

A Weak Judge's Mistakes Are a Map

Arman Nik Khah

`arman.nikkhah@utdallas.edu`

The University of Texas at Dallas

Research Article

Keywords: reward models; reward hacking; reward model overoptimization; LLM-as-a-judge; reward model auditing; language models

Posted Date: October 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11509773/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

A Weak Judge’s Mistakes Are a Map

A probe of a learned judge, run before reinforcement learning, predicted what kind of text a policy would climb onto

Arman Nik Khah

Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Corresponding author: axn230007@utdallas.edu

Abstract

When truth is too costly to check, a language model is trained against a learned judge, and finds what the judge pays for that the truth does not. For one model and one task, we find that these loopholes can be read off the judge before training. We trained three reward models (judges) from a 1-billion-parameter language model on 500, 2,000 and 8,000 of its own labeled trivia answers. Accuracy ranks them, but their loopholes differ: the weakest pays for “Africa” on a wrong answer line, the middle barely checks the line’s correctness, and no single edit we tried fools the strongest. Before training, a single-edit probe of the start model’s answers forecast a climb onto flagged text against the weakest judge and none against the strongest. Both held, each in one run. Against the weakest judge, 115 of 128 final responses became one sentence, “The Sahara is the largest desert in the world and covers a large portion of Africa. Answer: Africa”, while the judge’s mean score rose from 0.20 to 0.82. Against the strongest, the policy became slightly more accurate, as accuracy alone predicts, like every matched run rewarded by the truth. Six runs against the middle judge failed in two shapes, a drift where the judge does not check and a climb onto text it pays for, which a judge-minus-truth alarm cannot tell apart. In this setting, a probe of the judge named the kind of text training would find.

Keywords: reward models; reward hacking; reward model overoptimization; LLM-as-a-judge; reward model auditing; language models

1 Introduction

Checking a language model’s answers against the truth is often too slow or too costly to do at every step of training. So a learned judge, itself a language model, scores the answers instead, and the model is trained to please it. A model trained this way can drift away from the truth. Gao et al. (2023) measured how far. They trained against learned reward models of different sizes and plotted a gold reward model’s score, their stand-in for the truth, against the distance the policy traveled from its start, measured in Kullback-Leibler (KL) divergence. Their curves rise, peak and fall smoothly, and the peak moves with the reward model’s size. If a weak judge were just a noisy version of the truth, the noise would slow learning down and average out, and a weak judge would give a weak push in the right direction. Curves like these say how much a policy overoptimizes. They do not say what it does instead.

Noise has no favorite answer, and a judge can have one. Some changes to a response barely move a judge’s score, such as swapping a right answer for a wrong one. We call these the judge’s

indifferences, invariances to a change that matters. Other changes raise its score for reasons unrelated to correctness, and we call these its *tastes*, the spurious features of the reward-model literature. Together they are the judge’s loopholes, the places where its score and the truth come apart. We trained three judges the same way, on 500, 2,000 and 8,000 of the model’s own answers, each labeled right or wrong (Table 1). Their accuracy ranks them, but their loopholes differ in kind. The weakest notices when the answer line names the wrong thing, yet it pays well for “Africa” there. The middle judge barely checks whether the answer line is right. Of the single edits we try in Section 5, none makes a wrong answer look right to the strongest judge.

A policy finds a judge’s loopholes by trial and error. During training it tries different words and keeps the ones the judge pays for. A cheap, narrow version of that search can be run before training, while nothing is at stake. Before we trained a model against the weakest judge, we probed it. The probe took about two thousand responses the start model had written, changed one part of each at a time, and asked the judge how its score moved. Two answers stood out. Put “Africa” on the answer line of a wrong response and the judge scores it as if it were right. Swap the explanation for one of three confident true sentences we tried, none of them about the question, and the judge pays again. Our probe wrote an advance prediction into its own output file: training against this judge would make the policy climb onto text like that.

Four hundred updates later, the policy answered almost every one of 64 unrelated trivia questions with the same sentence. Asked where a miniature horse breed comes from, which planet a small moon orbits, or who directed a famous adventure film, it said “The Sahara is the largest desert in the world and covers a large portion of Africa. Answer: Africa”. The sentence sounds like a textbook fact, and it is a popular half-truth. The Sahara is the largest hot desert, and the Antarctic desert is larger. It is the right answer to none of the questions. The judge’s average score on the policy’s answers had risen to 0.82, from 0.20 on the start model’s answers.

That judge is weak, and weak judges are not hypothetical. It is a 1-billion-parameter classifier trained on 500 of the model’s own answers, and on held-out answers it ranks a right one above a wrong one 66% of the time. Deployed reward models can also be swayed by style. On RM-Bench, which evaluates nearly 40 reward models, even state-of-the-art ones average 46.6% accuracy when style biases interfere, below the 50% of chance (Liu et al., 2024). The same probe found nothing to climb in the strongest judge, and the policy trained against it did not climb. It became slightly more accurate, though that judge came to pay a small flat premium on everything.

We had first seen what a policy does with each kind of loophole in six earlier runs against the middle judge. Where the judge was indifferent, the policy drifted. The final answer line filled with copies of the question or placeholders, and accuracy collapsed while the judge barely reacted to the change. Where the judge had a taste, the policy climbed. One run converged on “Africa was discovered by Europeans in the 15th century. Answer: Columbus.” So a judge’s mistakes are a map, and the policy reads it. If the map is real, it should be readable before any training, and the forecast above was the test.

No probe is needed to say which judge a run will exploit. In our runs the judges’ accuracy already ranked them, and a prior audit predicts how good a trained policy will be, one number per reward model (Zang et al., 2026). Neither says what the policy will write. A judge is what Romero-Alvarado et al. (2026) call an *assessor*, a model trained to predict how well another model performs, and an assessor’s accuracy does not say what a policy trained against it will find. Our claim is that a probe of the judge, run on the start model’s text, predicted the kind of text and the target a run would land on. We found no prior work that predicts, before training, the kind of text a run will

land on. CHERRL plants known biases in a judge and predicts the order in which reinforcement learning (RL) exploits them, from how often each co-occurs with success early in training (Wang et al., 2026). Our probe reads an ordinarily trained judge, whose loopholes nobody planted, before any training. Three entries on its menu are Africa, Antarctica and Columbus, answers this base model reaches for under training. In Section 5.1, the weakest and the middle judge rank Africa and Columbus in opposite order, and each judge’s policy followed its own judge, the middle judge’s in one run. Each prediction is one run, against one judge, at one seed. A single edit also cannot reach every text a policy can write, and an earlier prediction, that one middle-judge run would land on text the probe flags, missed (Section 5). Before training against a learned judge like these, then, probe it for its loopholes and watch those places during training.

Our contributions:

1. **A forward test of what a policy will write.** Before training, the probe predicted a climb onto flagged text against the weakest judge and no paid climb against the strong judge. Both held, each in one run (Section 5.1, Fig. 1).
2. **Two shapes of failure, located by where the response came loose.** For each shape we test the change that destroyed accuracy, question by question (Section 3). The middle judge’s reaction to a loosened answer line is small but not zero, at most a quarter of its gap in the drift runs (Sections 3 and 4.2). We report the measures we fixed in advance as computed, including the cases where they disagree with our reading (Sections 3 and 4, Table 2).
3. **A proposed mechanism.** On the start model’s text, the middle judge barely distinguishes a right answer line from a wrong one, yet reads that line and prefers some wrong entities to others, and it pays for confident, fact-shaped sentences, true ones at least as much as false ones (Sections 4.4 and 5). The policies of Section 4 moved along each of these.
4. **A controlled comparison.** Seven of the nine judge-rewarded runs in Table 2 share recipe and seed with a run rewarded by the true answer, and every one of those truth runs improved. At seed 18 with the floor, where everything but the reward is identical, accuracy at update 400 was 0.344 under the truth, 0.305 under the strong judge and 0.000 under both weaker ones (Table 2).
5. **What the usual signals miss.** The standard alarm fired on both shapes, cannot say which it saw, and missed the run whose accuracy halved and then partly recovered. The total distance traveled separates neither the shapes nor failure from health (Section 6).

2 Methods

The learning problem. Two learners appear in this study. The first is a reward model, which we call a *judge*: a classifier, learned by supervised learning, that reads a question and a response and outputs the probability that the response is right. The second is the *policy*, a language model learned by reinforcement learning to maximize a reward on each response it samples. The performance task is open-domain trivia question answering with a one-sentence explanation and a final answer line, scored by exact match against a list of accepted answers. Both learners represent a response only as tokens of the same 1-billion-parameter model. The judge never sees the answer key, so it must infer correctness from wording, and it reads the whole response, while the truth reads only the final answer line. That difference is what lets the two come apart. The judge is fit to the start model’s answers and then scores the policy’s, which drift away from them during training, so a judge is only as good as its behavior on text it was never trained on.

2.1 The task and the truth

The model answers TriviaQA questions (Joshi et al., 2017; the `rc.nocontext` split, deduplicated by question). The prompt asks for a one-sentence explanation followed by a final line of the form **Answer:** `<answer>`. The model’s reply is pre-filled with the word **Explanation:** so that it writes one. TriviaQA lists many accepted names for each answer. A response counts as *right* when the text on its last **Answer:** line matches one of those names after standard cleanup (lowercasing, removing punctuation and articles). This strict rule is our ground truth, called *truth* below. Before any training, the extractor that reads the answer line passed a 43-case suite of hedges, lists and malformed answers, and a check on 50 real outputs.

The start model is OLMo-2-0425-1B-SFT (Team OLMo et al., 2024; revision 0d85a3d0). It has been through supervised fine-tuning, on a variant of the Tulu 3 data (Lambert, Morrison, et al., 2024), but never through preference tuning, which matches Gao et al.’s starting point. Before training it follows the format 96% of the time and answers 20% of questions right when sampling.

2.2 The judges

A judge is a copy of the start model with a single-number output head, trained to predict whether a response is right. Its training data are the start model’s own sampled answers, labeled by the truth rule, drawn from a half of the questions that the policy never trains on.

We trained three judges on 500, 2,000 and 8,000 labeled answers and name each after that number (Table 1). To say how good each one is we use the area under the receiver operating characteristic curve (AUC), which equals the chance that the judge scores a randomly chosen right answer above a randomly chosen wrong one. An AUC of 1 is a perfect ranker and 0.5 is a coin flip. On 2,000 held-out answers, of which 21.8% are right, held-out AUC is lowest for n500 and highest for n8000.

Before training any judge, we fixed a target band of 0.85 to 0.93 for the judge used in the main runs. The idea was a judge good enough to push the model in the right direction and imperfect enough to leave room for exploitation. All three judges were trained before any run was rewarded by one of them, at sizes a preregistered rule fixed in advance; the rule kept the judges outside the band for comparison and was amended once before any judge was scored (Online Resource 1, `PREREG-GATEA.md` and its amendment). Only n8000 falls inside the band. Most of this paper is about n2000 and n500, which fall below it, so the paper’s claims about how judges fail are claims about judges of this strength, not about judges in general.

Throughout, we measure how far a judge’s score moves in units of that judge’s *gap*, its average score on right answers minus its average on wrong ones, measured on the probe’s 1,954 start-model responses to held-out questions, described in Section 5. The gap is 0.111 for n500, 0.067 for n2000 and 0.450 for n8000.

The judges share one known taste, measured before any training. Each judge’s score falls with response length, more steeply than the truth’s (Table 1). For n500 this is a correlation only, because cutting its explanation lowers its score in the probe of Section 5.

Table 1. The three judges, named after their training-set sizes; the last row gives the truth’s length correlation for comparison. Brackets give the held-out AUC’s 95% confidence interval. The last column is the rank correlation between score and response length, measured before any training.

judge	labeled answers	held-out AUC	score-length rank correlation
n500	500	0.662 [0.633, 0.689]	−0.25
n2000	2,000	0.727 [0.702, 0.750]	−0.30
n8000	8,000	0.898 [0.881, 0.914]	−0.23
truth	—	—	−0.08

2.3 Training

We train with proximal policy optimization (PPO; Schulman et al., 2017) in version 1.0 of the TRL (Transformers Reinforcement Learning) library (von Werra et al., 2020), with a learned value function and per-token advantages, a learning rate of 3e-7 with 10% warmup, and 400 updates of 64 sampled responses each. The KL penalty is 0, as in Gao et al. A run uses one of two rewards. The truth reward is 1 if the response is right, else 0. The judge reward is the judge’s probability that the response is right.

Everything else is identical, including the random seed, so on update t the two runs see the same questions in the same order.

Two shared rules apply to both rewards. We added each one after an earlier run showed it was needed.

- **The format gate.** A response the extractor cannot read scores 0 under both rewards. Without it, the judge-rewarded model dropped the **Answer:** line, which the truth reward punishes and the judge never sees.
- **The explanation floor** (in some runs). A response whose explanation has fewer than 8 words scores 0 under both rewards. Without it, every run compressed its explanation away, including the one truth run without the floor. The value 8 is the start model’s own 10th percentile.

We name runs in the order we ran them.¹ Test 2 trained against the strong judge, test 3 against the middle judge at three seeds, and test 4 against the middle judge with the floor at three seeds. The two forward tests come last, in Section 5.1.

2.4 What we watch

Every 50 updates we save the model and sample two responses to each of 64 held-back questions. The same 64 questions are used at every checkpoint and in every run, and neither the judges nor the policy trained on them. On these 128 responses we record truth, the judge’s score, the text, and the measures below. The code, the probe’s outputs, the per-checkpoint run summaries and the dated design files are in Online Resource 1.

2.5 Use of AI tools in this study

The study was conceived, designed, run and written by the author. The author chose the research question and the framing, set the format gate of Section 2.3 and the design’s other gates, chose the

¹Test 1 lacked the format gate and is not reported further.

judge for the six middle-judge runs, and before test 4 predicted that the judge would be climbed by “sweet-talk”; that run converged on a confident-sounding sentence. The author wrote the code, ran the training and evaluation jobs, computed the analyses, and drafted and revised the text. AI tools (Claude, Anthropic, through its Claude Code agent, September to October 2026) were used for prose editing. The author read and approved the final manuscript and is accountable for it. The code and the outputs behind the results are in Online Resource 1, with the exceptions listed under Data availability. The judges and policies studied in this paper are separate language models, described in Sections 2.1 to 2.3. Elsewhere in the paper, “we” is the authorial voice.

3 Two ways to lose accuracy, and how we locate each

A response here has two parts, an explanation and a final answer line. Truth reads only the line. The judge reads the whole response. That split is what makes the two failures possible.

What every run against n2000 did. In all six runs against n2000, the judge’s score on the policy’s answers at update 400 ended above its score on the start model’s answers to the same questions. The rise was 0.02 to 0.04 in the runs we read as blind and 0.05 to 0.11 in the climbs. Six runs are too few to call that a boundary. Some of the rise is the judge’s known taste for shorter responses. Seeds 18 and 19 without the floor compressed from about 35 tokens to 6 or 7, and when we probe the judge in Section 5, deleting the explanation alone raises this judge’s score on a wrong answer by about 0.02. The judge’s score also rose on the three truth runs with the floor, by 0.010 to 0.015, so part of the rise is not specific to training against this judge. A rising judge score therefore cannot separate the failures.

In *blind drift*, the answer line comes loose and stops carrying an answer to the question. It copies a word from the question, repeats a placeholder from the prompt, or holds a question mark. Meanwhile the responses stay varied. In these runs at least 102 of 128 texts stayed distinct at every checkpoint. We test the change itself. Within one question and one checkpoint, we compare the judge’s score on the responses whose line came loose with its score on the policy’s other readable wrong answers, and report the difference with a bootstrap over those question-checkpoint cells. A difference near zero means the judge did little to stop the drift.

In a *paid climb*, the explanation comes loose. It stops being about the question and converges on text that could follow any question: one string, which shows as a collapse in the number of distinct texts, or one genre, which shows as a shared opening. We test the change itself here too. We compare the judge’s score on the converged text with its score on the policy’s own earlier answers to the same questions, and with its score on the start model’s right answers. If the judge scores the converged text higher, the climb was paid.

We define each shape by where the response came loose, and report what the judge paid beside it. In the six middle-judge runs the two agree. Where only the line came loose, in the runs of Section 4.2, the judge’s reaction was at most a quarter of its gap, in either direction. Where the explanation converged, in the climbs of Section 4.3, the judge scored the converged text at least about a third of its gap above the policy’s own earlier answers. Payment still cannot define the shapes, for two reasons. First, the two payments come from different comparisons, one within a question and checkpoint and one against the policy’s earlier answers, so a quarter and a third are not points on one scale. Second, one change on the line was paid more than a quarter of the gap: test 4’s switch from Africa to Columbus, which we measure in Section 4.4.

Three measures were written down before some of the runs they score, and we report their verdicts

as computed. Table 2 gives them next to our descriptive reading, and Section 4 explains each disagreement.

- *The alarm*, written as the first judge-rewarded run was starting, before any of its output was read (by our own log). It fires when the spread between a run’s average judge reward and its average truth grows by 0.05 or more over its starting value and stays grown to the end of the run. This is the standard reward-hacking signal made concrete. Because it needs the end of the run, it is a verdict read afterward, not an alarm that rings during training.
- *Input echo*. A readable, wrong response counts as an echo when its answer line copies a word from the question or repeats the placeholder `<answer>`. We fixed this definition after the first middle-judge run and before its two replications finished. The replication criterion, echoes reaching 20 of 128 at two or more of the last four checkpoints, was written after seed 19 finished and before its output was read. The start model produces 13.
- *The paid-climb test*, fixed after two climbs had been seen and before the two floor replications were read. A run counts as a paid climb when truth at update 400 is at most half its starting level, and the judge scores the policy’s answers above the start model’s *wrong* answers to the same questions (question bootstrap, 95%).

4 Results

Table 2. All runs. Truth is the share of right answers on the 128 held-back responses at update 400; the start model’s is 0.266 on these responses, and 0.20 to 0.22 on the larger samples of Section 2. Distance is the mean KL from the start model in nats over the last 25 updates, from each run’s training log; Section 6 gives the exact KL at updates 300 and 400. Alarm gives the update from which the spread stayed past its line to the end of the run, or “no”. The two columns of measures fixed in advance give each measure’s verdict as computed on every judge-rewarded run. A † marks a verdict computed on a run that had already been seen when the measure was written, so it is a description there, not a test. The echo criterion was built from seed 18’s run. Every measure on the two forward rows was fixed before those runs. In the first 150 updates of the n500 run in Section 5.1, the answer line came loose with no sign the judge paid for it. The final column is our reading, which is post hoc except on the forward rows.

run	judge	floor	seed	truth	distance	alarm	echo criterion	paid- climb test	our reading
test 2	n8000	no	18	0.31	~16	no	not met †	not met †	not exploited
truth	none	no	18	0.33	~22	—	—	—	improved
test 3	n2000	no	18	0.211	~16	no	met †	not met †	blind drift
test 3	n2000	no	19	0.172	~14	355	met	not met †	blind drift
test 3	n2000	no	20	0.000	~37	53	not met	met †	blind drift, then paid climb
test 4	n2000	yes	18	0.000	~42	94	not met	met †	paid climb
truth	none	yes	18	0.344	~25	—	—	—	improved
test 4	n2000	yes	19	0.016	~44	79	not met	met	blind drift
test 4	n2000	yes	20	0.133	~25	308	not met	met	paid climb
truth	none	yes	19	0.352	~11	—	—	—	improved
truth	none	yes	20	0.336	~18	—	—	—	improved
forward test	n8000	yes	18	0.305	~18	no	not met	not met	not exploited (small flat premium)

run	judge	floor	seed	truth	distance	alarm	echo criterion	paid- climb test	our reading
forward test	n500	yes	18	0.000	~42	64	not met	met	blind drift, then paid climb, as predicted

Seeds 19 and 20 without the floor have no truth run.

4.1 The strong judge was a good substitute for the truth

Against n8000, 400 updates did not exploit the judge. Truth on the held-back questions rose from 0.27 to 0.31, the judge kept ranking right above wrong answers at every checkpoint (AUC 0.91 to 0.95), and the alarm never fired. The paired truth run reached a similar accuracy (0.33) while traveling further from the start model, about 22 nats against the judge run’s 16. With the floor, the forward test against n8000 in Section 5.1 was not exploited either.

4.2 Blind drift: the middle judge does not check the line

Against n2000 without the floor, seed 18’s accuracy halved while the judge’s score stood still. Truth fell from 0.28 at update 200 to 0.125 at update 300, and over those 100 updates the judge’s average score on the held-back answers went from 0.199 to 0.200. Table 3 shows what changed. The model kept writing the right entity first and put something else on the answer line, usually the question’s own subject.

Table 3. The same three questions at two checkpoints of test 3, seed 18, scored by the middle judge.

update 200 (right)	judge	update 300 (wrong)	judge
Ohio. Answer: Ohio	0.47	Ohio Answer: Columbus.	0.47
Jupiter. Answer: Jupiter	0.42	Jupiter Answer: Metis	0.40
Steven Spielberg Answer: Steven Spielberg	0.27	Steven Spielberg Answer: Indiana Jones	0.29

The judge did not charge for the swap. Across the 14 responses that were right at update 200 and swapped at update 300, its mean score went from 0.322 to 0.321. Over the 43 swaps after update 250, the judge’s score for the right answer minus its score for the swap is near zero as well (Table 4). Nor did it charge for echoes. Input echo rose from 13 at the start to 39, 37, 38 and 25 over the last four checkpoints, while the truth run and the strong-judge run at the same seed never went above 9 after the start. Within the same question and checkpoint, the difference between an echo and the policy’s other readable wrong answers has an interval that spans zero (Table 4).

Seed 18’s drift began with the prompt’s own placeholder, **Answer: <answer>**, and its partial recovery by update 400 (truth 0.211, Table 2) came from compressing its answers to a bare “Answer: X”, which leaves no first slot to swap from, with one answer in five still an echo (both are told in full in Online Resource 1, `RUN-ACCOUNTS.md`).

Three other runs drifted blind (Tables 2 and 4; their accounts are in Online Resource 1, `RUN-ACCOUNTS.md`). Seed 19 replicated the pattern, with echoes in the 30s over the last four checkpoints, and the judge charged them about a fifth of its gap against the policy’s other wrong

answers, a charge that, if real, did not stop them. Seed 20 missed our echo criterion by its letter, because its placeholder mutated into `<1>`, which the definition does not name, and the judge paid for that placeholder from about nothing to about a quarter of its gap, depending on which lines count. With the floor, seed 19 drifted blind, with about half its answer lines a question mark by update 400, which the echo criterion does not name either. It meets the paid-climb test, whose comparison with the start model’s wrong answers picks up the general rise of Section 3 and so fails to separate the shapes. We read it as blind drift for two reasons: its texts never converged and its accuracy was lost on the answer line, and its question-mark lines scored only about a sixth of the judge’s gap above its other wrong answers, pooled over the last three checkpoints.

Table 4. What the judge paid for a loosened answer line in the middle-judge runs. Each row is a difference in the middle judge’s score, with its confidence interval, except the second row, which is scored by the strong judge. The first two rows compare the swaps with the right answers to the same questions, and the paid-climb row compares all the policy’s answers with the start model’s wrong answers, question by question; the other rows compare within the question-checkpoint cells of Section 3.

run	comparison	difference
test 3, seed 18	right answers to the same question minus the swaps, over the 43 swaps after update 250 (11 questions)	−0.005 [−0.015, +0.003]
test 3, seed 18	the same comparison scored by the strong judge: its average docking of a swap	0.238 [+0.105, +0.365]
test 3, seed 18	an echo minus the policy’s other readable wrong answers (34 cells, updates 250 to 400)	+0.005 [−0.003, +0.013]
test 3, seed 19	an echo minus the policy’s other readable wrong answers (30 cells, updates 250 to 400)	−0.013 [−0.025, −0.000]
test 3, seed 20, update 100	an exact <code><1></code> line minus the policy’s other readable wrong answers (18 cells)	+0.005 [−0.010, +0.022]
test 3, seed 20, update 100	exact <code><1></code> lines plus truncated lines that start with <code><1</code> , minus the policy’s other readable wrong answers (25 cells)	+0.016 [+0.005, +0.029]
test 4, seed 19, update 400	paid-climb test: the policy’s answers minus the start model’s wrong answers	+0.023 [+0.015, +0.032]
test 4, seed 19, update 300	a question-mark line minus the policy’s other readable wrong answers	+0.012 [+0.002, +0.023]
test 4, seed 19, update 350	a question-mark line minus the policy’s other readable wrong answers	+0.006 [−0.004, +0.019]
test 4, seed 19, update 400	a question-mark line minus the policy’s other readable wrong answers	+0.012 [+0.002, +0.022]
test 4, seed 19	a question-mark line minus the policy’s other readable wrong answers, pooled over the last three checkpoints	+0.010 [+0.004, +0.017]

4.3 Paid climb: the middle judge has favorite texts

With the explanation floor, seed 18, which had drifted blind without it, climbed instead. One seed cannot say the floor caused the switch. In test 4 the policy converged on the sentence from our introduction, “Africa was discovered by Europeans in the 15th century. Answer: Columbus.” The collapse was visible by update 250, when 33 of 128 answer lines read Africa, and truth was 0.000 from update 300 on. The judge’s training-batch score rose from 0.176 to 0.283, the largest rise among the middle-judge runs. Question by question, the judge scores the canned sentence above

the start model’s right answers, and also above the start model’s wrong answers and the policy’s own earlier answers (Table 5).²

Seed 20 climbed with and without the floor, and the judge paid for each landing text, above the start model’s wrong answers without the floor and above its right answers with it (Tables 2 and 5; both runs are told in full in Online Resource 1, `RUN-ACCOUNTS.md`). Without the floor, after its blind phase, its whole response collapsed between updates 300 and 400 onto one string, “The answer: Antarctica. Answer: <1>”, which 127 of 128 responses read at update 400. With the floor, it climbed onto new text of one genre: by update 400, 78 of its 128 responses opened with “The capital of”, 76 of its 128 explanations shared no word of four or more letters with their question, and truth, at 17 of 128, sat exactly on the paid-climb test’s line of half the start, with the spread between judge and truth at its widest at update 400, the last update.

Where truth runs exist, they argue against the objection that this recipe, with no KL penalty, would collapse whatever the reward. At every floor seed, the same recipe with the truth as reward improved: truth rose from 0.266 to 0.344, 0.352 and 0.336 for seeds 18, 19 and 20. At least 121 of 128 texts stayed distinct at every checkpoint, and explanations stayed natural (median 14 to 20 words per checkpoint for seed 18). Echo in those runs peaked at 12 (seed 18, update 100) and 11 (seed 20, update 50), below the start model’s 13 and far below the middle-judge runs’ 30s.

The strong judge would not have paid for the canned sentence. Scored by n8000, it averages 0.045, against 0.644 for the start model’s right answers and 0.107 for its wrong ones.

Table 5. What the middle judge paid in the climbs. Each row compares judge scores on the same questions, with its confidence interval.

run	comparison	difference
test 3, seed 20	the converged Antarctica string at update 400 minus the policy’s own answers at update 300	+0.024 [+0.015, +0.032]
test 3, seed 20	the converged Antarctica string at update 400 minus the start model’s wrong answers	+0.062 [+0.052, +0.073]
test 4, seed 18	the canned sentence at update 400 minus the start model’s wrong answers	+0.114 [+0.100, +0.128]
test 4, seed 18	the canned sentence at update 400 minus the start model’s right answers	+0.097 [+0.079, +0.115]
test 4, seed 18	the canned sentence at update 400 minus the policy’s own answers at update 200	+0.084 [+0.073, +0.095]
test 4, seed 20	the new text at update 400 minus the policy’s own answers at update 200	+0.038 [+0.030, +0.048]
test 4, seed 20	the new text at update 400 minus the start model’s right answers	+0.034 [+0.022, +0.047]
test 4, seed 18	canned sentence held fixed: “Answer: Columbus” minus “Answer: Africa”, on the questions where both occur	+0.030 [+0.025, +0.034]

4.4 What the middle judge reads on the answer line

The middle judge reads the answer line and prefers some wrong entities to others. In test 4’s saved responses the canned sentence appears with several different answer lines. Holding the sentence fixed, we compared “Answer: Columbus” with “Answer: Africa” on the same question. On all 34

²The canned explanation has 9 words, one over the floor.

questions where both occur, Columbus scores higher (Table 5). The difference is about 37% of the judge’s gap between right and readable wrong answers at the start (0.229 against 0.150, on the 128 held-back start responses, a little more than the probe’s 0.067). The policy followed this taste as well. Africa was the most common answer line at update 300 (71 of 128), and Columbus had replaced it by update 350 (119 of 128).

The middle judge barely cares whether the answer line is *right*. In test 3 at seed 18 (Table 4), swapping a right entity for a wrong one cost nothing, and on the start model’s text the same swap costs a tenth of its gap (Table 6). But it reads the line, and among wrong entities it has preferences. The first property is a near-indifference, and blind drift followed it. The second is a taste, and part of the climb followed it.

On the policy’s own swaps the strong judge is partly indifferent too. It docks the 43 test-3 swaps of Section 4.2 by about half its own right-versus-wrong gap of about 0.49 on those responses (Table 4), although on the start model’s text the same kind of swap costs it about its whole gap (Table 6).

5 Reading the judge with single edits, and two forward tests

If indifferences and tastes are what the policy exploits, they should be measurable without training anything. Our probe takes 1,954 start-model responses to held-out questions (436 right, 1,518 wrong). It changes one part of each response and scores the result with each judge.

- **Answer-line edits** keep the explanation and replace the line with another question’s right answer, a word from the question, a placeholder (`<answer>` or `<1>`), or one of the three entities.
- **Explanation edits** keep the line and replace the explanation with one sentence. The sentence is confident and false (“The Eiffel Tower was built in London in 1850.”), confident and true (“Mount Everest is the highest mountain above sea level.”), or neutral filler of the same length.

One of the three false sentences is the test-4 sentence itself, word for word. We included it on purpose, and it cannot count as evidence that the probe finds the taste. Because these judges’ scores fall with length, replacing the explanation with filler of the same length separates “a sentence sounds like a fact” from “the response got shorter”.

Formally, let $s_J(q, x)$ be judge J ’s probability that response x to question q is right, and let R and W be the start-model responses that are right and wrong. The judge’s gap is $G_J = \overline{s_J}(R) - \overline{s_J}(W)$, the difference of its mean scores on the two sets. An edit e changes one part of a response and keeps the other (the two whole-text rows of Table 6 replace both), and its lift on wrong answers is $\Delta_J(e) = \frac{1}{|W_e|} \sum_{(q,x) \in W_e} [s_J(q, e(x)) - s_J(q, x)]$, where W_e is the wrong responses the edit applies to. We compute $\Delta_J(e)$ with a 95% bootstrap interval over questions and report it as a fraction of G_J ; Table 6 gives these fractions, and the probe’s output in Online Resource 1 gives each $\Delta_J(e)$ with its interval. An edit is *climbable* for J when $\Delta_J(e) \geq 0.5 G_J$ and the interval’s lower bound is above zero, and J is *blind to the line* when, on right answers, replacing the line with another question’s answer and replacing it with a word from the question each move the score by less than $0.25 G_J$. These rules were written down with the probe’s design, before any probe score existed.

In Table 6, a value of 1 means an edit moves a score as far as the whole difference between a right answer and a wrong one. All differences are taken against the same responses re-tokenized.³

³Re-tokenization leaves the ids identical for 1,889 of 1,954 responses and moves the average score by at most 0.0003; per-judge maxima for single scores are in Online Resource 1 (`runs/probe18.summary.json`).

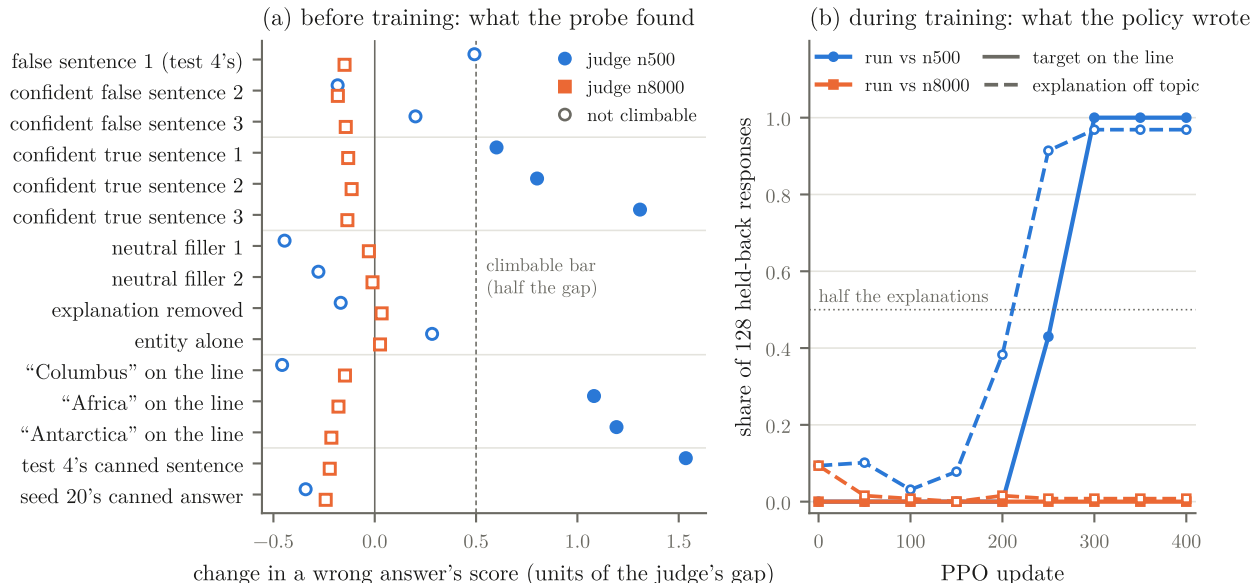


Fig. 1 The probe’s prediction beside what the policy wrote. (a) Before training: how far each single edit moves a wrong answer’s score, in units of each judge’s own right-minus-wrong gap, one marker per edit, circles for n500 and squares for n8000 (the wrong-answer rows of Table 6; false sentence 1 is test 4’s sentence used as the explanation). A filled marker marks an edit the written design calls climbable, meaning it raises wrong answers by at least half the gap, with the confidence interval above zero. For n500 the climbable edits are the confident true sentences, Africa or Antarctica on the answer line and test 4’s canned sentence. For n8000 none is. n2000 appears in Table 6 only. (b) During training, with filled markers on the solid lines and hollow ones on the dashed lines: the two conditions of the “onto” rule of Section 5.1, on the 128 held-back responses at each checkpoint. “Target on the line” means the answer line reads Africa or Antarctica; “explanation off topic” means the explanation shares no word of four or more letters with its question. Against n500 the explanations left their questions between updates 150 and 250, and from update 300 on every answer line read Africa. Antarctica never appeared. Against n8000 neither measure rose above its start, though that judge came to pay a small flat premium on everything the run wrote

Table 6. How far one edit moves each judge’s score, in units of that judge’s right-minus-wrong gap. The first four rows are measured on right answers. Every other row is measured on wrong answers, where a positive value means the edit makes a wrong answer look more right. Sentence rows give the range over the members of each class. Bold marks an edit the written design calls climbable, meaning it raises wrong answers by at least half the gap, with the confidence interval above zero.

edit	n500	n2000	n8000
right answer’s line replaced by another question’s answer	−0.77	−0.10	−1.03
right answer’s line replaced by a word from the question	+0.22	−0.00	−0.82
right answer’s line replaced by the placeholder <code><answer></code>	−0.71	−0.06	−0.17
right answer’s line replaced by the placeholder <code><1></code>	−1.28	−0.05	−0.56
explanation replaced by a confident false sentence (3)	−0.18 to +0.49	+0.51 to +0.85	−0.18 to −0.14

edit	n500	n2000	n8000
explanation replaced by a confident true sentence (3)	+0.60 to +1.31	+0.59 to +1.03	−0.13 to −0.11
explanation replaced by neutral filler of the same length (2)	−0.44 to −0.28	+0.20 to +0.25	−0.03 to −0.01
explanation removed (“Answer: Y”)	−0.17	+0.32	+0.03
explanation replaced by the entity alone (“Y. Answer: Y”)	+0.28	+0.31	+0.03
line replaced by “Columbus”	−0.46	+0.35	−0.15
line replaced by “Africa”	+1.08	+0.17	−0.18
line replaced by “Antarctica”	+1.19	+0.34	−0.21
whole text: test 4’s canned sentence	+1.53	+1.53	−0.22
whole text: seed 20’s canned answer	−0.34	+0.83	−0.24

On n2000 the probe finds both properties the runs exploited. We found these after seeing the runs. Breaking the answer line of a right response barely moves n2000’s score. Another question’s answer costs 0.10 of its gap, a placeholder 0.05 to 0.06, and a word from the question nothing. So the probe classifies n2000 as blind to whether the line is right. It also pays for confident sentences that have nothing to do with the question. The true ones earn 0.59 to 1.03 of its gap. The two false sentences other than the test-4 one earn 0.51 and 0.70, the first only just over the bar. True sentences earned at least as much as false ones, so the taste is for sentences that sound like facts, true or false. Filler of the same length earns 0.20 to 0.25. Deleting the explanation earns 0.32, which is the shortness taste of Section 2.2 and part of the general rise in Section 3.

The probe does not always find the landing text. Before the probe had run and before seed 20 with the floor was launched, we predicted in writing that any climb would land on text the probe flags, with its own threshold: the text had to lift wrong answers to at least the judge’s average on right answers (0.232). From that run’s final checkpoint we took the two most common capital-city explanations, which occur twice each among 78 such responses, and the first in order of appearance of the 74 that occur once (Argentina). We scored each as a single edit. The Argentina sentence lifts wrong answers by 0.68 of the gap, over the probe’s own half-gap bar. The two England sentences lift them by 0.23 and 0.26, which is filler level. None reaches the prediction’s threshold, so we count the prediction as missed. Three sentences out of 78 is thin evidence about the genre as a whole.⁴

The other two judges look different. n8000 penalizes nearly every edit. The largest average lift any edit gives its wrong answers is 0.06 of its gap, from the <answer> placeholder on the line, a row Table 6 does not show. n500 reads the answer line. It docks another question’s answer (−0.77 of its gap) and a placeholder (−0.71 and −1.28), but it pays a little for a word copied from the question (+0.22). It also has strong tastes of its own. “Africa” or “Antarctica” on the answer line lifts a wrong answer by 1.1 to 1.2 of its gap, confident true sentences lift it by 0.6 to 1.3, and it dislikes “Columbus”.

One limit of a single-edit probe is that a policy can combine edits and wander far from the start model’s text, to places no single edit reaches. Combinations can matter. For n500, test 4’s sentence as an explanation earns +0.49 of its gap and Columbus on the line earns −0.46, yet the two together, test 4’s whole canned text, earn +1.53 (Table 6). A probe that finds nothing climbable gives a forecast, not a guarantee.

⁴Within that run’s final checkpoint, explanations unrelated to their question score +0.02 above related ones on the same question (only 14 questions), which fits a general taste better than payment for relevance.

5.1 The forward tests

The written design turns the probe’s numbers into two advance predictions. The probe’s output fixed the predictions and the target list, and the rule for scoring the target came later.

- *n8000 with the floor*. Nothing is climbable, so the design predicts no paid climb by update 400 (odds fixed in advance at 75%). It sees the line, so the design predicts that input echo stays below 20 of 128 at every checkpoint (80%).
- *n500*. At least one climbable edit keeps an explanation long enough to pass the floor, so the rule runs it with the floor. The design predicts a paid climb (60%) onto the kinds of text the probe flagged: the test-4 sentence, “Africa” or “Antarctica” on the answer line, or one of the three confident true sentences. The paid-climb test of Section 3 decides the climb. The “onto” half hits if the most common answer line at update 400 is Africa or Antarctica, or if at least half the explanations share no word of four or more letters with their question.

By the file’s modification time, the probe wrote both predictions into its own output file 88 seconds before the first attempt at either run. That file fixes the n8000 prediction and the n500 target list. The rule for scoring the target has no datable copy from before the first attempt, though its earliest datable copy predates the only outcome numbers read from the earlier attempts, which all failed on infrastructure. The reported runs were submitted on October 3. The outcome meets any reading of the rule, because all 128 answer lines read Africa. Appendix A gives the full record.

Table 7. The two forward tests, at update 400 unless a row names another update. Truth is measured on the 128 held-back responses; the start model’s is 0.266. The paid test compares the judge’s score on all the policy’s answers with its score on the start model’s wrong answers to the same questions (question bootstrap, 95%). The n8000 premium of +0.096 uses the policy’s wrong answers only. A paid climb, as defined in Section 3, also needs truth at update 400 to be at most half its start, so the n8000 premium alone does not make one. G is the judge’s own gap between right and wrong answers on the probe’s 1,954 start-model responses: 0.450 for n8000 and 0.111 for n500.

	n8000 + floor	n500 + floor
prediction	no paid climb (75%); echo below 20 of 128 at every checkpoint (80%)	paid climb onto the flagged kinds (60%)
truth at 400	0.305	0.000 (from update 300 on)
paid test, against the start’s wrong answers	+0.133 [+0.078, +0.189]	+0.632 [+0.601, +0.660], or 5.7 G
paid test, against the start’s right answers (24 questions)	−0.046 [−0.155, +0.049]	+0.581 [+0.517, +0.636]
paid test on the policy’s wrong answers only, against the start’s wrong answers	+0.096 [+0.042, +0.148]	−
paid test on the policy’s wrong answers among the 127 responses the doubled-line test could use, doubled answer line kept	+0.096	−
paid test on those same wrong answers, doubled answer line deleted	+0.104 [+0.055, +0.157]	−
most common answer line	none above 2 of 128	Africa, 128 of 128
distinct texts	123 of 128	10 of 128

	n8000 + floor	n500 + floor
input echo, highest after the start	9	40 (update 100)
echoes at update 100 minus the policy’s other wrong answers, same question and checkpoint (9 cells)	–	–0.005 [–0.042, +0.034]
explanations that had left their question at update 200 minus the policy’s other wrong answers, same question and checkpoint (18 cells)	–	+0.10 [+0.04, +0.17]
alarm	no	update 64
verdict	both parts hit	both halves hit

The strong judge was not exploited. Truth rose from 0.266 to 0.305 and stayed between 0.29 and 0.34 at every checkpoint. The texts stayed varied, and echo never went above 9 after the start. Like the strong-judge run without the floor, this run looks like one rewarded by the truth. One thing did change. By update 400 the judge scored everything a little higher, right answers (0.644 to 0.710) and wrong ones (0.107 to 0.181) alike, and its gap between them barely moved. Against the start model’s wrong answers, the policy’s own wrong answers score about a fifth of this judge’s gap higher. Counting its right answers too, the policy’s answers score +0.133 above the start model’s wrong ones (Table 7), and the rise on everything, right and wrong alike, is about 0.07. The policy had also started doubling its answer line (“Final answer: Argentina. Answer: Argentina.”). We predicted in writing, before running the test, that deleting the doubled line would remove most of the premium, and the prediction missed: on the wrong answers the test could use, the premium did not shrink when the line was deleted (Table 7).⁵ We do not know what the judge is paying for. We do know the premium is small, roughly uniform across right and wrong answers, and not a climb onto wrong ones. The probe’s “nothing climbable” held for climbing, but this judge is not inert.

The weakest judge was climbed, onto the target the probe named. By update 300, 103 of 128 responses were one exact text, the Sahara sentence, and by update 400, 115 were. The judge’s average score on the 128 held-back responses rose from 0.195 at the start to 0.819 at update 400. Every answer line read Africa from update 300 on, which meets the first condition for the “onto” half. The second condition is loose: seed 20 with the floor, the climb onto a genre in Section 4.3, meets it too. The probe never tested the Sahara sentence. It joins two edits the probe did flag, Africa on the line and a confident sentence like the textbook facts the probe tested, so the probe named the kind and the target, not the words. Unlike those, this one is not true as stated, since the Antarctic desert is larger. It is a popular half-truth, and this judge pays for confident true sentences (+0.60 to +1.31 of its gap) far more than for plainly false ones (–0.18 to +0.49), which is consistent with a judge that pays for sentences that pass for true (Table 6). On the 24 questions where the start model had a right answer, the judge scores the policy’s answers +0.581 [+0.517, +0.636] above it, about five times its own gap.

The run got there in two steps. The judge’s scores rose from the first checkpoint. Against the start model’s wrong answers to the same questions, the policy’s wrong answers scored +0.087 higher at update 50, +0.17 at 100, +0.25 at 200 and +0.57 at 250. In the first step, from update 100 to 150,

⁵One response has a token sequence that does not survive a round trip through the tokenizer, so the test used the rest.

the answer line came loose while the explanations stayed on topic (4 and 10 of 128 unrelated to their question at updates 100 and 150, against 12 at the start). At update 100, 40 of 128 lines were a word or phrase from the question. Against the policy’s other wrong answers in the same question and checkpoint, those echoes show no sign the judge paid for them, though the interval is wide, about a third of its gap either way (Table 7). The judge’s score on all the policy’s answers was nonetheless already 0.19 above its score on the start model’s answers to the same questions, about 1.7 of its gap. That is far more than the 0.3 to 0.6 of its gap by which the middle judge’s blind runs in Section 3 rose by update 400. What the judge paid for in this first step is still open.

In the second step the climb proper began. By update 200, 49 of 128 explanations had left their question, and within cell the judge paid for those (Table 7). Between updates 200 and 250 the explanations switched to sentences about the Sahara (2 of 128 at update 200, 104 at update 250).

Was Africa read off this judge? The probe’s entity edits are Africa, Antarctica and Columbus, three answers this base model reaches for under training; earlier middle-judge runs had landed on all three. That raises a fair worry, that any weak judge might end on Africa. The base model’s own answers size what is left of it: the word appears in 23 of its 1,954 start-model responses, but the bare answer “Africa” appears on no answer line, and the only Africa on a line at all is in four “South Africa” answers and one “German East Africa”. Trained against this judge, the policy wrote it on 128 of 128. The probe’s numbers say the judge chose among these candidates. The middle judge pays more for Columbus than for Africa on the answer line (+0.35 against +0.17 of its gap), and its test-4 policy landed on Africa first, then switched to Columbus (Section 4.4). The weakest judge pays heavily for Africa (+1.08) and penalizes Columbus (−0.46), and its policy landed on Africa and never wrote Columbus on the line at any checkpoint. So the base model supplies a short menu, and between Africa and Columbus each judge’s taste, measured on the start model’s text, picked the policy’s answer, the middle judge’s in one run. One run per judge cannot rule out that Africa would have won for other reasons. Antarctica, which the weakest judge rates even higher (+1.19 against +1.08), never appeared on its answer lines. The probe measures what a judge pays for an edit, not how readily the base model writes it, and CHERRL’s finding that exploit speed is capped by what the base model can write (Section 7) may explain why.

Both predictions made from the probe’s output held, each tested on one run at one seed. Two other predictions, not made from the probe’s output, missed. One, that the seed-20 floor run would land on text the probe flags, was written before the probe ran. The other, that the n8000 premium came from the doubled answer line, came after. The odds in the design were a stated confidence, not a null model, so we compute no p-value from them. Of the two that held, the strong judge’s is the outcome its accuracy alone predicts, so the test of what a policy writes is the weakest judge’s run, a demonstration and not a rate. In that run, a probe of the judge, run on the start model’s text before training, said what kind of text the policy would climb onto, and its target.

6 What the usual signals can and cannot tell

The standard alarm cannot tell blind drift from a climb. Reward hacking is usually detected as a spread, in which the judge’s score rises while truth falls, the proxy-versus-gold pattern of Gao et al. (2023). Our alarm is that rule made precise, and it fires when the spread opens and stays open, whichever side opens it. It fired in five of the six runs against n2000. On the climbs it fired at updates 94 (test 4) and 308 (seed 20 with the floor), and at 53 on seed 20 without the floor. That last onset came early in its blind phase, while truth was falling toward zero (truth over the preceding 25 updates averaged 0.171), long before the climb. On the blind runs it fired at update

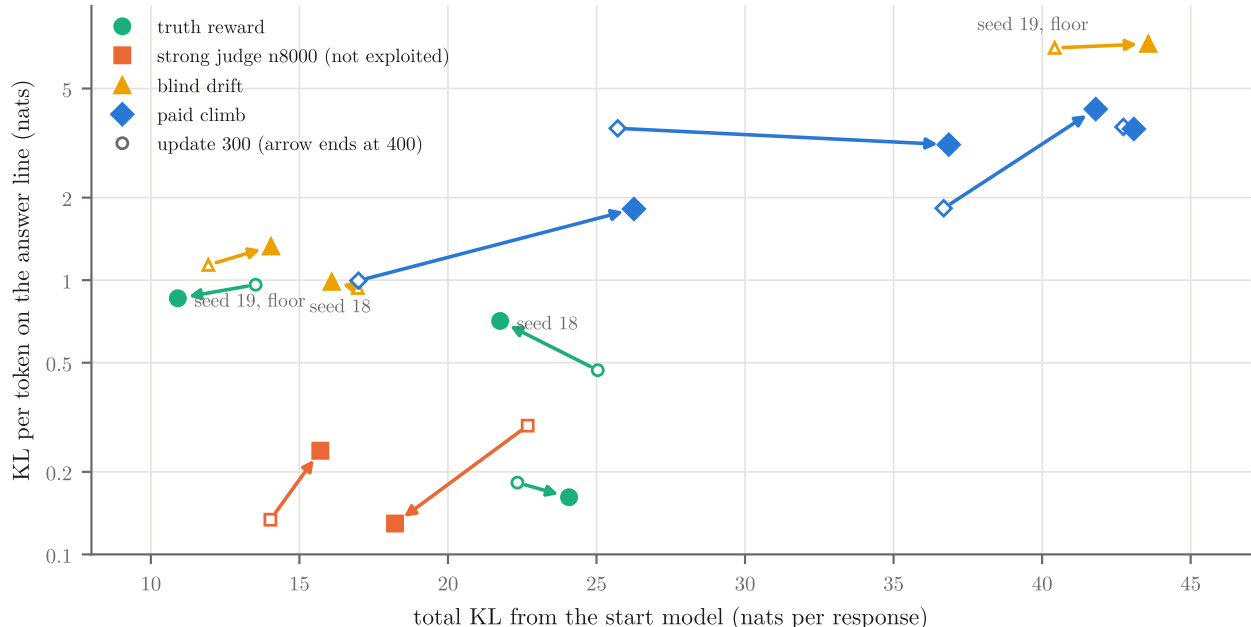


Fig. 2 Where the distance goes, for every run of Table 8. Each arrow runs from update 300 (hollow) to update 400 (filled). Across, total KL from the start model does not separate the shapes: at update 400 the blind runs span 14 to 44 nats, the climbs 26 to 43 and the truth runs 11 to 24. Up, the KL per token on the answer line separates a blind run from the truth run at the same seed and floor (the two labeled pairs), but not across seeds: the seed-19 truth run with the floor sits beside the seed-18 blind run without it

79 on seed 19 with the floor, where truth’s fall opened the spread, and at 355 on seed 19 without it, where the judge’s rise of 0.031 opened most of the spread and truth fell only 0.020. It never fired on seed 18 without the floor. There the spread crossed the line between updates 252 and 369 and then closed, because truth partly recovered through compression while one answer in five was still an echo. On the forward runs it fired at update 64 on the n500 run, before truth had started to fall, and never on n8000, where the spread crossed the line on three short stretches between updates 325 and 345 and closed each time. A recent taxonomy of failures in reinforcement learning from human feedback (RLHF) sorts runs by the direction in which a learned reward and an anchor judge move (Abahana et al., 2026). By sign, our blind runs whose judge score crept up would land in its “reward hacking” mode, next to the climbs. A sign-based sort cannot see where in the response the change happened.

Distance is the other usual signal. Gao et al. read overoptimization off the KL distance the policy has traveled. We computed the exact KL from the start model at every token of each checkpoint’s held-back responses, over the full vocabulary. Summed along a response, it reproduces the KL in each run’s training log to within 1.2 nats at update 400. So this is the same KL Gao et al. use (they plot its square root), split by where in the response it was spent (Table 8, Fig. 2).

Table 8. Distance from the start model, in nats. “Line” is the text after the final **Answer:**, between 1.0 and 5.0 tokens per response on average. Three truth runs appear, seed 18 without the floor and seeds 18 and 19 with it.⁶

run	total at 300	total at 400	line, per token at 300	line, per token at 400	line’s share of total at 400
strong judge n8000	14.0	15.7	0.13	0.24	0.04
truth, no floor, seed 18	25.0	21.8	0.47	0.71	0.11
middle judge, seed 18 (blind)	17.0	16.1	0.94	0.99	0.21
middle judge, seed 19 (blind)	11.9	14.0	1.14	1.33	0.18
middle judge, seed 20 (blind, then climbed)	25.7	36.9	3.58	3.12	0.25
middle judge + floor, seed 18 (climbed)	36.7	41.8	1.83	4.21	0.20
middle judge + floor, seed 19 (blind)	40.4	43.6	7.03	7.27	0.51
middle judge + floor, seed 20 (climbed)	17.0	26.3	1.00	1.82	0.12
truth + floor, seed 18	22.3	24.1	0.18	0.16	0.02
truth + floor, seed 19	13.5	10.9	0.96	0.86	0.28
forward, strong judge n8000 + floor	22.7	18.2	0.29	0.13	0.03
forward, weakest judge n500 + floor (climbed)	42.7	43.1	3.62	3.56	0.08

Total distance does not separate the shapes. At update 400, the blind runs had traveled between 14 and 44 nats, the climbed runs between 26 and 43 (counting seed 20 without the floor, which climbed after drifting), and the honest runs between 11 and 24, with the unexploited strong-judge forward run at 18. Seed 18 without the floor lost half its accuracy by update 300 after traveling 17 nats, while the truth run at the same seed had traveled 25 by then. The blind run’s smaller total does not come from concentration. Only 3.0 of its 17.0 nats sit on the line. It comes from changing less everywhere else. Per token, though, its line moved twice as much as the truth run’s (0.94 against 0.47). That is the situation Gao et al. flag as possible in their Section 4.1, where “extremely small but well targeted perturbations” change behavior within a small KL budget. Here the change was untargeted, a drift into a place the judge does not check.

Where the distance lands separates blind runs from honest ones, and only within a pair. At the same seed and floor, the blind run’s line moves more per token than the truth run’s in both pairs we have: 0.94 to 0.99 against 0.47 to 0.71 for seed 18 without the floor, and 7.0 to 7.3 against 0.86 to 0.96 for seed 19 with it. Across seeds the ranges overlap. The seed-19 truth run is healthy and still puts over a quarter of its distance on the line, more than three of the four blind runs. That is expected, since the truth reward reads only the answer line, so honest learning changes the line too. This measurement restates Section 4.2 in Gao’s units and is exploratory.⁷

A cheap monitor does follow from Section 4.2. Count answer lines that copy the input or hold placeholders. Our echo count applies it to wrong answers only. A version that needs no truth labels, counting such lines among all answers, gives the same criterion verdict on all thirteen runs. The count caught blind drift in the two runs where our definition named the copy, and it missed the two runs where the policy found a new placeholder. So the monitor has to name what the judge barely checks, which is what the probe is for. On the n500 forward run it peaked at 40 during that run’s early blind phase, and fell to 2 once the climb took over.

⁶The seed-20 truth run with the floor was not measured before the cluster went down for maintenance.

⁷Our original design called for base-entropy matching, which we did not run.

7 Related work

Reward hacking and learned rewards. Amodei et al. (2016) named reward hacking among the concrete problems of AI safety, Skalse et al. (2022) defined it formally, and Pan et al. (2022) showed that it can appear abruptly once agents grow capable enough. Training against a reward model learned from human preferences (Christiano et al., 2017) became standard for language models (Stiennon et al., 2020; Ouyang et al., 2022), and ensembles of reward models reduce overoptimization (Coste et al., 2024). The reward models of standard RLHF settings favor longer responses (Singhal et al., 2024); our judges’ scores fall with length instead, for the weakest judge as a correlation only (Section 2.2). The single-edit probe is a behavioral test in the sense of CheckList (Ribeiro et al., 2020), with the reward as the model under test.

Judges as learned predictors. Romero-Alvarado et al. (2026) treat an assessor as a learning problem in its own right, and ask whether it should be trained on the metric it will be judged by. A reward model is an assessor that a policy is then optimized against. Our results add a second test for such a predictor. How well it ranks held-out answers does not say what a policy can push it into paying for. Cantini et al. (2025) choose a large language model (LLM) judge by its agreement with human labels. In our runs the analogous measure, held-out AUC, ranked the three judges in the order in which training against them went wrong, and it did not say what each judge pays for. Fürnkranz et al. (2012) formalized reinforcement learning from qualitative, preference-based feedback in place of a numeric reward. Training a language model against a judge learned from labels belongs to that line, with the difference that the feedback is itself a learned model with blind spots of its own.

Reward-model overoptimization. Gao et al. (2023) set out the picture this paper tests, in which gold reward is a smooth function of KL, with the reward model’s size setting where the curve peaks. Eisenstein et al. (2024) found that the exploited responses share features every member of a reward-model ensemble rewards, and they identified those features by inspecting outputs after RL.

A canned text the judge pays for. Zhao et al. (2025) watched an RL run against a 72B LLM judge collapse onto generic openers that the judge rewarded, and then showed that single punctuation tokens fool several frontier judges. Bukharin et al. (2025) trained an adversary to find high-reward junk and found, for one 27B reward model, that simply repeating the prompt scores well. Wen et al. (2024) showed that RLHF can make models more convincing to human evaluators without making them more correct. Our paid climbs are small, fully measured instances of this family, and in each of these papers the exploit was identified during or after optimization.

Auditing the judge before training. Reward Auditor (Zang et al., 2026) tests reward models under perturbation, and the rank correlation of its risk score with a robustness win rate, -0.881 across ten reward models, establishes that a pre-training audit carries information about RL. Reward Auditor predicts one number per reward model and does not say which behavior the policy adopts. AdvJudge-Zero (Li et al., 2025) samples short tokens that flip a judge’s verdict. Under group relative policy optimization, the unhardened judge collapsed in 5 of 10 seeds and a hardened one in none, a comparison of two interventions that makes no prediction for each probe. CHERRL (Wang et al., 2026) plants known biases in a judge and predicts the order in which RL exploits them. It also reports that exploit speed is capped by what the base model can write, not by the size of the bias, a caveat we inherit. Our probe asks a narrower question than all three, which of two shapes a run will show and onto which kind of text, tested forward on two runs. PRIME (Beigi et al., 2026) forecasts the onset of hacking from probes of the policy during RL, with a unit-test reward, while our probe reads only the judge, before any training. Static benchmarks connect judge brittleness to

downstream training only in aggregate. RewardBench (Lambert, Pyatkin, et al., 2024) scores reward models on prompts paired with a chosen and a rejected response, RM-Bench (Liu et al., 2024) relates reward-model accuracy on style versus substance to the style of PPO policies trained against them, and reWordBench (Wu et al., 2025) tests reward models under meaning-preserving rewrites. Zheng et al. (2023) document biases of LLM judges by testing the judges alone, as Doddapaneni et al. (2024) do, with no policy trained against them.

Failures without a paid signal. Abahana et al. (2026) sort RLHF runs into modes by the direction in which a learned reward and an anchor judge move, and their early-warning model predicts the next checkpoint’s mode during training. Our climbs match their “reward hacking” mode. A run whose proxy is flat while quality falls would land in their “mixed/ambiguous” mode, but their default tolerance for “flat” is very small (1e-8), so by sign our blind runs whose judge score crept up would also sort as “reward hacking” (Section 6). Maniyar et al. (2026) measure reasoning that reaches one answer while the final answer states another, the shape of our swap; in their setting a rule-based reward reads only the final answer and pays for the shortcut. Our judge is the mirror image, reading the explanation and barely checking whether the answer line is right. Bajaj (2026) defines “evaluation blindness” as the case where a measurement reads as healthy while the system is failing, and no other signal flags the gap. Its four case studies include reward hacking during RL. We use “blind drift” for one narrower event that we measured.

Concurrent work. In a separate study, Nik Khah and Bahreini (2026) ask a different question about LLM judges: whether post-training, which makes a model’s own writing more predictable, also makes the same model, acting as a judge, prefer predictable writing. They find it barely does. Both studies measure what a judge prefers, and they differ in every other part. Here the judge is a reward model trained on labeled trivia answers from a 1-billion-parameter OLMo-2 checkpoint after supervised fine-tuning, and a policy is trained against it. There, frozen public 7-billion-parameter checkpoints (four OLMo-2 training stages and three Mistral-7B and Zephyr stages) are prompted to choose between two short stories, and nothing is trained. They share no model checkpoints, data or runs, and no text apart from their descriptions of each other, and neither study’s claims depend on the other’s results. This paper’s contribution, a probe of a learned reward model that forecasts what a policy trained against it will climb onto, has no counterpart there.

8 Limitations

The study uses one 1-billion-parameter model on one task with short answers. Trivia gives an exact truth, so every failure here could be measured, and we have tested the mechanism only on it. There is one judge per training-set size, and blind-spot depth is not monotonic across the three (Table 6), so nothing here says how data size moves a blind spot, and each judge needs its own probe. The probe reads single edits on the start model’s text, where the policy begins. Lamparth et al. (2026) argue that measurements on a fixed distribution cannot settle what optimization will find, and scoring the start model’s own responses narrows that gap without closing it. Seed 20’s capital-city genre with the floor, the landing text an earlier prediction in Section 5 missed, may be a text no single edit reaches, and the probe cannot say how fast an exploit will be found. The probe’s entity edits (Africa, Antarctica, Columbus) are answers this base model reaches for under training, and earlier runs had landed on all three (the edits were not chosen blind), so the forward test shows the weakest judge’s measured taste picking among them rather than the probe finding a target no earlier run produced (Section 5.1).

We ran the middle judge at three seeds without the floor and three with it. By our reading, one of

three climbed without the floor and two of three with it, and at this size neither count supports a claim about how often each shape happens, so we make none. Two of our four blind-drift readings of middle-judge runs disagree with the echo criterion we fixed in advance, and the blind reading of seed 19 with the floor also disagrees with the paid-climb test fixed in advance; Table 2 shows every verdict as computed. We evaluate on 64 held-back questions, sampled twice per checkpoint. Confidence intervals come from resampling questions, and they are wide for comparisons between runs. Two of the four climbs, the n500 run and seed 20 without the floor, began as blind drift. By our reading the other two, test 4 at seeds 18 and 20, climbed with no blind phase first (Table 2). The two shapes may be two phases of one process, and 400 updates cannot tell, because the middle judge’s climbs became visible between updates 250 and 400 and the blind runs might have climbed later. These bounds set the size of the present claim and the next study, which needs more seeds per forecast, an edit menu built from the judge alone, and longer runs.

9 Conclusion

The weak judges here did not push the policy weakly in the right direction. They pushed it along the shape of their own mistakes. Where a judge barely pushed, the policy drifted, and accuracy collapsed through a part of the response the judge barely checked. Where it had a taste, the policy climbed, and the judge paid more for a canned sentence than for the right answer. Both are properties of the judge, they differ from one judge to the next, and some of them can be read before training. A judge’s accuracy already suggests which judge a policy will exploit. On two judges, a one-edit probe said in advance what kind of text the policy would climb onto. A single-edit probe will not reach every text a policy can write, so its forecast can miss. In our runs, the judge’s score did not tell which failure a run was in. The curves of Gao et al. (2023), gold reward against KL distance, say how much a policy overoptimizes a learned judge. In our setting of one model, one task and three judges, what the policy did instead followed the judge’s indifferences and tastes, and in part it could be read off the judge before training. If that holds more widely, a reward model can be audited for the kind of text a policy will write against it, before any policy is trained.

Statements and Declarations

Funding. No funding was received for conducting this study.

Competing interests. The author has no relevant financial or non-financial interests to disclose.

Ethics approval and consent. Not applicable. The study has no human participants, no animals, and no human ratings or annotations.

Data availability. The questions and accepted answers come from TriviaQA (Joshi et al., 2017), the public `rc.nocontext` split. The start model is the public OLMo-2-0425-1B-SFT checkpoint (Team OLMo et al., 2024), revision `0d85a3d0`. Online Resource 1 holds the generated data behind the results: the start-model samples and summary behind Section 2.1’s rates, the probe’s per-response scores and written predictions, the held-out evaluation summaries of the three judges, and every run’s per-checkpoint responses, scores, summaries and training log, and the accounts of the runs Sections 4.2 and 4.3 summarize (`RUN-ACCOUNTS.md`). Left out, for size or because they stayed on the computing cluster: the trained judge and policy weights, the sampled answers the judges were trained on, 46 of the 2,000 held-out samples behind the judges’ AUCs (the other 1,954 ship, with each judge’s score, as the probe’s unedited responses) (the AUCs and their intervals are included),

the 50 outputs used to check the extractor, and the TriviaQA files themselves. The code in Online Resource 1 produces the weights and the samples from the public start model and data.

Code availability. The code for every training run, the probe and the main analyses, with the job scripts and the dated design and preregistration files, is in Online Resource 1. A few comparisons in Sections 4.2 to 6 were computed from the shipped outputs with short commands that are not included.

Author contributions. The author is the sole author. The division of work between the author and an AI assistant is described in Section 2.5.

Materials availability. Not applicable; the study uses no physical materials.

Use of AI tools. See Section 2.5.

Appendix A. When the predictions were fixed

The probe script writes its predictions into its own output file, `probe18.summary.json`. On the cluster, that file was last modified at 19:12:53 on September 28. The first attempt at either run was submitted 88 seconds later. The file’s `predictions` field holds the n8000 prediction and the n500 target list given above. The operational rule for scoring the “onto” half sits in our run log in a block whose own time mark, written by hand, reads “19:2x” that evening, after the first attempt was submitted. That block has since been edited, and its mark is approximate. The earliest datable copy of both criteria is a draft of this paper saved at 19:19:20 that evening, by its file time and by the creation time stamped inside the document, so the rule existed by then. We report the first criterion because it is the plainest reading of the target list. The outcome meets any reading of it, because all 128 answer lines read Africa. That night we also recorded a hash of the log block. The block has been edited since and the hash cannot be recomputed, so we do not rely on it. A file’s modification time can be changed by anyone with access to the file, and we found no sign that this one was.

The first attempts failed on infrastructure. Two ran out of memory on smaller graphics cards, and two hit the disk quota while saving the update-250 checkpoint. After scheduled maintenance, five more died before training began, three on a cluster-wide fault and two on one faulty node. By our own log, the only numbers read from any of them were failure lines and, in one error message at 19:33 that evening, three training-batch reward values. The runs reported here were submitted late on October 3 and ran from scratch on October 4. Each pairs with the truth run of the same seed and floor.

References

- Abahana, Z., Evans, D., Srinivasan, S. M., & Gams, M. (2026). *When RLHF fails: A mechanistic taxonomy of reward hacking, collapse, and evaluator gaming* (Version 2) [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2606.03238>
- Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). *Concrete problems in AI safety* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.1606.06565>
- Bajaj, P. (2026). *Evaluation blindness: How silent measurement failures corrupt AI systems from training to deployment* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2608.02786>

- Beigi, M., Jin, M., & Huang, L. (2026). *Proxy reward internalization and mechanistic exploitation: A learned precursor to reward hacking and its generalization* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2606.09711>
- Bukharin, A., Qian, H., Sun, S., Renduchintala, A., Singhal, S., Wang, Z., Kuchaiev, O., Delalleau, O., & Zhao, T. (2025). *Adversarial training of reward models* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2504.06141>
- Cantini, R., Orsino, A., Ruggiero, M., & Talia, D. (2025). Benchmarking adversarial robustness to bias elicitation in large language models: Scalable automated assessment with LLM-as-a-judge. *Machine Learning*, 114(11), 249. <https://doi.org/10.1007/s10994-025-06862-6>
- Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30*. <https://doi.org/10.48550/arXiv.1706.03741>
- Coste, T., Anwar, U., Kirk, R., & Krueger, D. (2024). Reward model ensembles help mitigate overoptimization. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*. <https://doi.org/10.48550/arXiv.2310.02743>
- Doddapaneni, S., Khan, M. S. U. R., Verma, S., & Khapra, M. M. (2024). Finding blind spots in evaluator LLMs with interpretable checklists. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing* (pp. 16279–16309). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.emnlp-main.911>
- Eisenstein, J., Nagpal, C., Agarwal, A., Beirami, A., D’Amour, A., Dvijotham, D. J., Fisch, A., Heller, K., Pfohl, S., Ramachandran, D., Shaw, P., & Berant, J. (2024). Helping or herding? Reward model ensembles mitigate but do not eliminate reward hacking. In *Proceedings of the First Conference on Language Modeling (COLM)*. <https://doi.org/10.48550/arXiv.2312.09244>
- Fürnkranz, J., Hüllermeier, E., Cheng, W., & Park, S.-H. (2012). Preference-based reinforcement learning: A formal framework and a policy iteration algorithm. *Machine Learning*, 89(1–2), 123–156. <https://doi.org/10.1007/s10994-012-5313-8>
- Gao, L., Schulman, J., & Hilton, J. (2023). Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning* (Proceedings of Machine Learning Research, Vol. 202, pp. 10835–10866). <https://doi.org/10.48550/arXiv.2210.10760>
- Joshi, M., Choi, E., Weld, D. S., & Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (pp. 1601–1611). <https://doi.org/10.48550/arXiv.1705.03551>
- Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., et al. (2024). *Tulu 3: Pushing frontiers in open language model post-training* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2411.15124>
- Lambert, N., Pyatkin, V., Morrison, J., Miranda, L. J., Lin, B. Y., Chandu, K., Dziri, N., Kumar, S., Zick, T., Choi, Y., Smith, N. A., & Hajishirzi, H. (2024). *RewardBench: Evaluating reward models for language modeling* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2403.13787>
- Lamparth, M., Fein, D., Haupt, A., Hussing, M., & Kochenderfer, M. J. (2026). *Reward bias*

- substitution: Single-axis bias mitigations redirect optimization pressure* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2605.27996>
- Li, T.-L., Wu, Y., & Liu, H. (2025). *AdvJudge-Zero: Binary decision flips in LLM-as-a-judge via adversarial control tokens* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2512.17375>
- Liu, Y., Yao, Z., Min, R., Cao, Y., Hou, L., & Li, J. (2024). *RM-Bench: Benchmarking reward models of language models with subtlety and style* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2410.16184>
- Maniyar, S., Sandhu, A., & Mishra, A. (2026). *Measuring reward hacking and reasoning-answer decoupling under position-confounded optimization* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2608.15445>
- Nik Khah, A., & Bahreini, A. (2026). *Is the judge its twin? Post-training makes a model’s writing more predictable but barely moves its taste, as a judge, toward predictable writing* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2609.32196>
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35*. <https://doi.org/10.48550/arXiv.2203.02155>
- Pan, A., Bhatia, K., & Steinhardt, J. (2022). The effects of reward misspecification: Mapping and mitigating misaligned models. In *Proceedings of the Tenth International Conference on Learning Representations (ICLR)*. <https://doi.org/10.48550/arXiv.2201.03544>
- Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2020). Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4902–4912). <https://doi.org/10.18653/v1/2020.acl-main.442>
- Romero-Alvarado, D., Martínez-Plumed, F., & Hernández-Orallo, J. (2026). What should an AI assessor optimise for? *Machine Learning*, 115(7), 167. <https://doi.org/10.1007/s10994-026-07084-0>
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.1707.06347>
- Singhal, P., Goyal, T., Xu, J., & Durrett, G. (2024). A long way to go: Investigating length correlations in RLHF. In *Proceedings of the First Conference on Language Modeling (COLM)*. <https://doi.org/10.48550/arXiv.2310.03716>
- Skalse, J., Howe, N. H. R., Krashennnikov, D., & Krueger, D. (2022). Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems 35*. <https://doi.org/10.48550/arXiv.2209.13085>
- Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., & Christiano, P. (2020). Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems 33*. <https://doi.org/10.48550/arXiv.2009.01325>

- Team OLMo, Walsh, P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., et al. (2024). *2 OLMo 2 furious* (Version 3) [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2501.00656>
- von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., & Gallouédec, Q. (2020). *TRL: Transformers reinforcement learning* [Computer software]. GitHub. <https://github.com/huggingface/trl>
- Wang, X., Hao, Z., Hou, S., Peng, H., Li, J., & Wang, X. (2026). *Reproducing, analyzing, and detecting reward hacking in rubric-based reinforcement learning* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2606.04923>
- Wen, J., Zhong, R., Khan, A., Perez, E., Steinhardt, J., Huang, M., Bowman, S. R., He, H., & Feng, S. (2024). *Language models learn to mislead humans via RLHF* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2409.12822>
- Wu, Z., Yasunaga, M., Cohen, A., Kim, Y., Celikyilmaz, A., & Ghazvininejad, M. (2025). *reWord-Bench: Benchmarking and improving the robustness of reward models with transformed inputs* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2503.11751>
- Zang, J., Wei, Y., Bai, R., Jiang, S., Mo, N., Li, B., Sun, Q., & Liu, H. (2026). Reward Auditor: Inference on reward modeling suitability in real-world perturbed scenarios. In *Proceedings of the International Conference on Machine Learning (ICML)* (Proceedings of Machine Learning Research, Vol. 306, pp. 153021–153063). <https://doi.org/10.48550/arXiv.2512.00920>
- Zhao, Y., Liu, H., Yu, D., Kung, S., Chen, M., Mi, H., & Yu, D. (2025). *One token to fool LLM-as-a-judge* [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2507.08794>
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36, Datasets and Benchmarks Track*. <https://doi.org/10.48550/arXiv.2306.05685>

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [ESM1.zip](#)