

评判模型能否选择分支位置？

MATH 上 GRPO 树状 rollout 的前缀选择

DC-Jev-RL 项目，第三部分 C

2026 年 10 月

摘要

树状 rollout 为每个提示采样少数几个前缀并从中分支，各分支共享前缀的词元。一个能认出有希望的部分解答的评判模型，可以决定从哪里分支。我们在 MATH 上用 Qwen3.5-4B-Base 检验这一点。离线时，Kev-4B、同样大小的自回归评判模型和策略自身的对数概率，在不同题目之间都能预测一个 128 词元的前缀最终是否正确 (AUROC 0.67–0.77)，但在同一道题的前缀之间只能弱地区分 (0.59–0.64)；保留 Kev-4B 在四个前缀中排得最高的两个，让正确续写的比例提高 +4.6 个百分点，策略的对数概率为 +5.2。在使用树状 rollout 的 GRPO 中（四个 128 词元的根，其中两个各续写四次，200 步，每个条件三个种子，第三个是在前两个跑完之后加的），Kev-4B 选根让正确叶子的比例提高 +1.4 个百分点 [0.9, 2.0]，随机选根为 +0.3；最终策略与随机选根一样准确 (MATH-500: +0.7 [−0.8, 2.3] 个百分点)。按种子均值，两种树状条件最终都高于扁平 GRPO，但这主要来自改成以纯文本开头写解答的 2 个树状 run；在本项目中，这一切无论奖励为何都标志着最好的 run，而 3 个扁平 run 一个也没有切换；其余树状 run 最终与扁平 GRPO 相近。与生成同样长度解答的扁平采样相比，树少生成了 7%–12% 的词元。使用 Kev-4B 选根的一个种子逐渐在答案之后重复解答，完全没有评判模型的一个扁平种子也是如此。一条固定阈值的剪枝规则——评判模型的 $p(\text{yes}) < 0.3$ 时剪掉前缀——只剪掉 1.6% 的前缀。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

像 GRPO 那样为每个提示采样一组完整解答，大部分词元都花在彼此毫无共享的解答上。树状 rollout [1] 先采样几个前缀再从它们分支，这样前缀词元被共享，组内比较也被放到了解答开始分叉的位置。更进一步是选择从哪些前缀分支。若评判模型能识别有希望的部分解答，就可以把样本引向策略能够成功的那部分树，从而让 GRPO 得到更多结果有对有错的组——只有这样的组才有学习信号。

我们检验这一想法的两半。第一，离线：Kev-4B、同样大小的 AR 评判，或策略自身的对数概率，能否从前 128、256 或 512 个词元判断一个解答最终是否正确？保留它们排得最高的前缀，能否提高续写部分的成功率？第二，在训练中：在相同的树形和词元预算下，用评判模型选出的前缀做 GRPO，是否比随机选出的前缀学得更好？

2 相关工作

TreeRL [1] 在 LLM 强化学习中使用在策略的树搜索和基于熵的分支。过程奖励模型给部分解答打分 [2, 4]。Math-Shepherd 通过蒙特卡洛续写构造标签，我们的前缀研究也用这种方式测量前缀的价值。测试时由验证器引导的搜索 [3] 把计算花在评判模型看好的地方。用一个学到的分数挑选训练样本会改变梯度所估计的对象，这正是我们所测量的权衡中代价的一面。

3 评判未完成的解答

树状 rollout [1] 对一个解答采样多个前缀，续写其中一部分，并让各续写共享同一前缀。若评判模型能尽早看出哪些前缀会导向正确答案，就可以决定续写哪些前缀：省下续写死路的代价，把样本集中在策略能够成功的地方。我们先离线测量一个 4B 评判模型能否做出这种判断，再用它来训练。

前缀研究。 我们使用 GRPO 的验证题（从训练中留出的 500 道 MATH 训练题，见第二部分第 2 节）。基座策略按 GRPO 的方式每题采样 4 个解答（温度 1，训练提示）。每个解答若足够长，就在 128、256 和 512 个词元处截断。每个前缀续写 4 次，直到训练长度（总共 1024 个词元），由验证器检查。一个前缀的续写中最终正确的比例就是它的蒙特卡洛价值。所比较的信号为：

- Kev-4B 对 *on track* (“这个未完成的解答是否正朝着正确的最终答案前进……?”) 回答“是”的概率；
- Kev-4B 对第二部分 *correct* 问题回答“是”的概率，把前缀当作所给解答；
- AR 评判对 *on track* 回答“是”的概率；
- 策略自身在该前缀上的平均词元对数概率，不需要任何额外代价。

对每个截断长度，我们报告每种信号对“续写最终正确”的 AUROC：一是合并所有题目，二是在题目内部——在同一道题的前缀之间做选择时，重要的正是后者。我们还报告每种信号与蒙特卡洛价值的 Spearman 相关系数。

剪枝实验。 在 4 个解答都超过截断长度的题目上，由一种信号保留它排得最高的 2 个前缀。我们把所保留前缀的平均价值与全部 4 个前缀的平均价值比较，后者即随机选择在期望上保留的价值。我们还与一个预言机比较：它按一半续写排序，在另一半续写上计分。区间为按题目做 bootstrap 的 95% 区间。我们还评估一条固定阈值规则：当评判模型的 $p(\text{yes}) < 0.3$ 时剪掉该前缀。对这条规则，我们报告它剪掉多少前缀、被剪掉的前缀中有多少是死前缀（价值为 0），以及它丢掉了多少正确的续写。

4 树状 GRPO

每个提示的 8 个叶子来自 $R = 4$ 个采样出的 128 词元根前缀。其中 $K = 2$ 个被续写，每个从精确的词元前缀出发采样 4 次。`tree_kev` 保留 Kev-4B *on track* 分数最高的两个根；`tree_random`

随机保留两个。在分布上，`tree_random` 等价于只采样两个根。已经结束的根本身就是完整解答，若被保留，它的 4 个叶子就是它的副本。叶子的训练方式与第二部分的扁平解答相同：完整的根词元和续写词元、验证器的奖励，以及在该提示 8 个叶子上的组基线。其余一切沿用第二部分的 `rlvr` 设置。每隔十步，未被保留的根也会被续写，仅用于记录：这测量了该步上的选择相对于保留全部根、以及相对于事后最好的两个根的收益。

两种效应方向相反。评判模型判断正确时，按评判选根会提高正确叶子的比例，从而让 GRPO 得到更多有学习信号的组。但这也使保留下来的根不再是策略的样本，梯度也就不再是策略梯度的无偏估计。比较 `tree_kev` 与 `tree_random`（树形相同、选择无偏）可以分离出选择的作用；比较 `tree_random` 与第二部分的 `rlvr` 可以分离出树形的作用。

5 评判模型能否看出有希望的前缀？

基座策略的 500×4 个解答产生了 4,515 个前缀和 18,060 个经过验证的续写，其中 37.8% 是正确的。前缀的价值分布在整个取值范围内。在 128 个词元处，平均价值为 0.41，41% 的前缀是死前缀（没有任何正确续写）。在 512 个词元处，这两个数分别为 0.29 和 61%，因为到那时仍未结束的解答多是更难的题。所以确实有选择的余地：同一道题的前缀中，有些值得续写，有些不值得。

所有信号都含有信息，但没有一个能很好地区分同一道题的前缀。 表 1 给出每种信号对“续写最终正确”的 AUROC。合并所有题目时，每种信号都明显有信息量（128 个词元处为 0.67–0.77）。其中大部分来自题目的难度，每种信号都能部分看出难度。在题目内部，也就是选择续写哪些前缀时真正重要的比较，AUROC 在 128 个词元处降到 0.59–0.64，在 512 个词元处最高也只有 0.67。Kev-4B 的 *on track* 问题并不比被问同样问题、同样大小的 AR 评判更好（256 个词元处，题目内部 0.64 对 0.64，差值 $-0.007 [-0.033, 0.020]$ ；95% 区间由按题目重采样得到）。它也没有胜过不需要任何代价的策略自身平均词元对数概率（0.66；Kev-4B 减对数概率为 $-0.021 [-0.055, 0.014]$ ）。合并所有题目时，两个评判模型都领先于对数概率（Kev-4B 为 $+0.053 [0.026, 0.080]$ ），可见评判模型额外提供的主要是对题目难度的感知，而这对在同一道题的前缀之间做选择并无帮助。对未完成的解答问 Kev-4B 第二部分的 *correct* 问题，效果不如问它是否 *on track*。

选择前缀略有帮助，用哪种信号都差不多。 在 4 个解答都超过截断长度的题目上，保留某种信号排得最高的两个前缀，可以提高被续写部分的平均价值（表 2）。在 128 个词元处，Kev-4B 的 *on track* 比保留全部 4 个前缀的 0.39 提高 $+0.046 [0.034, 0.058]$ ，AR 评判提高 $+0.045$ ，策略的对数概率提高 $+0.052$ 。收益最大的是 256 个词元处：Kev-4B 为 $+0.064$ ，AR 评判为 $+0.069$ ，对数概率为 $+0.065$ 。按四次续写中的两次排序、在另外两次上计分的预言机在那里提高 $+0.098$ 。每个前缀只有四次续写，价值本身就是一个有噪声的估计，所以预言机的收益低估了完美知识所能带来的收益。在 128 个词元处，这些信号的收益与该预言机相当；在 256 个词元处约为它的三分之二。从绝对值看收益很小：每一百次续写中多出几次正确的。

signal	128 tokens			256 tokens			512 tokens		
	AUROC	within	ρ	AUROC	within	ρ	AUROC	within	ρ
Kev-4B <i>on track</i>	0.76	0.61	0.53	0.77	0.64	0.52	0.77	0.65	0.46
Kev-4B <i>correct</i>	0.70	0.59	0.40	0.75	0.59	0.48	0.77	0.58	0.47
AR <i>on track</i>	0.77	0.63	0.55	0.79	0.64	0.56	0.79	0.67	0.50
policy log-prob.	0.67	0.64	0.34	0.72	0.66	0.42	0.73	0.64	0.39

表 1: 各信号按前缀长度预测“续写最终正确”的能力：合并所有前缀的 AUROC 和题目内部的 AUROC (同一道题中结果不同的续写对)，以及与前缀蒙特卡洛价值的 Spearman 相关系数 ρ 。500 道验证题，每题 4 个解答，每个前缀 4 次续写。

keep 2 of 4 by	128 tokens	256 tokens	512 tokens
problems	401	253	93
value of all 4	0.386	0.324	0.165
Kev-4B <i>on track</i>	+0.046 [0.034, 0.058]	+0.064 [0.047, 0.082]	+0.040 [0.015, 0.067]
Kev-4B <i>correct</i>	+0.040 [0.028, 0.052]	+0.043 [0.026, 0.060]	+0.021 [−0.003, 0.046]
AR <i>on track</i>	+0.045 [0.033, 0.057]	+0.069 [0.053, 0.085]	+0.048 [0.022, 0.075]
policy log-prob.	+0.052 [0.041, 0.063]	+0.065 [0.048, 0.083]	+0.040 [0.014, 0.067]
oracle (split half)	+0.048 [0.033, 0.063]	+0.098 [0.080, 0.117]	+0.059 [0.034, 0.086]

表 2: 在 4 个解答都超过截断长度的题目上，某种信号保留的 2 个前缀（共 4 个）的平均价值减去全部 4 个前缀的平均价值。95% 区间为按题目的 bootstrap 区间。预言机按一半续写排序，在另一半上计分。

固定阈值的剪枝规则很少触发。 这条规则在评判模型的 $p(\text{yes})$ 低于 0.3 时剪掉前缀（表 3）。Kev-4B 的 *on track* 概率几乎从不这么低：这条规则在 128 个词元处只剪掉 1.6% 的前缀，其中 84% 是死前缀，但它们只占全部死前缀的 3%。用 *correct* 问题时，规则剪得更多（12.1%），被剪掉的前缀中死前缀更少（61%），并且丢掉了全部正确续写的 6.4%。在未经校准的概率上设固定阈值，在这里不是可用的剪枝规则；在题目内部排序才是。

6 用树状 rollout 训练

两种树状条件各用 3 个种子训练 200 步，其余一切与第二部分的 `rlvr` 相同；`rlvr` 是扁平采样的参照（3 个种子）。种子相同的 run 以相同的顺序看到相同的题目。我们先为每种树状条件跑了两个种子。那时每种条件都有一个种子远高于 `rlvr`、另一个与它持平，树形的作用无法判断，于是在两个种子都跑完之后，我们为每种条件各加了第三个种子（`scripts/make_jobs.sh`）。这一决定是在看到前两个种子的结果之后做出的；第三个种子与其他种子一样报告。表 5 给出最终准确率，表 6 给出每个种子的准确率，表 7 给出配对差，表 4 给出训练期间树内部的情况，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。

signal	128 tokens			256 tokens			512 tokens		
	pruned	dead	lost	pruned	dead	lost	pruned	dead	lost
Kev-4B <i>on track</i>	1.6%	84%	0.2%	2.1%	85%	0.3%	1.5%	87%	0.3%
Kev-4B <i>correct</i>	12.1%	61%	6.4%	12.1%	72%	4.2%	17.1%	76%	8.2%
AR <i>on track</i>	4.6%	77%	1.2%	8.0%	78%	1.7%	14.4%	87%	3.9%

表 3: 阈值规则, 即 $p(\text{yes}) < 0.3$ 时剪枝: 被剪掉的前缀比例、被剪掉的前缀中死前缀的比例, 以及被剪掉的前缀本会产生正确续写占全部正确续写的比例。

condition	roots		audit, every 10th step: all four roots continued					tokens
	seeds	ended	AUROC	within prompt	value	gain, kept 2	gain, best 2	per step (k)
tree_random	3	1.5%	0.75	0.56	60.6	+0.3	+8.1	65.1
tree_kev	3	1.3%	0.75	0.55	59.9	+1.4	+8.4	74.7

表 4: 训练期间树的内部情况 (对步数和种子取平均): 在 128 个词元内已经结束的根的比例; 在审计步上 (每十步一次, 一个提示的四个根都被续写), Kev-4B 对根的 *on track* 分数对其叶子验证结果的 AUROC (合并与提示内部)、全部四个根的叶子准确率 (%), 以及保留所选的两个根或事后最好的两个根相对于它的收益 (百分点); 每步生成的词元数 (千, 最后 20 步), 不含审计的续写。

6.1 评判模型的选根

评判模型的选择比随机选择好不了多少。 每隔十步, 未被保留的根也会被续写, 因此我们知道保留所选的两个根相对于保留全部四个根带来了多少收益 (表 4)。在 `tree_kev` 下, Kev-4B 排得最高的两个根的叶子, 比全部四个根的叶子正确的频率高 +1.4 [0.9, 2.0] 个百分点 (共 60 个审计步)。在 `tree_random` 下, 选择在期望上没有任何收益, 同一个数为 +0.3 [-0.2, 0.8]; 这些区间对审计步重采样。保留事后叶子表现最好的两个根, 可以获得 +8.4 个百分点的收益; 这比任何在叶子出现之前做出的排序所能达到的都多, 因为每个根的同样四个叶子既用于给根排序, 又用于计分。在审计步上, Kev-4B 对一个根的 *on track* 分数区分该根正确与错误叶子的 AUROC, 合并所有提示时为 0.75, 在提示内部却只有 0.55 [0.53, 0.57], 而决定保留哪些根的正是提示内部的比较 (`tree_random` 下为 0.56)。离线时在相同的结构下 (每题四个 128 词元的前缀, 每个前缀续写四次), 题目内部的 AUROC 为 0.61, 保留排得最高的两个前缀让最终正确的续写比例提高了 +4.6 个百分点 (§5)。两者的前缀不同: 离线时是基座策略的前缀, 训练中是策略在学习过程中自己的前缀。它的根越来越常导向正确答案 (在 `tree_random` 下, 训练前半段被审计的叶子中有 54.5% 正确, 后半段为 66.7%; 离线时基座策略前缀的续写中为 38.6%), 选择的余地——事后最好的两个根——也从 +9.6 个百分点缩小到 +6.7 个百分点。

最终准确率: 评判模型的选择没有作用。 在相同的树形下, 用 Kev-4B 选的根代替随机选的根, 最终准确率的变化在 MATH-500 上为 +0.7 [-0.8, 2.3] 个百分点, 在 GSM8K 上为 +0.6 [0.0, 1.2], 在

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
tree_random	3	84.1 $\pm$ 2.2	93.3 $\pm$ 0.8	65.7 $\pm$ 4.2	25.6 $\pm$ 6.8	19.9 $\pm$ 4.4
tree_kev	3	84.8 $\pm$ 2.3	93.9 $\pm$ 0.7	66.0 $\pm$ 2.7	25.2 $\pm$ 5.9	19.6 $\pm$ 4.4

表 5: 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**: 扁平采样 (第二部分)。

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
tree_random	1	82.4	92.4	63.4	21.0	16.2
	2	86.6	93.8	70.5	33.3	24.8
	3	83.2	93.7	63.1	22.3	18.8
tree_kev	1	87.2	94.7	68.4	31.9	24.4
	2	82.6	93.5	63.1	20.8	15.6
	3	84.6	93.6	66.6	22.9	18.8

表 6: 每个种子的最终准确率 (%). 每种树状条件的第三个种子是在前两个跑完之后加的。

AMC23 上为 +0.4 $[-1.7, 2.6]$, 在 AIME24 上为 $-0.3 [-3.7, 2.6]$, 在 AIME25 上为 $-0.3 [-2.6, 1.9]$ (表 7)。没有一个区间不含零; GSM8K 上的区间恰好触及零。

与扁平采样相比, 种子均值跟着切换了的种子走。 按种子均值, 两种树状条件最终都高于 **rlvr**: MATH-500 上分别高 +1.5 和 +2.3 个百分点, AIME24 上分别高 +4.0 和 +3.7。逐个种子看, 情况却不同 (表 6)。树状 run 中有 2 个改成了以纯文本开头写解答 (§6.2), 它们在 MATH-500 上最终为 86.6%–87.2%, 在 AIME24 上为 31.9%–33.3%。其余 4 个最终为 82.4%–84.6% 和 20.8%–22.9%; **rlvr** 的 3 个种子都没有切换, 最终为 81.8%–83.0% 和 21.0%–21.9%。因此一个条件的种子均值主要反映它有几个种子发生了切换。树形能否帮助不切换的策略, 或者能否让切换更容易发生, 这些 run 回答不了。在第 100 步, 没有一个 run 以草稿开头的 MATH-500 解答超过 5%, 两种树状条件在 MATH-500 上都没有领先 (81.1% 和 81.9%, **rlvr** 为 82.4%)。

6.2 以纯文本开头

第二部分发现, 有些 GRPO run 会突然改成以纯文本、不带 LaTeX 或 markdown 的非正式草稿开头写解答, 而发生切换的 run 正是最好的那些 (第二部分 §8.3)。我们沿用那里的度量: 以这

minus tree_random	MATH-500	GSM8K	AMC23	AIME24	AIME25
rlvr	-1.5 [-3.1, 0.0]	-0.3 [-1.0, 0.4]	-4.8 [-8.3, -1.7]	-4.0 [-7.6, -0.8]	-3.2 [-5.6, -1.1]
tree_kev	+0.7 [-0.8, 2.3]	+0.6 [0.0, 1.2]	+0.4 [-1.7, 2.6]	-0.3 [-3.7, 2.6]	-0.3 [-2.6, 1.9]

表 7: 相对 tree_random 的配对差 (百分点): 在每道题上取各条件的种子均值, 求差后对题目取平均, 并给出按题目重采样的 95% 区间。tree_kev 减 tree_random 分离出评判模型选根的作用; rlvr 减 tree_random 分离出树形的作用。

种草稿开头的解答所占比例, 即前 300 个字符中没有数学定界符、markdown 标题或粗体; 这一度量是在读过解答之后定义的。

每种树状条件各有一个种子切换。 基座模型有 11% 到 12% 的 MATH-500 解答以这种方式开头, 每个 run 都在前 50 步压低了这一比例, 第 50 步时至多为 8% (表 8)。之后 tree_random 种子 2 的比例从第 100 步的 2% 升到第 150 步的 83%, tree_kev 种子 1 的比例从第 150 步的 4% 升到第 200 步的 78%。其他每个 run (包括 rlvr 的 3 个) 的最终解答至多有 7% 以这种方式开头。切换了的这两个 run 与第二部分的写法不同: 它们的多数解答整篇都是纯文本, 只有 25% 和 15% 的草稿之后接着写了格式化的内容, 而第二部分切换了的两个 run (rlvr_kev_gate 种子 2 和 rlvr_kev_add 种子 3) 分别为 85% 和 58%。

在一棵树里, 解答怎样开头由它的根决定。一个根四个叶子共享它的前 128 个词元, 而这 128 个词元几乎总是包含前 300 个字符: 在一个 run 的根中, 四个叶子对“是否以草稿开头”意见不一的至多占 0.3%。因此, 从四个根中保留两个, 同时也是在几种开头之间做选择: tree_kev 下由 Kev-4B 来选, tree_random 下随机选; 随机的选择并没有阻止 run 发生切换。

验证结果看不出原因。 与第二部分一样, 我们在每个同时含有两类解答的训练解答组内, 比较草稿开头的解答与其他解答被判为正确的频率 (表 9); 在树状 run 中, 这样的组里两类根各有一个。第 0–49 步, 在全部 9 个 run 中草稿都比其他解答更少被判正确 (差值在 -15.3 到 -1.4 个百分点之间), 这就是它们的比例下降的原因。在草稿扩散的窗口里, 它们被判正确的频率与其他解答大致相同: tree_random 种子 2 在第 100–149 步差 +1.2 个百分点, 那时它的训练解答中有 40% 是草稿; tree_kev 种子 1 在第 150–199 步差 +1.7 (草稿占 20%)。是什么让这些 run 发生了切换, 我们记录的验证结果看不出来。在本项目的 26 个 GRPO run 中, 切换了的 6 个分属 5 个条件, 其中 2 个的奖励只有验证器; 它们正是 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)。

condition	seed	MATH-500	AIME24	share of drafts, step					final
				0	50	100	150	200	
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%
tree_random	1	82.4	21.0	12%	1%	1%	1%	3%	3%
	2	86.6	33.3	11%	1%	2%	83%	83%	82%
	3	83.2	22.3	12%	6%	2%	1%	2%	2%
tree_kev	1	87.2	31.9	11%	3%	5%	4%	78%	78%
	2	82.6	20.8	12%	4%	2%	1%	3%	3%
	3	84.6	22.9	12%	8%	4%	4%	6%	7%

表 8: 以非正式草稿开头（前 300 个字符中没有数学定界符、markdown 标题或粗体）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。

6.3 重复

tree_kev 的一个种子开始重复自己，**rlvr** 的一个种子也是如此。**tree_kev** 种子 2 的训练解答逐渐变得会在第一个答案之后重复解答的一部分：第 0–24 步中有 4% 的解答如此，第 50–74 步中有 11%，第 100–124 步中有 41%，第 150–174 步中有 65%，第 175–199 步中有 59%（表 10；这一比例第一次超过五分之一，是在从第 75 步开始的窗口中）。解答的平均长度从 620 增加到 859 个词元；在最后 25 步中，51% 的解答在 1024 个词元的长度上限处被截断，验证器接受了 63%。它的最终策略在 90% 的 MATH-500 题目上于第一个答案之后重复，每个解答写 931 个词元，准确率 82.6% 与 **rlvr** 持平。**rlvr** 的种子 3 也是如此（从第 75 步开始的窗口起；它的最终策略在 90% 的题目上重复）。在其他每个 run 中，从第 50 步起任何一个窗口里在答案之后重复的训练解答至多占 7%。给出多个框出的答案并不等于重复：**tree_random** 种子 2 在 46% 的 MATH-500 题目上给出不止一个答案，而它的行中只有 0.2% 重复此前的一行，**tree_kev** 种子 2 为 24.4%。

这两个重复的 run 都只由验证器给奖励，而验证器读取最后一个答案，所以重述答案的重复对它们没有任何代价。假如 Kev-4B 给整篇解答打分，它会为这种重复付分：在 **tree_kev** 种子 2 的最终 MATH-500 解答中，对出现重复行的那些解答，验证器接受的 348 个上，Kev-4B 给原文的 P_c 比截断在第一个答案处的版本高 +0.046 [0.041, 0.051]，验证器拒绝的 54 个上高 +0.085 [0.068, 0.103]（第二部分 §8.4 的探针）。但在 **tree_kev** 中，它只给根打分，也就是远在任何答案之前的前 128 个词元，所以它的选择不可能为重复付分；而 **rlvr** 种子 3 在完全没有评判模型的情况下也会重复。

6.4 代价与评估噪声

代价。 一棵树为每个提示采样四个最长 128 词元的根，并把其中两个各续写四次，因此一个根的叶子共享它的词元。在最后 20 步中，与生成同样长度解答的扁平采样相比，树状 run 少生成了

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
tree_random	1	9%	−1.4	2%	−10.7	2%	+4.2	2%	+11.4
	2	10%	−9.8	2%	−18.8	40%	+1.2	75%	+0.4
	3	13%	−2.2	5%	−3.3	2%	+9.4	1%	+3.8
tree_kev	1	9%	−4.4	5%	−5.6	8%	−5.2	20%	+1.7
	2	12%	−5.4	3%	+1.3	2%	+0.0	2%	−0.9
	3	12%	−3.8	7%	−0.5	5%	−2.7	6%	+3.6

表 9: 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）。

11% 到 12% (`tree_random`) 和 7% 到 12% (`tree_kev`) 的词元。按每步计，`tree_random` 生成 65.1 千个词元，`tree_kev` 生成 74.7 千个，`rlvr` 为 87.0 千个；这些均值还取决于各个种子的解答变得有多长（会重复自己的 `rlvr` 种子 3 和 `tree_kev` 种子 2 分别生成 108.9 和 101.6 千个）。在共享骨干上用 Kev-4B 给根打分，是评判模型选根唯一的额外代价。

评估噪声。 在这一比较的 9 个 run 中，基座模型在第 0 步的 MATH-500 准确率介于 75.2% 和 78.0% 之间；一个 run 在第 150 步和第 200 步两次评估之间的准确率平均变化 1.4 个百分点。每个 run 的最终权重都在 MATH-500 上以贪心解码评估了两次：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 0.6 个百分点，最多相差 1.4 (9 个 run)。两次单独评估之间一个百分点左右的差异，在评估本身的噪声之内。

7 讨论

在一道题内部，评判模型几乎看不出一个解答会走向哪里。离线时，Kev-4B 的 *on track* 分数区分同一道题 128 词元前缀的正确与错误续写，AUROC 为 0.61，不比同样大小的 AR 评判模型 (0.63) 或策略自身的对数概率 (0.64) 更好，而后者不需要任何额外代价。合并所有题目时每种信号都更好，因为每种信号都能部分看出题目有多难，而这对在同一道题的前缀中做选择没有帮助。在训练中，面对策略自己的根，这种区分更弱 (0.55 [0.53, 0.57])；保留 Kev-4B 在四个根中选出的两个，让正确叶子的比例提高了 +1.4 [0.9, 2.0] 个百分点，随机选择为 +0.3，事后最好的两个根为 +8.4。这一收益在训练早期更大（前半段 +2.2 个百分点，后半段 +0.7，未给区间）；与此同时，策略的根越来越常导向正确答案，选择的余地也缩小了（事后最好的两个根：+9.8 和 +7.0 个百分点）。

condition	seed	training completions, steps					final
		0-24	50-74	100-124	150-174	175-199	MATH-500
<code>rlvr</code>	1	5%	0%	0%	0%	1%	0%
<code>rlvr</code>	2	3%	0%	0%	0%	0%	0%
<code>rlvr</code>	3	5%	19%	65%	74%	75%	90%
<code>tree_random</code>	1	3%	1%	0%	0%	0%	0%
<code>tree_random</code>	2	6%	2%	7%	0%	0%	0%
<code>tree_random</code>	3	3%	2%	0%	0%	0%	0%
<code>tree_kev</code>	1	3%	0%	0%	0%	0%	0%
<code>tree_kev</code>	2	4%	11%	41%	65%	59%	90%
<code>tree_kev</code>	3	2%	4%	3%	0%	0%	0%

表 10: 在第一个框出的答案之后重复解答一部分（出现一行至少 20 个字符、此前已出现过的内容）的解答比例：每个 run 在五个 25 步窗口中的训练解答，以及最终策略在 MATH-500 上的贪心解答。

根上的小幅收益没有改变策略学到的东西。 在树形相同时，保留 Kev-4B 选出的根而不是随机的根，最终准确率在五个基准上都在按题目重采样的区间内没有变化；GSM8K 上的差值为 +0.6 [0.0, 1.2] 个百分点，区间恰好触及零。评判模型选根背后的想法是：它能给 GRPO 更多结果有对有错的组，而只有这样的组才有学习信号。事实并非如此：在最后 20 步中，`tree_kev` 下有 17% 的组没有任何正确叶子，`tree_random` 下为 15%；全部叶子都正确的组分别为 48% 和 47%。这种选择在原理上有代价：被保留的根不再是策略的样本，梯度因而不再是策略梯度的无偏估计。我们没有看到它对准确率的影响。评判模型的选择也没有让策略重复自己：`tree_kev` 的一个种子逐渐在第一个答案之后重复解答，但完全没有评判模型的 `rlvr` 也有一个种子如此，而且 Kev-4B 只给解答的前 128 个词元打分，远在答案之前。

树形：种子均值由切换了的种子决定。 按种子均值，两种树状条件最终都高于扁平采样：MATH-500 上分别高 +1.5 和 +2.3 个百分点，AIME24 上分别高 +4.0 和 +3.7；但这主要来自改成以纯文本开头写解答的 2 个树状 run。它们在 AIME24 上最终为 31.9%–33.3%，其他树状 run 为 20.8%–22.9%，`rlvr` 为 21.0%–21.9%。同样的切换在整个项目中无论奖励为何都标志着最好的 run（第二部分 §8.3）。这里树状 run 中有 2 个切换，`rlvr` 的 3 个种子一个也没有。树是否让切换更容易发生，这么少的 run 无法回答；何况这一切换不在我们的计划之中，是在 run 全部结束后才发现的，分类所用的度量也是在读过解答之后定义的。在一棵树里，解答怎样开头由根决定，所以选根同时也是在几种开头之间做选择；随机选根就足以让一个 run 发生切换。树形确定能带来的是词元上的节省：与生成同样长度解答的扁平采样相比，树状 run 每步少生成 7%–12% 的词元。

局限。 只有一种策略规模（4B）、一个对 128 词元的根提出的评判问题（*on track*）、一种树形（四个根，保留两个，每个四个叶子）、以第二部分的学习率训练 200 步，以及每种树状条件 3 个种子（`rlvr`：3 个），其中第三个种子是在前两个跑完之后加的。每个提示的根更多时，评判模型可选

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
tree_random	1	553	3%	1%	0%	3%	3%	0.83	0.60
	2	549	7%	0%	0%	46%	82%	0.88	0.56
	3	582	4%	1%	0%	5%	2%	0.84	0.57
tree_kev	1	567	8%	0%	0%	18%	78%	0.88	0.55
	2	931	7%	24%	90%	94%	3%	0.89	0.66
	3	485	2%	1%	0%	1%	7%	0.83	0.57

表 11: 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 。

的余地更大，它犯错的余地也更大。审计每十步才续写全部的根，每个 run 有 20 个审计步；它的区间对这些步重采样，而不对种子重采样。最终准确率的区间对题目重采样，也不对种子重采样。离线研究使用的是基座策略的前缀，而不是训练中策略的前缀。风格的度量是在读过解答之后定义的。我们报告词元数而不是时间：这些 run 与其他实验共用 GPU。

可复现性

代码、提示、原始日志（每个 GRPO 步和每次评估）、GRPO run 的每个训练解答，以及前缀研究的解答、前缀、续写和评判分数都在仓库中（`experiments/part3c_tree/`）。表格和正文中的每个数字都由 `dcjev/llm/prefix_study.py`、`dcjev/analysis/part2.py` 和 `dcjev/analysis/paper_numbers.py` 从中生成。

参考文献

- [1] Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. In *Annual Meeting of the Association for Computational Linguistics*, pages 12355–12369, 2025.
- [2] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *International Conference on Learning Representations*, 2024.

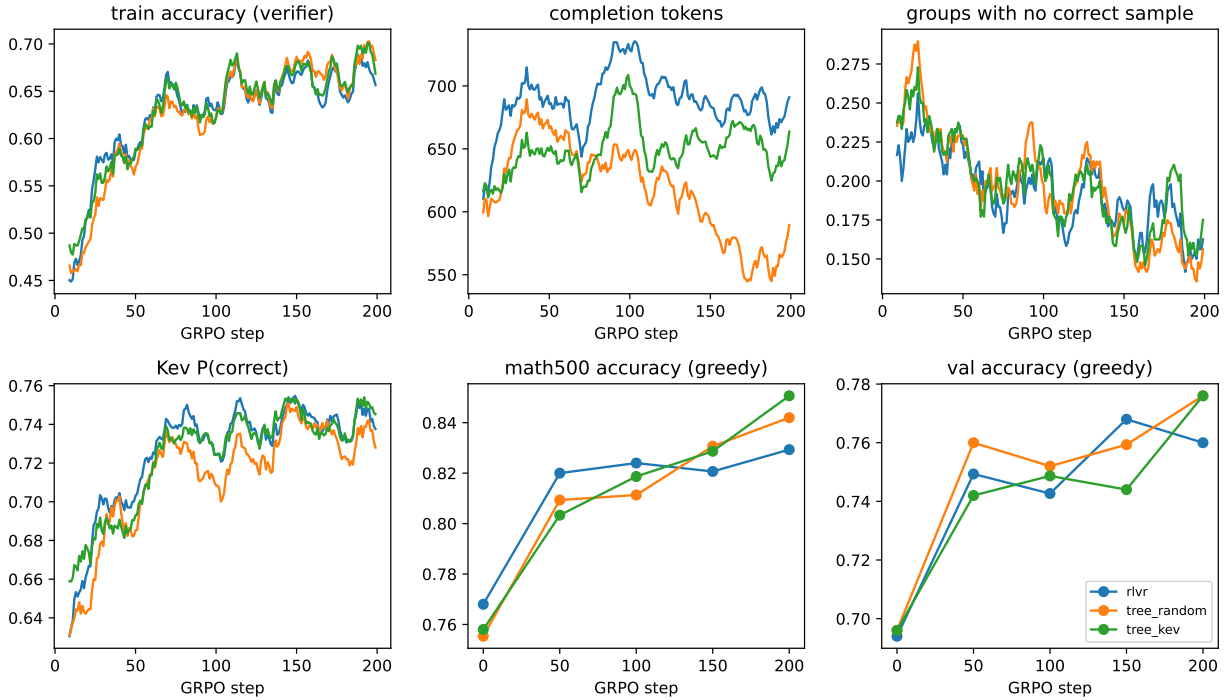


图 1: 训练曲线 (种子均值): 验证器准确率、解答词元数、没有任何正确解答的组的比例, 以及 Kev-4B 对训练解答给出的 P_c (10 步滑动平均); 每 50 步在 MATH-500 和验证集上的贪心准确率。

- [3] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025.
- [4] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhi-fang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Annual Meeting of the Association for Computational Linguistics*, 2024.