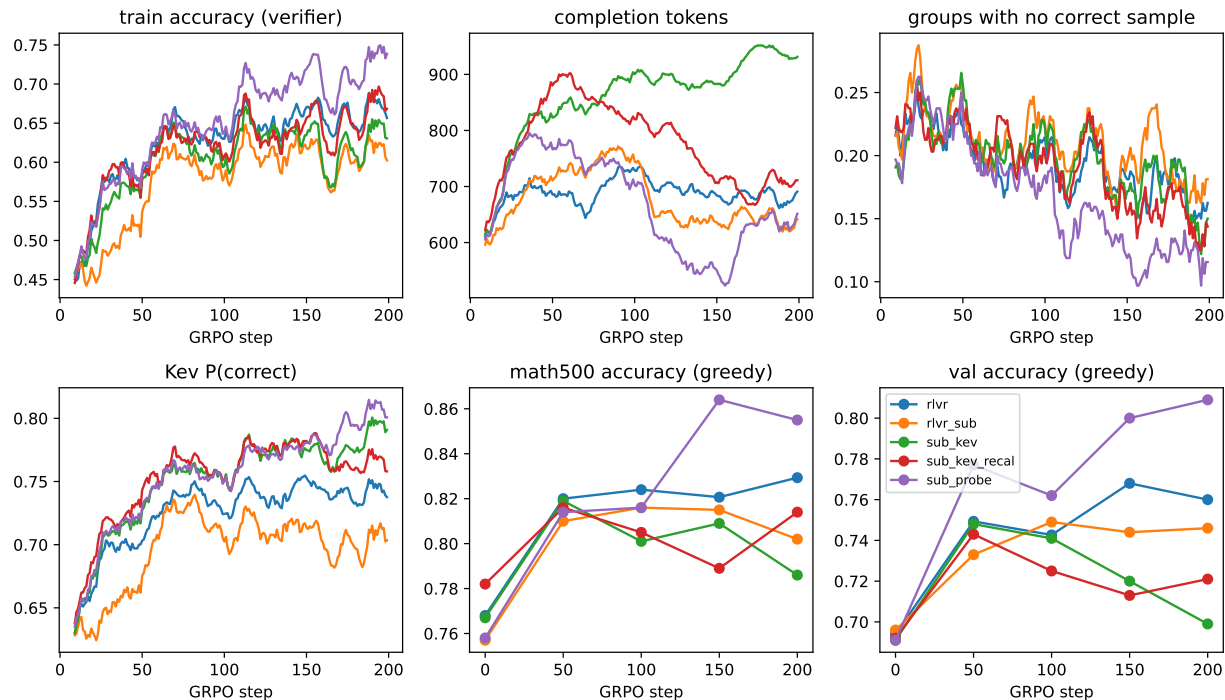


图 1: 训练曲线 (种子均值): 全部训练解答 (无论是否标注) 的验证器准确率、解答词元数、没有任何正确解答的组所占比、训练解答上 Kev-4B 的 P_c (10 步滑动平均), 以及每 50 步在 MATH-500 和验证集上的贪心准确率。

5.1 先写草稿

第二部分发现, 有些 GRPO run 会突然切换到这样一种风格: 解答先用纯文本、不带 LaTeX 或 markdown 写一份非正式草稿, 给出答案, 之后常常再写格式化的解答; 而发生切换的 run 正是最好的那些。我们用以这种草稿开头的解答所占比例来衡量这一风格 (前 300 个字符中没有数学定界符、markdown 标题或粗体; 第二部分 §8.3, 在读过解答之后定义)。基座模型的 MATH-500 解答有 11% 到 12% 以这种方式开头。

只有探针的策略发生了切换。 除一个 run 外, 每个 run 在前 50 步都压低了草稿的比例 (表 6)。例外是 sub_probe 种子 1: 它在第 50 步有 28% 的 MATH-500 解答以草稿开头, 第 100 步为 91%。sub_probe 种子 2 的这一比例在第 50 步降到 5%, 随后上升, 从第 100 步的 9% 升到第 150 步的 90%。标注预算条件的其他六个 run 以及 rlv_r 的三个 run, 在最终评估中都至多为 7%。在本项目的 26 个 GRPO run 中, 发生切换的 6 个分属 5 个不同条件, 正是 AIME24 上最好的 6 个 (30.4% 到 34.6%, 其他 run 至多 24.4%; 第二部分 §8.3)。sub_probe 相对 rlv_r_sub 的增益伴随着这一切换; 只有两个种子, 我们无法把探针的作用与切换的作用分开。

补全奖励为草稿付了多少。 在一组之内, GRPO 偏向奖励更高的解答, 因此当一种风格的解答比同组其他解答得到更多回报时, 这种风格就会扩散。表 7 按验证器的判定比较草稿与同组其他解

condition	seed	MATH-500	AIME24	share of drafts, step					
				0	50	100	150	200	final
rlvr	1	82.8	21.0	12%	1%	0%	1%	1%	1%
	2	83.0	21.9	12%	0%	0%	0%	0%	0%
	3	81.8	21.7	12%	1%	1%	0%	1%	1%
rlvr_sub	1	80.6	19.6	11%	3%	1%	1%	1%	1%
	2	83.0	17.1	12%	3%	2%	1%	2%	2%
sub_kev	1	78.6	14.8	11%	2%	1%	1%	2%	1%
	2	80.4	17.1	12%	2%	1%	1%	1%	2%
sub_kev_recal	1	79.4	17.9	12%	2%	1%	2%	6%	7%
	2	83.0	24.4	12%	1%	2%	3%	1%	1%
sub_probe	1	87.2	30.6	12%	28%	91%	98%	91%	90%
	2	84.6	30.4	12%	5%	9%	90%	89%	90%

表 6: 以非正式草稿开头的 MATH-500 解答所占比例：每次定期评估（贪心，每 50 步）和最终评估，逐个 run 列出，并附最终准确率。

答，表 8 按判定相同的未标注解答上的补全奖励比较二者。第 0–49 步，在全部 11 个 run 中草稿都比同组其他解答更少被判正确（差值在 -15.3 到 -2.4 个百分点之间），这就是它们的比例下降的原因。在 `sub_probe` 的两个 run 中，从第 50 步起，草稿被判正确的频率不低于其他解答（种子 1 为 $+1.5$ 和 $+3.8$ 个百分点，种子 2 为 $+1.4$ 和 $+0.6$ ）；在草稿扩散的那些窗口里，探针给正确草稿的回报高于其他正确解答（种子 1 第 50–99 步高 $+2.1$ 个百分点，种子 2 第 50–149 步高 $+5.3$ 和 $+1.9$ ）。其他补全 run 的情况则不一。在 `sub_kev_recal` 种子 2 中，run 的后半段草稿比其他解答更常被判定正确（ $+1.6$ 和 $+4.8$ 个百分点），重新校准的输出头却给正确草稿更低的回报（ -2.5 和 -3.2 ），草稿也没有扩散。在 `sub_kev_recal` 种子 1 中，输出头在第 100–199 步给正确草稿更高的回报（ $+3.6$ 和 $+3.6$ ），草稿同样没有扩散。几个 run 的几个窗口、事后读出的结果，说明不了是什么让探针的 run 发生了切换；何况第三部分 C 中也有只用验证器奖励就发生切换的 run（第二部分 §8.3）。

5.2 评判模型会为重复付费吗？

GRPO run 结束之后，我们用第二部分的探针（`dcjev/llm/repeat_probe.py`；表 9）检验了 Kev-4B 是否会为 `sub_kev` 和 `sub_kev_recal` 的策略在第一个答案之后所写的内容付费。`sub_kev` 的种子 1 一遍又一遍地重述结论；`sub_kev_recal` 的种子 1 用几乎相同的话把题目再解一遍，就像第二部分中重复的策略那样。我们把种子 1 在第一个答案之后出现重复行的最终 MATH-500 解答截断在第一个答案处，就像策略在那里停下了一样。在第一个答案之前就开始重复的解答，以及第一个答案与最后一个答案不同的解答，不计在内（分别 66 和 44 个）。所有改动都不改变验证器

condition	seed	steps 0–49		steps 50–99		steps 100–149		steps 150–199	
		drafts	Δ right	drafts	Δ right	drafts	Δ right	drafts	Δ right
rlvr	1	7%	−8.5	2%	+4.2	1%	−3.9	2%	+1.8
	2	5%	−15.3	1%	−9.3	0%	−5.0	0%	−2.1
	3	6%	−14.3	1%	−0.7	1%	+3.3	1%	+2.6
rlvr_sub	1	13%	−8.3	3%	−7.2	2%	−5.1	1%	−5.5
	2	11%	−13.7	4%	−3.3	2%	−10.0	4%	+0.2
sub_kev	1	10%	−8.9	4%	−5.9	2%	−5.7	3%	+2.8
	2	12%	−11.0	6%	−3.6	1%	−8.8	1%	−5.0
sub_kev_recal	1	8%	−8.9	3%	−2.9	2%	−4.0	5%	+0.0
	2	11%	−5.7	2%	−0.3	3%	+1.6	2%	+4.8
sub_probe	1	14%	−2.4	52%	+1.5	94%	+3.8	96%	+1.6
	2	10%	−8.3	3%	+1.4	56%	+0.6	88%	+2.8

表 7: 以非正式草稿开头的训练解答, 按 50 步的窗口: 它们所占比例, 以及 Δ right: 在同时含有两类解答的组上取平均的、草稿被验证器判为正确的比例减去同组其他解答的这一比例 (百分点)。

读到的答案。

第一个答案之后的部分让发布版 Kev-4B 的 P_c (也就是 `sub_kev` 的补全奖励) 在正确答案上提高 +0.031 [0.025, 0.038], 在错误答案上提高 +0.049 [0.032, 0.067]。在 `sub_kev_recal` 的解答上, 它提高 +0.059 [0.053, 0.065] 和 +0.110 [0.096, 0.124], 96% 的错误答案分数上升。单纯的重复价值不大: 写进基座策略的解答时, 它让 Kev-4B 的 P_c 在正确答案上提高 +0.012, 在错误答案上变化 −0.002。所以在未标注的解答上, 补全奖励为这些策略在第一个答案之后学会写的内容付了费, 在错误答案上尤其如此——而验证器对这些错误答案本不会给任何回报。探针用发布版输出头打分; `sub_kev_recal` 的输出头在训练中被重新拟合过, 它在 run 结束时自己的偏好这里没有测量。没有任何条件用到的 AR 评判模型, 给带有第一个答案之后部分的 `sub_kev_recal` 解答打更高的分 (正确答案上 +0.034), 却给 `sub_kev` 无休止的重述打更低的分 (正确答案上 −0.018, 错误答案上 −0.074)。

6 讨论

排序好还不够。 冻结的 Kev-4B 对未标注解答的排序很好 (AUROC 0.91), 用它打分却仍然比丢掉这些解答更差。它在正确性之外所付费的两样东西, 后果不同。它的慷慨—— P_c 远高于正确解答所占比例——让每个解答的期望奖励提高大致相同的量, 会被组基线消掉。它对答完之后还继续写的解答的偏好则不会: 这样的文字不改变验证器的标签, 所以已标注的解答永远不会惩罚它, 而评判模型在未标注的解答上为它付费 (§5.2)。 `sub_kev` 的两个策略都学会了重复自己。评价一个补全奖励, 要看它在判定相同的解答之间为什么付费; 对判定的 AUROC 衡量不了这一点。

condition	seed	steps 0–49		steps 50–99		steps 100–149		steps 150–199	
		right	wrong	right	wrong	right	wrong	right	wrong
sub_kev	1	−2.5	−1.6	+0.5	−0.7	+0.0	−3.2	+0.0	−2.5
	2	+2.2	−1.4	−1.1	−2.2	−1.8	−3.5	−2.4	−1.3
sub_kev_recal	1	−5.4	−10.4	−0.5	+5.1	+3.6	−4.6	+3.6	+6.3
	2	−2.9	−6.0	−0.3	−0.5	−2.5	−4.6	−3.2	−1.3
sub_probe	1	+0.5	−7.0	+2.1	−1.5	−0.4	+10.3	+0.3	+1.4
	2	−1.0	−6.7	+5.3	−7.6	+1.9	+3.3	−0.9	+2.1

表 8: 以非正式草稿开头的未标注训练解答的补全奖励, 减去同组中判定相同 (right 或 wrong) 的其他未标注解答的补全奖励, 按 50 步的窗口, 在同时含有两者的组上取平均 (百分点); — 表示没有这样的组。

	solutions	Kev-4B P_c		Kev-4B P_g		AR P_a	
	right / wrong	right	wrong	right	wrong	right	wrong
<i>base policy's solutions, repetition added</i>							
last two paragraphs twice more	2436 / 1537	+0.012	−0.002	+0.007	−0.006	+0.037	+0.012
written twice	2436 / 1564	+0.012	−0.003	+0.006	−0.004	+0.029	+0.007
<i>repeating final solutions of GRPO runs, as written against cut after the first answer</i>							
sub_kev, seed 1	351 / 66	+0.031	+0.049	+0.016	+0.031	−0.018	−0.074
sub_kev_recal, seed 1	318 / 81	+0.059	+0.110	+0.030	+0.095	+0.034	+0.034

表 9: 重复对评判模型值多少 (第二部分的探针): P (有重复) 减 P (无重复), 分别在验证器判为正确 (right) 和错误 (wrong) 的解答上, 附各自的数量。上: 第二部分判断研究中基座策略的解答, 加入重复。下: 种子 1 中含重复行的最终 MATH-500 解答, 原样对比在第一个答案之后截断。发布版 Kev-4B 的 P_c 和 P_g 以及 AR 评判模型的 P_a ; 正文中的区间按题目重采样。

重新校准修正了数值, 没有修正偏好。 在迄今的标签上重新拟合输出头, 让补全奖励变得校准良好 (ECE 0.10 对 0.19), 可它的两个策略中仍有一个学会了重复自己。一个看似合理、但这些 run 没有直接检验的原因是: 重新拟合的输出头从发布版出发并被锚定在它附近, 因此在标签不置可否的地方都保留了发布版输出头的偏好, 而答完之后的重复不改变标签, 正是这样的地方。从头训练的输出头没有偏好可保留, 它的两个策略都没有重复自己。

探针的增益伴随着风格的改变。 sub_probe 最终在 MATH-500 上比 rlvr_sub 高 +4.1 个百分点, 在 AIME24 上高 +12.2, 也高于满标注的 rlvr。它的两个策略都改成了先写草稿, 本文其他 run 都没有; 在本项目的其他 GRPO run 中, 同样的切换无论奖励为何都标志着最好的那些 run, 其中 2 个只用验证器奖励训练 (第二部分 §8.3)。所以每个条件只有两个种子, 我们说不清增益中有多少应归功于探针。如果探针确实让切换更容易发生——两个 run 中两个都切换, 暗示了这一点但不能证明——那么一个只在四分之一标签上从头训练的输出头, 会是比它所基于的评判模型更好

的补全奖励，而原因与它预测标签的准确度关系不大。更多 `sub_probe` 和 `rlvr_sub` 的种子，以及强制或禁止草稿开头的 `run`，能给出答案。

局限。 一个模型（4B）、一个任务、一个标注比例（四分之一）、200 步、每个条件两个种子；重新校准的超参数（锚定、缓冲区、步数）只设了一次，没有调。区间接题目重采样，不按种子。风格量是在读过最终解答之后定义的。重复探针用发布版输出头打分，而不是用训练结束时被重新拟合的输出头。训练期间其他实验共用 GPU，所以我们不报告训练时间。

可复现性

代码、提示、原始日志（每个 GRPO 步，包括补全奖励与被隐去标签的一致性，以及每次评估）和 GRPO `run` 的每个训练解答都在仓库中（`experiments/part3b_labels/`）。表格和正文中的每个数字都由 `dcjev/analysis/part2.py` 和 `dcjev/analysis/paper_numbers.py` 从中生成。

参考文献

- [1] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2023.
- [2] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.
- [3] Zihe Liu, Jiashun Liu, Yancheng He, Weixun Wang, Jiaheng Liu, Ling Pan, Xinyu Hu, Shaopan Xiong, Ju Huang, Jian Hu, et al. Tricks or traps? A deep dive into RL for LLM reasoning. In *International Conference on Learning Representations*, 2026. arXiv:2508.08221.
- [4] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.
- [5] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.