META-RUBRICS: LEARNING TO DISCOVER RUBRICS AT DEPLOYMENT FOR EFFICIENT TASK ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents are increasingly deployed in open-ended environments where task objectives are under-specified and no task-specific evaluator is available. Reinforcement learning (RL) and inference-time search have driven major gains in verifiable domains such as math and coding, but these gains rely on a verification–generation (VG) gap: checking a solution is easier than producing one. In open-ended tasks, however, this gap cannot be assumed, because a verifier must first infer what constitutes success. This raises a central question: can an agent can construct or widen such a gap at deployment time from a small amount of environment interaction and use this gap to aid policy adaptation? We formulate this as a reward discovery setting and propose a framework, `Meta-Rubrics`, which infers a reward function, represented as a natural-language rubric, which can accurately assign credit to guide policy behavior at deployment time. In this framework, a rubric generator infers task-specific evaluation criteria, and an LLM judge applies these criteria to produce scalar scores and criterion-level feedback for candidate trajectories. The generator and judge are jointly optimized with RL over branched rollouts so that induced scores recover oracle reward orderings on held-out trajectories. At deployment time, this inferred reward can rank candidate responses or supply a signal for lightweight policy optimization with RL. We evaluate `Meta-Rubrics` on user simulation in Human-LM and curating data for a language model for an unseen task with a fixed fine-tuning budget. In both settings, `Meta-Rubrics` outperforms the highest non-oracle adaptation approaches such as in-context learning, evolutionary search, and rubric-based fine-tuning baselines.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed in environments where they must adapt to tasks, interfaces, and objectives that were not encountered during training. In these settings, their ability to generalize has been largely driven by web-scale pre-training and diverse, high-quality post-training data (Bai et al., 2026). This has been supplemented with models leveraging inference-time computation (OpenAI, 2024), which has enabled these models to utilize the inference budget to chain longer and more sophisticated behaviors (Setlur et al., 2025) that can be used to adapt to an unseen task. However, this form of adaptation is effective only when the target task is well represented in the training distribution or explicitly specified at test time (Zahavy, 2026). Many real-world tasks are instead open-ended: their objectives are under-specified, feedback is sparse, and no task-specific evaluator is readily available. Humans can still adapt in these settings by inferring latent objectives from their limited experience in a new environment and adapt their behavior accordingly (Allen et al., 2020). Can LLM agents similarly infer *what should be optimized* in a novel environment and then use that inferred objective to adapt at test time?

One instantiation for adaptation in these open-ended settings is to use a small budget of environment interactions with the target environment (Yuksekgonul et al., 2026), which can steer an agent to align its behavior with the underlying task. The recent family of approaches that utilize environmental interaction focuses on adapting the policy while treating the objective as fixed. One such family is in-context learning (ICL) (Brown et al., 2020), which conditions autoregressive generation on successful demonstrations supplied in context. However, ICL can require many demonstrations to be effective, and its gains in low-data regimes may be limited (Agarwal et al., 2024). A second family is evolutionary search methods (Madaan et al., 2023; Lee et al., 2025; 2026), which instead improve candidate responses using a history of environmental feedback or model-generated critiques.

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

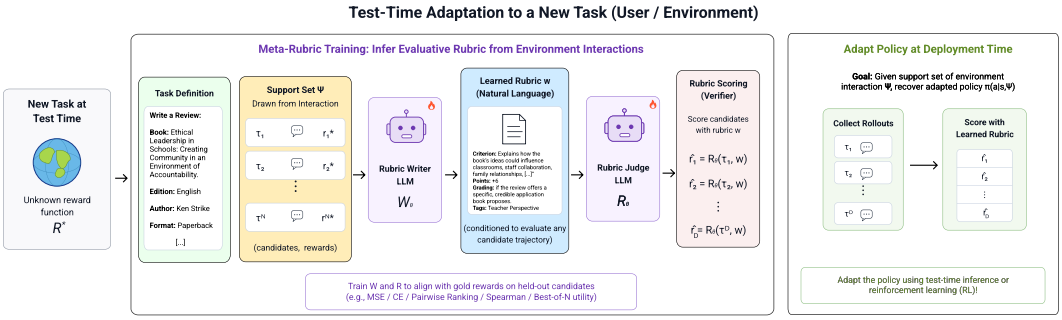

Figure 1: **Overview of `Meta-Rubrics`.** Given a small support set of interaction data for a new task, a rubric generator produces task-specific evaluation criteria. An LLM judge applies the rubric to score candidate trajectories. During training, the rubric generator and judge are optimized so that their scores agree with oracle rewards on held-out query trajectories. At test time, the resulting proxy reward guides inference-time search or lightweight reinforcement learning.

These methods can be effective when a reliable evaluation signal is available, but their performance is constrained by the quality of that signal. These methods are suboptimal when they optimize against feedback that is unavailable, poorly calibrated, or misaligned with the true objective (Song et al., 2026). Can we use this environment interaction beyond strict policy adaptation to better infer what should be optimized?

One promising alternative is to exploit the verification-generation (VG) gap (Song et al., 2025), where for several problems verifying a solution is often easier than generating one from scratch. This asymmetry has been successfully leveraged in reinforcement learning settings where a trusted verifier is available, such as mathematics. For novel and underspecified tasks, however, the VG gap may not arise naturally, because the verifier must first identify the properties that define success and assign credit appropriately to distinguish among candidate solutions. This raises a natural question: can we widen the VG gap by inferring a reward function from a small amount of support set of environment interactions from a task and then using the resulting reward signal for subsequent policy adaptation?

We introduce `Meta-Rubrics`, a framework that formulates reward discovery as a meta learning problem (Yu et al., 2019). Rather than fitting a scalar reward model from scratch for every task, an approach that can be sample-inefficient and prone to overfitting, we use the strong semantic priors of LLMs to represent the underlying reward function as a natural-language rubric (Mu et al., 2024). For each task, `Meta-Rubrics` conditions on a support set of environment interactions, such as demonstrations or reward-labeled trajectories. With this conditioning, a rubric generator produces task-specific criteria, and an LLM judge uses those criteria to assign scores to candidate trajectories. We jointly optimize the rubric generator and judge through reinforcement learning with branched rollouts and an alignment reward so that the induced scores recover the ordering imposed by the oracle reward on a held-out query set.

Once instantiated for a novel task, the discovered proxy reward can support two complementary forms of policy adaptation. First, it can guide *inference-time search* by ranking a set of candidate responses and selecting the highest-scoring one. Second, it can provide a training signal for *test-time reinforcement learning*, enabling lightweight parameter updates, such as low-rank adaptation, directly against the inferred objective. `Meta-Rubrics` disentangles the problem of discovering a useful task objective from the choice of mechanism used to optimize the policy against it.

We evaluate `Meta-Rubrics` across two adaptation settings: (1) user simulation in Amazon Reviews from the Human-LM benchmark (Wu et al., 2026) and (2) a data-curation task, where an agent is given access to a mid-training corpus, such as Nemotron Cascade SFT (Yang et al., 2026b), together with a fixed fine-tuning budget. The agent must select training examples that, when used to fine-tune a target model such as SmolLM2-135M (Allal et al., 2025), improve performance (perplexity/negative log likelihood) on a specified downstream benchmark, such as the International Law subset of MMLU-SR (Wang et al., 2024a). In both settings, `Meta-Rubrics` obtains the highest non-oracle scores among the methods we compare, including evolutionary search and parametric reward models. Its scores correlate with the oracle reward at a Spearman correlation of 0.67 in data curation and 0.42 in review generation, rising to 0.63 on augmented candidate pools.

2 Adaptation at Deployment Time via Reward Discovery

We study the problem of adaptation to unseen tasks during deployment. As a motivating example throughout this section, consider the domain of user simulation (Wu et al., 2026), where each task represents a distinct user whose unique writing style and content preferences needs to be emulated (e.g generate a review of a book on Amazon). Our objective is to adapt a model to a new task (user), given access to a sample budget of environment interactions (e.g previous user-written reviews).

Problem Setup We formalize each unseen task (user) μ at deployment time t as a token-level Markov decision process (MDP), $\mathcal{M}_\mu^t := (S, A, R_\mu^t, P)$. Here, t indexes deployment times rather than token positions; the reward may vary across deployment times but is fixed within each adaptation run. A state $s \in S$ encompasses the conditioning context (e.g., the book being reviewed) alongside the text generated thus far. An action $a \in A$ denotes the next token. The reward function $R_\mu^t(s, a)$ scores a state–action pair according to the task's preferences at time t, such as the alignment of generated text with the user's writing style. The transition dynamics function $P(s' \mid s, a)$ is deterministic, simply appending token a to the current state. Our overarching objective is to recover the optimal, user-specific policy $\pi_\mu^{t,*}$ that maximizes expected return under R_μ^t. During test-time adaptation at time t, we assume access to a small support set of environment interactions, Ψ_μ^t. This may manifest as expert demonstrations, $\{y_i\}_{i=1}^N$ where $y_i \sim \pi_\mu^{t,*}(\cdot \mid s_i)$, or as reward-labeled trajectories, $\{(y_i, R_\mu^t(y_i))\}_{i=1}^N$. Here, $R_\mu^t(y_i)$ denotes the trajectory return induced by $R_\mu^t(s, a)$. We define an adaptation policy $\pi_\mu^{t,adapt}(\cdot|s, \Psi_\mu^t)$, which conditions on this support set to predict actions in the environment. For the remainder of the paper, we fix a deployment time t and suppress its index. Our experiments evaluate this fixed-reward setting.

Towards Policy Adaptation via VG Gap Amplification In prior work, the support set facilitates policy adaptation via techniques like in-context learning or evolutionary search. However, to efficiently adapt the policy given the interaction set, these adaptation methods may not be efficient: in-context learning typically yields only marginal gains in low-data regimes (Agarwal et al., 2024), whereas evolutionary search demands repeated environment interactions to obtain feedback, limiting their applicability in domains without reliable automated evaluators (Novikov et al., 2025), which is often costly, time-consuming, or infeasible at scale (Jackson et al., 2026; Trabucco et al., 2021). Can we instead design an alternative optimization paradigm that complements direct policy adaptation?

A promising alternative is to exploit the verification-generation (VG) gap (Song et al., 2025) to steer policy adaptation toward high-reward trajectories. The VG gap underlies several forms of LLM improvement, from self-verification in chain-of-thought reasoning to reward-model-based RL. We hypothesize that environment interactions can similarly yield an exploitable gap: if a learned, environment-specific verifier is accurate, the model can use it to improve through test-time inference or reinforcement learning. To instantiate this VG gap, we can learn an environment-specific reward model $\hat{R}_\theta^\mu(s, a)$, using the environment interaction information available for each task as conditioning information. By widening this VG gap, policy adaptation becomes more tractable; the model can leverage this learned verification signal for sustained self-improvement via approaches such as test-time inference and reinforcement learning.

To study whether amplification of the VG gap can be a more efficient strategy for adaptation compared to direct policy adaptation and a population based reward function, we consider a synthetic bandit setting in Appendix A.9. In this setting, each bandit task models an individual environment (or user), and observing a subset of p reward-labeled arms reflects the support set Ψ_μ available at deployment time. By comparing a reward inference model against a direct optimal-action predictor, this synthetic benchmark explicitly tests whether inferring the reward function is more sample-efficient than direct policy inference under limited observations. In this controlled setting, reward inference achieves lower regret than direct action prediction across support sizes, which motivates our broader rubric-learning paradigm for LLM adaptation.

To recover the reward function, we frame the task as a meta-learning problem: rather than training a monolithic global reward model, we condition the model on a small support set of environment interactions (e.g., demonstrations or reward-labeled trajectories) to synthesize a reward function tailored to the target environment. Specifically, for each environment, we partition the available environment interaction data into a support (meta-train) set and a query (meta-test) set. The model leverages the support set to infer the environment-specific reward structure and is subsequently evaluated on its ability to accurately assess responses in the held-out query set. Crucially, we

parameterize this reward function as a *rubric*: a set of natural-language criteria (Gunjal et al., 2025; Arora et al., 2025) derived from the support set that effectively distinguishes between candidate responses. This choice is natural as LLMs have a strong prior for generating and utilizing rubrics, enabling a useful prior for the model for further adaptation. This rubric then serves as grounding context for either a judge model (yielding a scalar score) to guide the candidate policy during inference time or further fine-tuning. The core challenge then becomes: how can we optimize a model to write effective rubric and corresponding judge score that can be useful for future policy adaptation?

Overview of Adaptation Setting

We study adaptation to an open-ended environment where the underlying reward function is unknown. We infer the reward function in the form of a natural-language rubric derived from a set of environment interactions, which then steers the policy on new queries from that task.

3 What constitutes a good inferred rubric for adaptation?

To define an appropriate objective for rubric learning, we begin by considering the role that the inferred reward function must play in downstream policy adaptation. Predictive accuracy on held-out responses is necessary, but not sufficient: the reward function must also provide a reliable and informative optimization signal as the policy changes. We therefore identify three desiderata for an effective rubric: (1) *accurate credit assignment*, (2) *reward generalization*, and (3) a *dense reward signal*. But why are these necessary?

Accurate Credit Assignment: It is necessary for the rubric to distinguish among responses spanning a wide range of quality, including candidates that differ only along subtle stylistic or task-specific dimensions. Rather than merely separating successful responses from unsuccessful ones, the reward function should identify which properties of a response contribute to its quality and reflect these differences in its scores or feedback.

Reward Generalization: It is also important for the inferred reward to remain informative beyond the distribution of responses observed in the support set or produced by the current policy. This property is essential during adaptation because, as the policy improves, it may generate qualitatively novel responses for which the reward model has received little or no direct supervision. Without such extrapolation, reward-guided optimization may only sharpen the existing policy distribution (Huang et al., 2024) or may be overly prescriptive, picking up on spurious features from the interaction data rather than enabling genuine improvement beyond it.

Dense Reward Signal: The reward signal should provide graded information about candidate quality, rather than collapsing evaluation into a sparse or binary judgment. Dense scalar scores or targeted natural-language critiques can expose incremental improvements, reduce ambiguity in the search process, and make test-time inference or reinforcement learning substantially more sample-efficient.

For more intuition, we discuss each desideratum in a didactic example in Appendix A.10. Taken together, these desiderata characterize a reward function that is not only faithful to the available privileged information, but also useful for steering a changing policy toward increasingly high-quality behavior. How can we translate this into an objective for learning rubrics and candidate judge scores?

3.1 Exploring Reward-Learning Objectives

We now wish to define a clear optimization objective for reward learning with rubrics. As the desiderata suggest, the goal of rubric learning is not necessarily to recover the exact numerical values of the gold reward. Instead, the learned reward $\hat{r}$ should recover the *structure* of the gold reward function r^*: which responses are better (credit assignment), how candidate quality varies within a response set (dense reward), and whether these relationships remain stable as the policy changes (extrapolation). This perspective shifts the learning problem from pointwise label fitting toward functional agreement between $\hat{r}$ and r^*. Our goal is to optimize the rubric writing and judge models using reinforcement learning, defining an effective reward function for optimization.

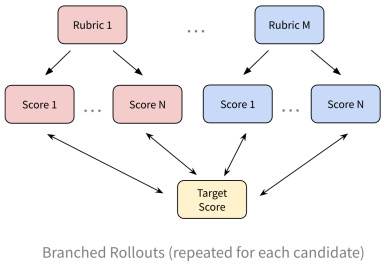

Figure 2: **Training of `Meta-Rubrics`:** Rubric writing and judge scores are generated with branched rollouts during training.

Optimization Procedure. We jointly optimize rubric generation and rubric-conditioned scoring using reinforcement learning. We formulate this process as a two-turn cooperative objective. In the first turn, the model generates a rubric conditioned on a set of demonstrations. In the second turn, the model conditions on the generated rubric and the candidate responses to assign a score to each candidate. For simplicity, we use a single model for both rubric writing and judge scoring. During training, as seen in Figure 2, we employ branched rollouts (Kazemnejad et al., 2025) to reduce variance (8 rubrics x 8 judgments): for each problem, we sample m rubrics and produce n rubric-conditioned judgments for each rubric, yielding $m \cdot n$ rubric–judgment trajectories. The resulting predicted scores are compared with the gold rewards to construct a learning signal that jointly updates both stages. Here, we use a listwise judge for efficient optimization, where a judge provides a score for a list of candidates. Advantages are computed per step for branched rollouts. We then have a self-distillation phase (Shenfeld et al., 2026), where we distill the listwise judge into a point-wise judge for scoring. The remaining question is how to define a reward function that captures the desired structural agreement between the predicted and gold rewards.

Objectives for reward function fitting. Given predicted and gold reward vectors $\hat{\mathbf{r}}_b$ and $\mathbf{r}_b^*$ for D candidate responses to each of B problems, we jointly fit the rubric writer and judge models using a reward function $g(\cdot)$ that measures the alignment of the predicted and gold rewards:

$$\mathcal{R}_{\text{align}} = \frac{1}{B} \sum_{b=1}^{B} g(\hat{\mathbf{r}}_b, \mathbf{r}_b^*)\,, \tag{1}$$

where g can measure numerical calibration, relative ordering, or candidate-selection quality. We use the harmonic best-of-n function (Equation 6) as our default reward for reinforcement learning with branched rollouts, since policy adaptation relies on relative credit assignment. This objective directly rewards correct relative ranking and gives graded credit across candidate-set sizes, targeting the credit-assignment and dense-signal desiderata.

Appendix A.7 details alternative objectives formulations and Appendix A.8 specifies a dense reward formulation for better credit assignment. We summarize `Meta-Rubrics` in Algorithm 1. None of these objectives guarantees extrapolation beyond the candidate sets seen during training, which raises the question of how to enable such extrapolation in the reward model.

3.2 Encouraging Generalization via Candidate Augmentation

The distribution of candidate responses is itself an important part of the reward-learning problem: the inferred rubric is evaluated only on its ability to distinguish among the candidates it is given. This restriction can simplify reward learning, since the rubric need not make reliable distinctions across all possible responses. However, a limited query set can also contain spurious correlations that a rubric exploits without recovering the underlying reward criteria. The rubric must also accommodate the multimodality of the solution space: equally valid responses may follow different strategies, use different intermediate steps, or express the same solution in different ways. For example in the user simulation setting, a user may write a review in many ways, which our rubric should capture.

To encourage generalization, we augment the query set with samples spanning a broad but controlled range of qualities, styles, and failure modes. Candidates that are too similar, or consist only of obvious successes and failures, provide little pressure to recover fine-grained reward criteria. Conversely, substantial differences between the support and query candidate distributions can confound evaluation: poor performance may reflect distribution shift rather than a failure to infer the reward structure. Candidate augmentation therefore aims to construct pools that are diverse enough to probe the inferred rubric, yet comparable enough to support meaningful generalization. Useful additions include hard negatives, near-miss responses, and improved variants produced through refinement.

Instantiation of Candidate Augmentation. We synthetically augment the candidate set by explicitly varying solution strategies, since unconditioned sampling alone may produce limited strategic diversity. Following RLAD (Qu et al., 2025), we first prompt the base policy to sequentially propose a set of abstractions: concise natural-language descriptions of viable strategies for solving the problem. We then use the same base policy to independently generate a candidate response conditioned on each abstraction. Both stages use the existing policy, without training a separate augmentation model; unlike RLAD, we apply no additional SFT or RL. This procedure serves as one possible instantiation of candidate augmentation for increasing the diversity of responses presented to the rubric learner. We perform this augmentation both during training/test time.

Overview of Learning Effective Rubric-Based Rewards

We jointly optimize the rubric writer and judge via branched RL rollouts to align proxy and gold rewards, targeting accurate credit assignment, generalization, and dense feedback. Structured candidate augmentation supports test-time extrapolation.

4 Policy Adaptation with the Amplified VG-Gap

Thus far, we have developed an approach for strengthening the verifier on a target task by inferring an environment-specific rubric and rubric conditioned scoring model. This enables us to distinguish between several candidate samples, which we expect to widen the verification-generation (VG) gap. We now consider how this widened VG gap can support efficient policy adaptation. In particular, we study two complementary approaches: (1) selecting better responses through test-time inference and (2) fine-tuning the policy with regularized reinforcement learning using the inferred reward.

Test-Time Inference. The inferred rubric can be used directly at inference time to select high-quality responses from a set of candidates. One simple approach is Best-of-N search (Snell et al., 2024), in which the policy samples multiple candidate responses and the rubric-conditioned verifier selects the highest-scoring one. The policy may additionally condition on the available demonstrations or other privileged information through in-context learning. This approach requires no parameter updates and therefore provides a lightweight mechanism for adapting the model to the target environment.

Policy Adaptation with Regularized Reinforcement Learning. Finally, the inferred rubric-based reward can serve as a proxy objective for reinforcement-learning-based policy fine-tuning. Unlike inference-time methods, this approach distills the verifier's preferences into the policy parameters, potentially improving generation without requiring repeated search or refinement at deployment time. However, optimizing against an imperfect learned reward may lead to reward overoptimization or exploitation. We therefore constrain policy updates using regularization techniques such as a KL penalty relative to the initial policy, low-rank adaptation (LoRA) to limit update capacity, and weight decay. Together, these constraints aim to preserve the base policy's capabilities while allowing it to adapt to the task-specific reward.

Overview of Policy Adaptation Approaches

We use the inferred rubric to adapt the policy through two complementary mechanisms: Best-of-N test-time inference and regularized RL with the rubric-based reward.

5 Experiments

Having formulated reward discovery as a meta-learning problem in Section 2, we now evaluate whether the resulting reward functions are useful for adapting a policy to previously unseen environments. Our experiments are organized around three questions. **RQ1** asks whether the learned rubrics improve test-time adaptation over baselines by selecting and refining better candidate responses. **RQ2** asks whether the inferred reward provides a sufficiently reliable signal for direct policy optimization through parameter-efficient fine-tuning. Finally, **RQ3** examines whether `Meta-Rubrics` recovers the structure of the underlying reward function, both quantitatively through agreement with oracle rewards and qualitatively through the criteria expressed in its rubrics.

Tasks. We evaluate `Meta-Rubrics` across two adaptation domains. First, we build on the book-review setting from HumanLM (Wu et al., 2026), where the task is to simulate a human user for writing a product review in terms of the style and content. We expand its coverage by removing the book-only filter from the original processing pipeline and incorporating additional Amazon categories, including fashion, appliances, electronics, and beauty. Applying the HumanLM preprocessing procedure to a subset of the data from Hou et al. (2024) produces a dataset containing 2,160 users (2,000 train, 60 val, 100 test). In this domain, the model must adapt to a user and product that were both unseen during training. We note that like HumanLM (Wu et al., 2026), we validate the adaptation to a new user and product using similarity with an LLM judge proxy to the unseen target review from the target user (which the authors showed had high agreement with human users). Second, in the DataCuration domain, the model must select or process examples from the Nemotron Cascade mid-training corpus (Yang et al., 2026b), subject to a variable token budget per task (e.g., 32k/64k/128k tokens). The curated data is used to fine-tune a target model, such as SmolLM2 (Allal et al., 2025), for a specified downstream domain or benchmark, such as a mathematics subset of

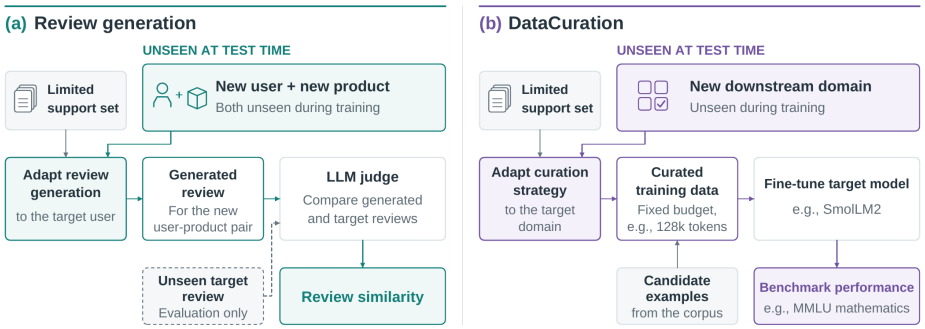

Figure 3: **Overview of the Reviews and DataCuration domains.** In Reviews (left), a new user's prior reviews guide adaptation to generate a review for an unseen product, evaluated for similarity to the target review. In DataCuration (right), the model curates data within a fixed token budget (e.g., 128k tokens) to fine-tune a target model for an unseen benchmark (e.g., MMLU mathematics).

MMLU. Here, the model must adapt its curation strategy to a target domain unseen during training. Both of these are ideal adaptation domains: collecting environment interaction data is hard to do at scale from your test user (e.g human annotations or fine-tuning LLMs) is a costly task, making adaptation with a limited support set necessary. A visualization for each domain is provided in Figure 3 and further details about the domains are provided in Appendix A.11.

Baselines. We compare against three families of baselines. The first comprises policy-adaptation methods that condition, refine, or aggregate responses without learning an explicit task-specific reward: in-context learning (ICL) (Brown et al., 2020), Self-Refine (Madaan et al., 2023), RSA (Venkatraman et al., 2026), and Feedback Descent (Lee et al., 2025), including variants with support-set-conditioned feedback. The second comprises evaluator- and consensus-based selection methods that do not meta-learn task-specific natural-language rubrics: an LLM-as-a-judge (Zheng et al., 2023) (`Meta-Rubrics` without training), population-level reward models trained with cross-entropy (CE) or Bradley–Terry (BT) objectives (Gao et al., 2022), and semantic majority voting (Snell et al., 2024; Zuo et al., 2025), where an LLM judge determines equivalence between candidate responses. Finally, oracle variants select the highest-reward candidate using the environmental reward, providing an upper bound for each sampled candidate set. All non-oracle methods share the same base policy, and support-conditioned variants receive the same privileged information. We report `Meta-Rubrics` both with and without our proposed candidate augmentation, whose sampling configuration matches RSA at depth 1, to assess the performance of the complete framework and the incremental contribution of augmentation. Each approach uses the same decoding hyperparameters and a standardized token budget for generation and verification as stated in Table 4 to control for inference compute.

Table 1: **Test-time inference results on Amazon Reviews.** We follow the HumanLM protocol to evaluate `Meta-Rubrics` using normalized review similarity, outperforming baselines across sample budgets.

Method	1	2	4	8	16	32
Self-Refine (Base)	0.35 (0.026)	0.31 (0.026)	0.33 (0.026)	0.33 (0.024)	0.32 (0.023)	0.33 (0.024)
Self-Refine (Support)	0.35 (0.023)	0.36 (0.023)	0.37 (0.024)	0.37 (0.023)	0.37 (0.024)	0.38 (0.025)
RSA (Support)	0.35 (0.023)	0.35 (0.024)	0.36 (0.024)	0.37 (0.024)	0.37 (0.024)	0.37 (0.025)
Feedback Descent (Support)	0.35 (0.023)	0.35 (0.024)	0.36 (0.024)	0.36 (0.024)	0.36 (0.025)	0.35 (0.025)
LLM Rubric Judge (Base)	0.35 (0.026)	0.35 (0.026)	0.35 (0.026)	0.35 (0.026)	0.36 (0.026)	0.36 (0.026)
LLM Rubric Judge (Support)	0.35 (0.023)	0.37 (0.024)	0.37 (0.025)	0.38 (0.026)	0.39 (0.027)	0.39 (0.027)
Fine-Tuned BT RM (Base)	0.35 (0.026)	0.35 (0.026)	0.35 (0.026)	0.35 (0.026)	0.36 (0.026)	0.36 (0.026)
Fine-Tuned BT RM (Support)	0.35 (0.022)	0.35 (0.022)	0.35 (0.022)	0.35 (0.022)	0.36 (0.022)	0.37 (0.022)
Fine-Tuned CE RM (Base)	0.35 (0.026)	0.36 (0.027)	0.36 (0.027)	0.36 (0.027)	0.37 (0.028)	0.37 (0.028)
Fine-Tuned CE RM (Support)	0.35 (0.022)	0.37 (0.025)	0.38 (0.027)	0.38 (0.027)	0.38 (0.027)	0.38 (0.028)
Semantic Maj@k (Base)	0.35 (0.026)	0.35 (0.026)	0.35 (0.026)	0.36 (0.027)	0.36 (0.027)	0.36 (0.028)
Semantic Maj@k (Support)	0.35 (0.022)	0.35 (0.023)	0.36 (0.024)	0.36 (0.025)	0.35 (0.026)	0.35 (0.026)
Ours (Support)	0.35 (0.022)	0.38 (0.025)	0.39 (0.027)	0.40 (0.028)	0.41 (0.028)	**0.43** (0.021)
Ours (Support + Cand. Aug.)	0.37 (0.026)	0.48 (0.020)	0.52 (0.024)	0.53 (0.024)	0.54 (0.026)	**0.55** (0.026)
Oracle (Base)	0.35 (0.026)	0.41 (0.029)	0.46 (0.040)	0.51 (0.040)	0.56 (0.029)	0.60 (0.028)
Oracle (Support)	0.35 (0.022)	0.43 (0.025)	0.49 (0.027)	0.54 (0.027)	0.58 (0.026)	0.62 (0.025)
Oracle (Support + Cand. Aug.)	0.37 (0.026)	0.53 (0.029)	0.64 (0.026)	0.71 (0.023)	0.76 (0.027)	0.80 (0.024)

RQ1: Test-Time Inference for Adaptation. We first evaluate whether the inferred reward can construct and leverage the verification-generation gap at inference time. For each query, the policy produces up to $k \in \{1, 2, 4, 8, 16, 32\}$ candidate trajectories. Methods that employ a reward model (e.g LLM judge) score these candidates and return the highest-scoring trajectory, while evolutionary methods (e.g RSA) use the corresponding generation budget to iteratively critique and revise their outputs. Sweeping the number of samples k allows us to understand the efficiency of the adaptation method. Table 1 reports the mean normalized similarity score and standard error on Amazon Reviews and Table 2 on Data Curation. `Meta-Rubrics` achieves the highest non-oracle mean score at every $k \geq 2$ in both domains. At $k = 32$, `Meta-Rubrics` captures **30%** of the oracle headroom and **75%** on Amazon Reviews and Data Curation, respectively. With candidate augmentation, this oracle headroom grows to **42%** for Amazon Reviews. Because augmentation also improves the candidate pool itself, as the higher oracle shows, this gain reflects both better candidates and better selection.

RQ2: Policy Optimization with Learnt Reward. We next consider a more demanding setting in which the learned reward must remain informative as the policy itself changes. For each test environment, we freeze the inferred rubric and rubric-conditioned judge and fine-tune a candidate policy with LoRA (Hu et al., 2021) and GRPO (Shao et al., 2024) for 20 steps ($\approx$ 1.3 million tokens per user) using the predicted scalar reward. Following Section 4, we regularize the update with a KL penalty relative to the initial policy, and use an ensemble of rubrics and weight decay. We compare `Meta-Rubrics` with rewards supplied by an LLM judge (Zheng et al., 2023), an inferred rubric without fine-tuning, RLER (Shao et al., 2026), and Rubric Dropout (Yang et al., 2026a). Each of these approaches are supplied the support set of environment interactions in context and are evaluated using the environmental oracle reward. We report mean and standard error over $n = 128$ samples for 30 held out (user, product) pairs in the book subset of Amazon (matching HumanLM books). As shown in Figure 4, optimizing against the reward inferred by `Meta-Rubrics` raises the oracle reward from 0.31 to 0.38. This is the highest among the learned rewards we compare (next best: RLER at 0.35), though below direct optimization of the oracle reward (0.43). Its rewards also correlate more strongly with the oracle during fine-tuning as seen in Table 6.

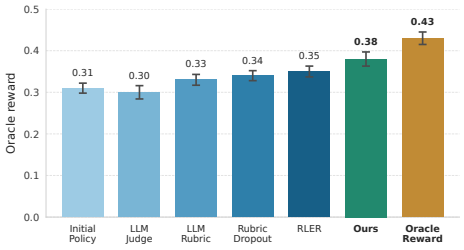

Figure 4: **LoRA-FT of `Meta-Rubrics`.** We fine-tune a LoRA of `Qwen 3.6 27b` with GRPO for 20 steps on Amazon Reviews for `Meta-Rubrics` and baseline methods.

RQ3: Does `Meta-Rubrics` align with the gold reward? As shown in Figure 5, `Meta-Rubrics` scores reach a Spearman correlation of 0.67 with the oracle reward, pooled across tasks, compared with 0.05 for the support-conditioned LLM judge. In Reviews, the correlation is 0.42 on standard candidates and 0.63 on the augmented pools. The increased spearman partly also reflects the greater spread of augmented candidates. These results suggest Meta-Rubrics recovers substantially more of the oracle's ranking structure than an untrained LLM judge. We also do an ablation of rubric-shuffling in Table 5, where scoring with a rubric from a different (user, product) pair lowers $k = 32$ performance from 0.43 to 0.37, indicating that the rubric carries task-specific information.

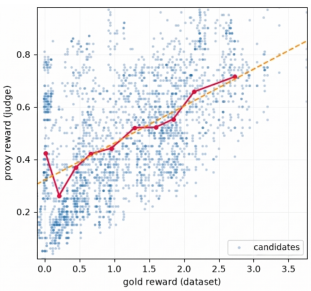

Figure 5: **Reward recovery.** Spearman correlation between evaluator and true reward for `DataCuration`.

Qualitative analysis. Finally, we inspect the natural-language rubrics and their criterion-level judgments in Section A.5. Across domains, `Meta-Rubrics` identifies properties that are not stated explicitly in the task prompt but are supported by the demonstrations. For example, the Amazon Reviews rubrics capture user-specific preferences over tone, level of detail, and product emphasis and Data Curation rubrics distinguish examples according to their relevance, difficulty, and expected training utility. The criteria also list common failure cases, including under-specification and shifts outside of the candidate distribution. Figure 6 illustrates how these inferred preferences guide generation: compared with the initial policy (Qwen 3.6 27B), `Meta-Rubrics` produces a review of *Stuart Little* that more closely matches the target review's emphasis on audience-specific details, such as the book's larger format for reading aloud to children and appealing illustrations.

Taken together, the results provide supportive evidence for all three questions: the inferred rubrics improve test-time selection, provide a useful signal for policy optimization, and recover part of the structure of the underlying reward.

6 Related Work

Test-time adaptation of LLM agents. Adapting pretrained models to new tasks at test time encompasses in-context learning (Brown et al., 2020), test-time training (Akyürek et al., 2024), and inference-time compute scaling (Wei et al., 2023; Snell et al., 2024). Recent approaches even apply test-time reinforcement learning using majority voting as a pseudo-label (Zuo et al., 2025). Crucially, these methods generally assume the task objective is known or approximated by self-consistency, and focus on updating the policy. In contrast, we treat the objective itself as an unknown quantity to be inferred, enabling adaptation in open-ended environments.

Evolutionary Search. A parallel line of research improves generations through textual feedback, spanning self-refinement and reflective memory (Madaan et al., 2023; Shinn et al., 2023), feedback-driven prompt and pipeline optimization (Pryzant et al., 2023; Yang et al., 2024; Khattab et al., 2023; Yuksekgonul et al., 2024), and explicit evolutionary methods such as Promptbreeder and GEPA (Fernando et al., 2024; Agrawal et al., 2026). However, the sample efficiency of these loops is bounded by the quality of their evaluation signal. In environments lacking a ground-truth evaluator, feedback from an ungrounded critic often misaligns with the true objective. `Meta-Rubrics` offers one way to address this bottleneck by supplying a environment-specific signal as our inferred rubrics are complementary and can be substituted into any existing refinement loop.

Rubrics and LLM judges as reward functions. Evaluating free-form outputs with LLM judges is standard practice (Zheng et al., 2023; Kim et al., 2024), potentially with reasoning (Zhang et al., 2025; Mahan et al., 2024) or pluralistic alignment (Sorensen et al., 2024). Structuring these judgments with explicit criteria, such as constitutional principles (Bai et al., 2022), programs (Gao et al., 2023) or rubrics (Gunjal et al., 2025; Liu et al., 2026; Viswanathan et al., 2025), improves reliability. While recent work explores evolving rubrics during training (Shao et al., 2026), the evolution is done by prompting a language model whose parameters remain fixed. We introduce two key distinctions: we infer rubrics per environment at test time from environment interactions and we jointly optimize the rubric writer and judge so criteria are shaped by their downstream utility on a given task.

Meta-learning rewards and inverse reinforcement learning. Recovering an objective from behavior is the focus of inverse RL (Ziebart et al., 2008; Finn et al., 2016; Fu et al., 2018), with meta-IRL amortizing this process to identify new rewards from few demonstrations (Xu et al., 2019; Poddar et al., 2024). However, these approaches encodes tasks as latent vectors, which doesn't leverage a foundation model's prior and yields an uninterpretable representation of the reward function. While recent approaches prompt LLMs to serve as zero-shot reward functions (Kwon et al., 2023; Ma et al., 2023; Xie et al., 2024), the resulting rewards may not faithfully capture the target task's objective. We bridge these paradigms: our task variable is a natural-language rubric inferred in context, which is actively trained with RL to achieve downstream rank agreement with an oracle reward, merging the interpretability of LLM critics with a meta-IRL objective for optimization.

7 Discussion, Limitations, and Future Work

We study the capability of *rubric discovery*, where agents deployed in novel, open-ended environments can recover the underlying reward function using a limited amount of environment interaction to construct and leverage the verification-generation (VG) gap. To instantiate this capability, we propose `Meta-Rubrics`, a framework which consists of a rubric generator that synthesizes task-specific evaluation criteria, and an LLM judge that applies these criteria to score candidate trajectories, both conditioned on a small support set of environment. `Meta-Rubrics` achieves the highest non-oracle scores among the methods we evaluate on two adaptation settings: (1) user simulation and (2) data curation, outperforming policy-only baselines and reward based evaluators.

A limitation of this work is that the inferred rewards may become unreliable as policy optimization produces responses beyond the support distribution. Our primary policy optimization experiments treat the inferred rubric and corresponding judge as fixed to isolate the learning problem and mitigate reward overoptimization. Future research could explore a variant where the rubric generator, the judge, and the candidate policy are dynamically co-adapted during the reinforcement learning phase. Furthermore, using the learnt rubrics to generate structured feedback to guide evolutionary search can be an interesting direction to explore.

8 AI USE STATEMENT

Large language models were used to assist with manuscript preparation, specifically for proofreading, suggesting phrasing improvements for clarity and conciseness, and generating boilerplate code for data formatting and baseline implementations. All conceptual contributions, including the meta-learning formulation, the identification of rubric desiderata, the choice of reward objectives, and the experimental designs, were done by human authors.

9 ETHICS STATEMENT

Meta-Rubrics infers reward functions from limited task demonstrations, which could inadvertently reinforce existing biases present in the support set or amplify problematic preferences (e.g., echo chambers in user simulation). We recommend that practitioners carefully curate support sets to promote diversity and implement safeguards to detect and mitigate the reflection of harmful responses. Future work should investigate mechanisms to incorporate explicit fairness constraints into the rubric generation process.

10 REPRODUCIBILITY STATEMENT

We provide reproducibility documentation in the Appendix, including detailed hyperparameters for training and inference (Tables 3–4), model configurations (base models used: `Qwen/Qwen3.6-27B`), complete prompting templates (Appendix A.4), and evaluation procedures. For the Amazon Reviews domain, we build on the HumanLM benchmark (Wu et al., 2026) with expanded categories from Hou et al. (2024). For Data Curation, we use the Nemotron Cascade mid-training corpus and evaluate on benchmarks such as MMLU. For fine-tuning, we use the Miles framework for all experiments in the paper.

REFERENCES

David Ifeoluwa Adelani, Jessica Ojo, Israel Abebe Azime, Jian Yun Zhuang, Jesujoba Oluwadara Alabi, Xuanli He, Millicent Ochieng, Sara Hooker, Andiswa Bukula, En-Shiun Annie Lee, Chiamaka Ijeoma Chukwuneke, Happy Buzaaba, Blessing Kudzaishe Sibanda, Godson Koffi Kalipe, Jonathan Mukiibi, Salomon Kabongo Kabenamualu, Foutse Yuehgoh, Mmasibidi Setaka, Lolwethu Ndolela, Nkiruka Odu, Rooweither Mabuya, Salomey Osei, Shamsuddeen Hassan Muhammad, Sokhar Samb, Tadesse Kebede Guge, Tombekai Vangoni Sherman, and Pontus Stenetorp. IrokoBench: A new benchmark for African languages in the age of large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2732–2757. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.139. URL https://aclanthology.org/2025.naacl-long.139/.

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning, 2024. URL https://arxiv.org/abs/2404.11018.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Ekin Akyürek, Mehul Damani, Linlu Qiu, Han Guo, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for abstract reasoning, 2024. URL https://arxiv.org/abs/2411.07279.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Kelsey R. Allen, Kevin A. Smith, and Joshua B. Tenenbaum. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117(47):29302–29310, November 2020. ISSN 1091-6490. doi: 10.1073/pnas.1912341117. URL http://dx.doi.org/10.1073/pnas.1912341117.

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. Healthbench: Evaluating large language models towards improved human health, 2025. URL https://arxiv.org/abs/2505.08775.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4623–4637. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.421. URL https://aclanthology.org/2020.acl-main.421/.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT-23th Conference on learning theory-2010*, pp. 13–p, 2010.

Javad Azizi, Branislav Kveton, Mohammad Ghavamzadeh, and Sumeet Katariya. Meta-learning for simple regret minimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6): 6709–6717, 2023. doi: 10.1609/aaai.v37i6.25823. URL https://ojs.aaai.org/index.php/AAAI/article/view/25823.

Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, Xinxing Zu, and Kimi Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness

from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Dimitris Bertsimas and Alex Paskov. An exact and interpretable solution to wordle. 2022. URL https://api.semanticscholar.org/CorpusID:252403210.

BIG-bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR, 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 49–58. PMLR, 2016. URL https://proceedings.mlr.press/v48/finn16.html.

Justin Fu, Katie Luo, and Sergey Levine. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1710.11248.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022. URL https://arxiv.org/abs/2210.10760.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2023. URL https://arxiv.org/abs/2211.10435.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. Are we done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.262. URL https://aclanthology.org/2025.naacl-long.262/.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains, 2025. URL https://arxiv.org/abs/2507.17746.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

Yu-Guan Hsieh, Shiva Kasiviswanathan, Branislav Kveton, and Patrick Blöbaum. Thompson sampling with diffusion generative prior. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 13434–13468, 2023. URL https://proceedings.mlr.press/v202/hsieh23a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Audrey Huang, Adam Block, Dylan J. Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism, 2024. URL https://arxiv.org/abs/2412.01951.

Jacob Jackson, Ben Trapani, Nathan Wang, and Wanqi Zhu. Improving Composer through real-time RL. Cursor Blog, March 2026. URL https://cursor.com/blog/real-time-rl-for-composer. Published March 26, 2026.

Jaap Jumelet, Leonie Weissweiler, Joakim Nivre, and Arianna Bisazza. MultiBLiMP 1.0: A massively multilingual benchmark of linguistic minimal pairs. *Transactions of the Association for Computational Linguistics*, 14:193–216, 2026. doi: 10.1162/tacl.a.600. URL https://aclanthology.org/2026.tacl-1.10/.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms, 2025. URL https://arxiv.org/abs/2410.01679.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines, 2023. URL https://arxiv.org/abs/2310.03714.

Seungone Kim, Jay Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Ryan S. Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing Fine-grained Evaluation Capability in Language Models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/803485352e61e3ebf41221e4776c9fd4-Abstract-Conference.html.

Branislav Kveton, Mikhail Konobeev, Manzil Zaheer, Chih-Wei Hsu, Martin Mladenov, Craig Boutilier, and Csaba Szepesvári. Meta-Thompson sampling. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5884–5893, 2021. URL https://proceedings.mlr.press/v139/kveton21a.html.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models, 2023. URL https://arxiv.org/abs/2303.00001.

Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi, and Thien Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 318–327. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-demo.28. URL https://aclanthology.org/2023.emnlp-demo.28/.

Yoonho Lee, Joseph Boen, and Chelsea Finn. Feedback descent: Open-ended text optimization via pairwise comparison, 2025. URL https://arxiv.org/abs/2511.07919.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7315–7330. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.653. URL https://aclanthology.org/2020.acl-main.653/.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. OpenRubrics: Towards Scalable Synthetic Rubric Generation for Reward Modeling and LLM Alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17417–17437. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.791. URL https://aclanthology.org/2026.acl-long.791/.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models, 2023. URL https://arxiv.org/abs/2310.12931.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models, 2024. URL https://arxiv.org/abs/2410.12832.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule based rewards for language model safety, 2024. URL https://arxiv.org/abs/2411.01111.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

Jessica Ojo, Odunayo Ogundepo, Akintunde Oladipo, Kelechi Ogueji, Jimmy Lin, Pontus Stenetorp, and David Ifeoluwa Adelani. AfroBench: How good are large language models on African languages? In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19048–19095. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.976. URL https://aclanthology.org/2025.findings-acl.976/.

OpenAI. Learning to reason with LLMs, September 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534. Association for Computational Linguistics, 2016. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Sriyash Poddar, Yanming Wan, Hamish Ivison, Abhishek Gupta, and Natasha Jaques. Personalizing reinforcement learning from human feedback with variational preference learning, 2024. URL https://arxiv.org/abs/2408.10075.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. Rlad: Training llms to discover abstractions for solving reasoning problems, 2025. URL https://arxiv.org/abs/2510.02263.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN 9780262182539. URL https://gaussianprocess.org/gpml/.

Amrith Setlur, Matthew Y. R. Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms, 2025. URL https://arxiv.org/abs/2506.09026.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. Dr tulu: Reinforcement learning with evolving rubrics for deep research, 2026. URL https://arxiv.org/abs/2511.19399.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/pdf?id=fR3wGCk-IXp.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Max Simchowitz, Christopher Tosh, Akshay Krishnamurthy, Daniel J. Hsu, Thodoris Lykouris, Miroslav Dudík, and Robert E. Schapire. Bayesian decision-making under misspecified priors with applications to meta-learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26382–26394, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/ddcbe25988981920c872c1787382f04d-Abstract.html.

Shivalika Singh, Angelika Romanou, Clémentine Fourrier, David Ifeoluwa Adelani, Jian Gang Ngui, Daniel Vila-Suero, Peerat Limkonchotiwat, Kelly Marchisio, Wei Qi Leong, Yosephine Susanto, Raymond Ng, Shayne Longpre, Sebastian Ruder, Wei-Yin Ko, Antoine Bosselut, Alice Oh, Andre Martins, Leshem Choshen, Daphne Ippolito, Enzo Ferrante, Marzieh Fadaee, Beyza Ermis, and Sara Hooker. Global MMLU: Understanding and addressing cultural and linguistic biases in multilingual evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18761–18799. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.919. URL https://aclanthology.org/2025.acl-long.919/.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models, 2025. URL https://arxiv.org/abs/2412.02674.

Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J. Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback, 2026. URL https://arxiv.org/abs/2602.02482.

Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, Tim Althoff, and Yejin Choi. A roadmap to pluralistic alignment, 2024. URL https://arxiv.org/abs/2402.05070.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/hash/f33ba15effa5c10e873bf3842afb46a6-Abstract.html.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization, 2021. URL https://arxiv.org/abs/2107.06882.

Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R. Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, Nikolay Malkin, and Moksh Jain. Recursive self-aggregation unlocks deep thinking in large language models, 2026. URL https://arxiv.org/abs/2509.26626.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models, 2025. URL https://arxiv.org/abs/2507.18624.

Wentian Wang, Sarthak Jain, Paul Kantor, Jacob Feldman, Lazaros Gallos, and Hao Wang. MMLU-SR: A benchmark for stress-testing reasoning capability of large language models, 2024a. URL https://arxiv.org/abs/2406.15468.

Zi Wang, George E. Dahl, Kevin Swersky, Chansoo Lee, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-trained gaussian processes for bayesian optimization. *Journal of Machine Learning Research*, 25(212):1–83, 2024b. URL https://jmlr.org/papers/v25/23-0269.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Shirley Wu, Evelyn Choi, Arpandeep Khatua, Zhanghan Wang, Joy He-Yueya, Tharindu Cyril Weerasooriya, Wei Wei, Diyi Yang, Jure Leskovec, and James Zou. Humanlm: Simulating users with state alignment beats response imitation, 2026. URL https://arxiv.org/abs/2603.03303.

Tianbao Xie, Siheng Zhao, Chen Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward Shaping with Language Models for Reinforcement Learning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9941833e8327910ef25daeb9005e4748-Abstract-Conference.html.

Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning, 2019. URL https://arxiv.org/abs/1805.12573.

Weihao Xuan, Rui Yang, Heli Qi, Qingcheng Zeng, Yunze Xiao, Aosong Feng, Dairui Liu, Yun Xing, Junjue Wang, Fan Gao, Jinghui Lu, Yuang Jiang, Huitao Li, Xin Li, Kunyu Yu, Ruihai Dong, Shangding Gu, Yuekang Li, Xiaofei Xie, Felix Juefei-Xu, Foutse Khomh, Osamu

Yoshie, Qingyu Chen, Douglas Teodoro, Nan Liu, Randy Goebel, Lei Ma, Edison Marrese-Taylor, Shijian Lu, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. MMLU-ProX: A multilingual benchmark for advanced large language model evaluation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1513–1532. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.79. URL https://aclanthology.org/2025.emnlp-main.79/.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3339f19c5fcee3ad74502947a32be9e6-Abstract-Conference.html.

Minglai Yang, Xinyu Guo, Utkarsh Tyagi, Mian Zhang, Razvan Dumitru, Sunjie Hou, Yunzhong He, Daniel Yue Zhang, and Ying Liu. Rubric dropout: A simple way to mitigate reward hacking in rubric-as-reward rl, 2026a. URL https://arxiv.org/abs/2608.11669.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, Renjie Pi, Grace Lam, Nayeon Lee, Alexander Bukharin, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-cascade 2: Post-training llms with cascade rl and multi-domain on-policy distillation, 2026b. URL https://arxiv.org/abs/2603.19220.

Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-inverse reinforcement learning with probabilistic context variables, 2019. URL https://arxiv.org/abs/1909.09314.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text, 2024. URL https://arxiv.org/abs/2406.07496.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175.

Tom Zahavy. Position: LLMs can't jump. In *Forty-Third International Conference on Machine Learning Position Paper Track*, Seoul, South Korea, July 2026. URL https://icml.cc/virtual/2026/poster/67091. Poster presentation.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2025. URL https://arxiv.org/abs/2408.15240.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

A Appendix

A.1 Algorithm

Our meta-rubrics framework separates adaptation into three stages, as summarized in Algorithm 1. First, the rubric generator and judge are jointly meta-trained using an alignment reward, after which the listwise judge is distilled into a pointwise verifier. At deployment, the frozen model infers a task-specific rubric from a small support set. This rubric then guides adaptation either through inference-time Best-of-(N) selection or through parameter-efficient reinforcement learning regularized toward the base policy.

Algorithm 1 `Meta-Rubrics` Adaptation Framework

Require: Support set Ψ_μ, Base policy π_0, Meta-trained generator/judge f_θ
Ensure: Task rubric $\mathcal{R}_\mu$, Adapted policy π_{adapted} (or best output τ^*)

// Phase 1: Joint Meta-Training

for each batch $b = 1 \dots B$ **do**
  Sample rubrics $\mathcal{R}_b \sim f_\theta(\cdot \mid \Psi_b)$ and listwise scores $\hat{r}_b$ via branched rollouts
  Update θ via RL with alignment reward $\mathcal{R}_{\text{align}} = \frac{1}{B}\sum_b g(\hat{r}_b, r_b^*)$ — (Eq. 1)
Distill listwise judge f_θ into pointwise verifier $f_\theta^{\text{pointwise}}$

// Phase 2: Deployment & Initialization

Freeze θ and π_0; deploy to target environment
Infer task rubric $\mathcal{R}_\mu \sim f_\theta(\cdot \mid \Psi_\mu)$ using support info Ψ_μ

// Phase 3: Policy Adaptation

if Best-of-N Search **then**
  Sample candidates $\{\tau_i\}_{i=1}^N \sim \pi_0$ and return $\tau^* = \arg\max_i f_\theta^{\text{pointwise}}(\tau_i \mid \mathcal{R}_\mu)$
else if Regularized RL **then**
  Update π_ϕ (via LoRA) against verifier $f_\theta^{\text{pointwise}}(\cdot \mid \mathcal{R}_\mu)$ with KL penalty:

$$\max_\phi \mathbb{E}_{\tau\sim\pi_\phi}\left[f_\theta^{\text{pointwise}}(\tau \mid \mathcal{R}_\mu)\right] - \beta D_{\text{KL}}(\pi_\phi \,\|\, \pi_0)$$

A.2 Additional Test-Time Inference Results

Table 2 presents the test-time inference results for $n = 100$ tasks in the Data-Curation domain, using the same baselines and inference samples as in Table 1. The training reward function optimizes for the delta in negative likelihood of the target prediction. To derive an aggregate normalized score, we scale this reward per panel using the minimum and maximum values across $n = 32$ samples, uniformly drawn from the `Qwen 3.6 27b` model. Standard retrieval (top 2048) has a performance of 0.5445 ± 0.0780 using the support set, which Qwen 3.6 27b improves upon even at 1 sample.

Table 2: **Test-time inference results on Data Curation across inference samples.** Values are per-panel min–max normalized rewards, with standard errors in parentheses.

Method	1	2	4	8	16	32
Self-Refine	0.55 (0.052)	0.68 (0.066)	0.61 (0.077)	0.42 (0.078)	0.62 (0.078)	0.50 (0.076)
Self-Refine (Support)	0.68 (0.033)	0.76 (0.061)	0.43 (0.076)	0.49 (0.083)	0.40 (0.083)	0.44 (0.085)
RSA (Support)	0.68 (0.033)	0.50 (0.071)	0.53 (0.071)	0.56 (0.073)	0.59 (0.064)	0.62 (0.070)
Feedback Descent (Support)	0.68 (0.033)	0.45 (0.100)	0.54 (0.099)	0.53 (0.095)	0.60 (0.095)	0.77 (0.088)
LLM Rubric Judge (Base)	0.55 (0.052)	0.52 (0.052)	0.56 (0.057)	0.60 (0.058)	0.54 (0.066)	0.55 (0.055)
LLM Rubric Judge (Support)	0.68 (0.033)	0.69 (0.058)	0.75 (0.038)	0.78 (0.040)	0.81 (0.040)	0.79 (0.041)
Fine-Tuned RM (Base)	0.55 (0.052)	0.55 (0.034)	0.56 (0.035)	0.57 (0.037)	0.58 (0.039)	0.57 (0.043)
Fine-Tuned RM (Support)	0.68 (0.033)	0.70 (0.035)	0.71 (0.037)	0.71 (0.040)	0.70 (0.042)	0.71 (0.047)
Semantic Maj@k (Base)	0.55 (0.052)	0.55 (0.052)	0.55 (0.052)	0.55 (0.052)	0.55 (0.052)	0.55 (0.052)
Semantic Maj@k (Support)	0.68 (0.033)	0.75 (0.044)	0.75 (0.044)	0.75 (0.044)	0.75 (0.044)	0.75 (0.044)
Ours (Support)	0.68 (0.033)	0.78 (0.031)	0.84 (0.028)	0.87 (0.025)	0.90 (0.022)	**0.92 (0.029)**
Oracle (Base)	0.55 (0.052)	0.65 (0.043)	0.74 (0.042)	0.79 (0.041)	0.87 (0.038)	1.00 (0.000)
Oracle (Support)	0.68 (0.033)	0.79 (0.042)	0.86 (0.032)	0.91 (0.024)	0.94 (0.020)	1.00 (0.000)

A.3 HYPERPARAMETERS FOR META-RUBRICS AND BASELINES

Below in Table 3 are hyperparameters for the training and inference of Meta-Rubrics.

Table 3: Key training hyperparameters used in Meta-Rubrics.

Hyperparameter	**Value**
algorithm	DaPO (Yu et al., 2025)
train batch size	128
max prompt length	32768
max response length	32768
learning rate	1e-6
clip ratio (low / high)	0.2 / 0.5
entropy coefficient	0.001
KL loss coefficient	0.001
KL loss type	low variance kl
sampling temperature (train / val)	1.0 / 1.0
samples per prompt (train / val)	8x8 / 1x1
max batched tokens	131072
inference branching size (rubrics/judge)	8/8

Below, in Table 4 are hyperparameters for the inference baselines.

Table 4: Hyperparameters for inference (including Meta-Rubrics).

Setting	Value
Evaluation setup	
Cohort	100
Candidate budgets	$k \in \{1, 2, 4, 8, 16, 32\}$
Bank size	256 candidates per task (to calculate pass@k)
Banks	Independent Base and ICL banks
Support Set	Eight demonstrations in fixed order
Candidate generation	
Model	`Qwen/Qwen3.6-27B`
Temperature	0.7
Top-p	0.8
Top-k	20
Max tokens	4,096 (reviews), 16,384 (data curation)
Thinking	Disabled
Stop sequence	`</response>`
Rubric Writing (frozen/trained)	
Model	`Qwen/Qwen3.6-27B`
Decoding	Temperature-Based
Temperature	0.7
Top-p	0.8
Top-k	20
Max tokens	4,096
Thinking	Disabled
Judge	
Model	`Qwen/Qwen3.6-27B`
Decoding	Temperature-Based
Temperature	0.7
Top-p	0.8
Top-k	20
Max tokens	4,096
Thinking	Disabled
Evaluation oracle (Reviews)	
Model (train/eval)	`openai/gpt-oss-20b` & `openai/gpt-oss-120b`
Decoding	Greedy
Max tokens	4,096
Reasoning effort	Low
Score range	Clipped to $[0, 1]$

A.4 PROMPTS

Rubric Writing Prompt

```
Write a weighted rubric for how closely a candidate Amazon review matches what THIS
reviewer would likely write about THIS product, using their past reviews and the
product context.

There is no fixed number of criteria, but do not stop after a few broad criteria.
Prefer many atomic, highly specific criteria over a small set of compound ones.
Include every distinct judgment, product detail, use case, reaction, preference,
caveat, or contradiction that would help distinguish this reviewer's likely review,
while avoiding padding, duplication, and claims unsupported by the supplied evidence.

Decompose the likely review as finely as the evidence allows. Consider its overall
verdict and rating; the product features or attributes the reviewer would mention;
intended use and audience; setup, fit, sizing, or compatibility when relevant;
usability and performance; materials, quality, reliability, or durability; value;
packaging or delivery when supported; comparisons; praise, criticism, caveats, and
recommendations; and recurring content priorities from the reviewer's history. These
are prompts for evidence-grounded specificity, not a mandatory checklist. Split
independently gradeable ideas into separate criteria. A criterion that requires two
ideas which could appear independently is usually two criteria.

For each criterion:
- Make it directly gradeable and specific to this reviewer and product. State the
expected stance or product detail precisely enough to distinguish a match from generic
category-level praise or criticism. A product-content criterion should not apply
unchanged to an unrelated product.
- Give it a signed, nonzero point weight. Positive points reward likely content.
Negative points penalize a specific contradiction or clearly off-target claim. Point
magnitudes express relative importance and do not need to add to any particular total.
Do not use a negative criterion merely for omission or uncertainty.
- Give a criterion-specific guide for assigning an activation fraction from 0 to 1.
For positive points, 1 means a complete match and 0 means no match. For negative
points, 1 means the contradiction is fully present and 0 means it is absent.
Intermediate fractions must also stay in [0,1].
- Give generous partial credit for a correct paraphrase, clearly implied meaning, or
the correct central judgment with missing detail. For negative criteria, require
affirmative evidence and resolve ambiguity toward the smaller penalty.
- Add optional descriptive tags when they clarify scope, stance, evidence, importance,
product detail, or reviewer preference. Tags never affect the score.

Avoid generic criteria about review quality, formatting, headings, HTML, length, or
punctuation unless that preference is clearly characteristic of this reviewer and
relevant to this product. Do not invent private reviewer attributes, firsthand
experiences, or product claims that are unsupported by the demonstrations and supplied
product context.

Put every criterion in one Markdown rubric inside a single pair of `<rubric>` tags.
Make criterion boundaries unmistakable. The repeated layout below is strongly
recommended, but equivalent clear Markdown is acceptable; the labels, grading anchors,
number of levels, and tags may vary. Keep each criterion's point weight and grading
guide in the same section, leave a blank line between criteria, and never open a
separate `<rubric>` block for each criterion.

This abbreviated example demonstrates formatting only. A real rubric should be much
more comprehensive:

<rubric>
### [Short criterion title]
**Criterion:** [One atomic, gradeable statement]
**Points:** [+ or - numeric weight]
**Grading:** [Criterion-specific anchors from 0 to 1]
**Tags:** [Optional descriptive tags]

### [Next short criterion title]
**Criterion:** [Another atomic, gradeable statement]
**Points:** [+ or - numeric weight]
**Grading:** [Its own criterion-specific anchors from 0 to 1]
</rubric>

Output only the complete Markdown rubric inside `<rubric>` tags.
```

Rubric Judging Prompt (listwise)

```
Score EACH proposed training-data curation independently against the weighted rubric
for expected downstream fine-tuning usefulness. Judge the submitted data and curator
evidence, not writing polish, and do not infer hidden evaluation examples or outcomes.

The target task:
<|The Start of Target|>
{target_prompt}
<|The End of Target|>

The weighted rubric:
<|The Start of Rubric|>
{rubric}
<|The End of Rubric|>

The {n_candidates} candidate curations, each containing its tool trajectory, aggregate
selection statistics, and a uniform sample of resolved training examples:
{candidates_block}

Treat each clearly separated Markdown section as one criterion. For every candidate
and criterion, identify the signed point weight `p` and assign an activation fraction
`f` in [0,1] from its grading guide.

- For positive points, `f` measures how fully the candidate satisfies the requirement.
- For negative points, `f` measures how fully the stated defect or contradiction is
present.
- Use partial credit when visible evidence supports it, and resolve uncertainty
conservatively when sampled examples do not expose the complete selected set.

For each candidate, convert every activation fraction to a match fraction `m`:

- If `p > 0`, use `m = f`.
- If `p < 0`, use `m = 1 - f`.

Compute one absolute-weighted match score per candidate:

`score = 100 * sum(abs(p) * m) / sum(abs(p))`

Clamp each final score to [0,100]. The weights need not sum to 100. Do not rank,
spread, or calibrate scores across candidates; equivalent evidence may receive the
same score. If there is no usable nonzero-weight criterion, use 0.

Return exactly one Markdown `json` code fence containing a JSON object that maps every
candidate number from `"1"` through `"{n_candidates}"` to its numeric score in [0,100].
Include each candidate exactly once and output no prose outside the fence.
```

Rubric Judging Prompt (pointwise)

```
Score this candidate review against the weighted rubric for what this user would
likely write about this product. Do not reward general review quality.

<|The Start of Rubric|>
{rubric}
<|The End of Rubric|>

Book context:
<|The Start of Context|>
{context}
<|The End of Context|>

Candidate:
<|The Start of Candidate|>
{candidate}
<|The End of Candidate|>

Treat each clearly separated Markdown section or list item as one criterion, even if
the rubric does not use a uniform layout. For every criterion, identify its signed
point weight `p` and assign an activation fraction `f` in [0,1] from its grading guide.
For positive points, `f` measures how fully the candidate matches the desired content.
For negative points, `f` measures how fully the stated contradiction or off-target
content is present. Infer or interpolate a fraction when useful, but never use a
fraction outside [0,1].

Convert each activation fraction to a match fraction `m`:
- If `p > 0`, use `m = f`.
- If `p < 0`, use `m = 1 - f`.
```

```
Compute the absolute-weighted match score over all nonzero-weight criteria:

`score = 100 * sum(abs(p) * m) / sum(abs(p))`

Then clamp the final score to [0,100]. The points are relative weights and do not need
to sum to 100. Do not separately average criteria, normalize against other candidates,
or otherwise rescale or calibrate the result. If no usable nonzero-weight criterion
exists, use 0.

Give generous criterion-specific partial credit for paraphrases, implied meaning, and a
correct central judgment with missing detail. For a negative criterion, require
affirmative evidence before assigning a nonzero contradiction fraction and resolve
ambiguity toward the smaller penalty.

End with one line containing only `Final score: ` followed by the numeric score in
[0,100], with no percent sign. Example: `Final score: 73.5`
```

A.5 Sample Rubrics and Judgements

Sample Rubric for HumanLM Amazon Reviews

```
### Critiques Realism/Plausibility ("Edge of Reality")
**Criterion:** Evaluates the story's grounding in reality, explicitly noting if the
action or plot is "lacking the edge of reality," "non-stop action" that feels too
easy/fair unearned, or conversely, "plausible" and realistic.
**Points:** +4
**Grading:** 1 if the reviewer comments on the plausibility, realism, or-handedness of
the plot/action (e.g., "unrealistic," "too easy," "plausible," "edge of reality"); 0.5
if they mention action quality without linking it to realism/plausibility; 0 if no
comment on the nature of the action/plot mechanics.
**Tags:** Content Focus, Criticism, Realism

### Praises "Edge of Your Seat" Action Thriller Quality
**Criterion:** Explicitly praises the book for being a high-intensity, "nail-biting,"
or "edge of your seat" action thriller, specifically noting if the book delivers on
this promise or fails to do so compared to expectations.
**Points:** +5
**Grading:** 1 if the review explicitly mentions "edge of your seat," "nail-biting,"
or similar high-intensity action descriptors in the context of the book's quality; 0.5
if it mentions the book is "action-packed" or "fast-paced" without the specific
intensity qualifiers; 0 if it discusses pace or action generally without claiming
high-intensity thrills.
**Tags:** Content Focus, Action, Vince Flynn Style

### Expresses Caution About Future Purchases Upon Disappointment
**Criterion:** If the review is critical, it explicitly states an intention to be
"cautious" or hesitant when purchasing future books by the same author, reflecting a
pattern of withdrawing loyalty only after specific dissatisfaction.
**Points:** +4
**Grading:** 1 if the review explicitly mentions being cautious, hesitant, or doubtful
about buying the author's next work due to this specific disappointment; 0.5 if it
expresses general disappointment or lowered ratings without the specific
future-purchase caveat; 0 if it criticizes the book but does not mention future buying
behavior.
**Tags:** Behavioral Pattern, Criticism, Vince Flynn Style

### References Specific Heroes or Series Lineage
**Criterion:** Explicitly names the primary protagonist series (e.g., "Mitch Rapp,"
"John Clark," "Jack Ryan") or compares the current work to other entries/books in that
specific universe or by that specific author.
**Points:** +4
**Grading:** 1 if the review names the specific hero or series; 0.5 if it refers
generically to "the hero" or "this author" without naming; 0 if no specific
identification.
**Tags:** Content Focus, Character Identification, Specificity

### Validates Authenticity Mission-Focused Narrative
**Criterion:** Evaluates whether the narrative stays faithful to the protagonist's
core mission (national security, counter-terrorism) rather than getting bogged down in
bureaucratic politics or "naysayers," explicitly praising or criticizing the focus on
"doing what he must" vs. "political bull."
**Points:** +4
**Grading:** 1 if the review comments on whether the book maintained a strong focus on
the hero's mission/action versus political bureaucracy; 0.5 if it mentions politics or
mission but doesn't critique the balance; 0 if it discusses plot points without
addressing the thematic focus on mission vs. politics.
**Tags:** Content Focus, Thematic Balance, Vince Flynn Style
```

```
### Compares Quality to a "Gold Standard" Author or Book
**Criterion:** Explicitly compares the current book's quality, style, or pacing to a
superior benchmark author (e.g., Tom Clancy) or a specific "best" book in that
author's catalog (e.g., "Without Remorse"), often using phrases like "not in the
Clancy Class" or "never topping my favorite."
**Points:** +4
**Grading:** 1 if the review names a specific author or book as a superior benchmark
for comparison; 0.5 if it makes a general comparison to "other thrillers" or
"Clancy-style" without naming a specific superior reference; 0 if no comparative
benchmark is mentioned.
**Tags:** Content Focus, Comparison, Benchmarking

### Praises "Good Ole" Character-Driven Action Over Politics
**Criterion:** Specifically praises the focus on the core action heroes (Mitch Rapp
and Mike Nash) and dismisses or critiques the presence of political bureaucracy,
government officials, or "political bull" as detracting from the core appeal.
**Points:** +4
**Grading:** 1 if the review explicitly contrasts the enjoyment of the
characters/action with a dislike for political elements or government interference;
0.5 if it praises the characters but does not explicitly critique the political
aspects; 0 if it focuses only on plot summary or general action without the
character/politics dichotomy.
**Tags:** Content Focus, Vince Flynn Style, Character Preference

### Uses Conversational, Fragmented Syntax
**Criterion:** Employs a conversational, somewhat informal tone characterized by
sentence fragments, direct addresses to the author (e.g., "Tom, you can do
better..."), or colloquial phrasing (e.g., "good ole," "bully"), rather than formal,
structured prose.
**Points:** +3
**Grading:** 1 if the review uses sentence fragments, direct address to the author, or
distinctively colloquial/conversational markers; 0.5 if it is informal but uses
standard full sentences and no direct address; 0 if the tone is formal, academic, or
strictly professional.
**Tags:** Writing Style, Voice, Informality
```

Sample Rubric for Data Curation

```
Task: Egyptian secondary-school Arabic-language multiple-choice questions
### Arabic-language match
**Criterion:** Selected examples are primarily written in Arabic and preserve the
target's Egyptian classroom register.
**Points:** +18
**Grading:** 1 = nearly all examples match; 0.5 = mixed Arabic/register quality; 0 =
little relevant Arabic content.

### Curriculum relevance
**Criterion:** Examples teach Egyptian secondary-level Arabic grammar, morphology,
rhetoric, literature, or textual analysis.
**Points:** +20
**Grading:** 1 = strongly aligned curriculum content; 0.5 = generally related language
material; 0 = unrelated domains.

### Multiple-choice format
**Criterion:** Examples use an Arabic question with clearly enumerated alternatives and
a single correct choice.
**Points:** +10
**Grading:** 1 = format consistently matches; 0.5 = partial format match; 0 = primarily
non-MCQ data.

### Answer-format match
**Criterion:** Responses teach the model to return the requested option letter
concisely, such as `Alif`, `Baa`, `Jeem`, or `Daal`.
**Points:** +10
**Grading:** 1 = answers consistently use the target format; 0.5 = answers include the
option plus extra text; 0 = incompatible outputs.

### Label correctness
**Criterion:** Visible selected examples have correct questions, options, and answer
labels.
**Points:** +18
**Grading:** 1 = no evident errors; 0.5 = occasional questionable labels; 0 = frequent
contradictions or incorrect answers.

### Subskill coverage
**Criterion:** The selection covers multiple relevant Arabic-language subskills without
overconcentrating on one narrow topic.
```

```
**Points:** +10
**Grading:** 1 = broad, balanced coverage; 0.5 = useful but narrow coverage; 0 = highly
repetitive or irrelevant coverage.

### Difficulty match
**Criterion:** Examples require reasoning and knowledge appropriate for Egyptian
secondary-school examinations.
**Points:** +8
**Grading:** 1 = closely matched difficulty; 0.5 = mostly easier or harder; 0 = clearly
inappropriate level.

### Token-budget use
**Criterion:** The selection uses the 65,536-token budget efficiently for relevant,
nonduplicative examples.
**Points:** +8
**Grading:** 1 = near-full, efficient use; 0.5 = substantial unused budget or some
waste; 0 = severe underuse or invalid over-budget submission.

### Distribution mismatch
**Criterion:** The selection contains substantial non-Arabic, non-language-subject, or
incompatible task-format material.
**Points:** -15
**Grading:** 1 = mismatch dominates; 0.5 = noticeable mismatch; 0 = no meaningful
mismatch.

### Redundancy
**Criterion:** The selection wastes capacity on near-duplicate questions or repeated
templates without adding useful coverage.
**Points:** -8
**Grading:** 1 = heavily repetitive; 0.5 = moderate redundancy; 0 = examples are
meaningfully distinct.
```

Sample Pointwise Judgement for Amazon Reviews

```
### Characterization of Reading Experience (Immersion/Pacing)
**Points:** +1.5
**Activation:** 1.0
The candidate uses phrases like "haunting," "lingers in the mind," and "completely
absorbed." This aligns well with the "immersive" or "unable to put down" descriptor,
although it leans slightly more towards emotional resonance than pure physical
page-turning, but "completely absorbed" is strong evidence.

### Plot Summary Presence
**Points:** +1.5
**Activation:** 0.8
The candidate mentions "young boy living in isolation on a hill," "witness a traumatic
event," and "eccentric parents." It misses the specific detail of the census
taker/stranger arrival which is a key part of the conflict, but covers the premise
(boy, hill, trauma) adequately.

### Emotional Reaction & Character Comparison
**Points:** +1.0
**Activation:** 0.0
The candidate describes the prose and atmosphere ("exquisite," "unease") but does not
characterize the boy's internal state (vulnerable, lonely, complex) or compare him to
other archetypes. It mentions "psychological exploration" but doesn't delve into the
character's specific psychology as requested by the rubric criteria regarding the
boy's identity/isolation connection.

### Evaluation of Tone and Genre
**Points:** +1.0
**Activation:** 1.0
The candidate explicitly uses "haunting," "dreamlike," "dark fairy tale for adults,"
"surreal," and "literary." This perfectly matches the positive criterion for defining
the book's tone/genre.

### Comparison to Other Works
**Points:** +1.0
**Activation:** 1.0
The candidate mentions "George Saunders or Karen Russell," which matches the rubric's
example comparisons.

### Author Assessment
**Points:** +0.5
**Activation:** 1.0
The candidate names China Mieville and praises his world-building ("strange, surreal
world Mieville creates").
```

```
### Direct Recommendation or Warning
**Points:** +0.8
**Activation:** 1.0
The candidate ends with "Highly recommended."

### Sentiment Alignment (Positive Lean)
**Points:** +0.5
**Activation:** 1.0
The review is overwhelmingly positive ("Unsettling and beautiful," "exquisite,"
"Highly recommended").

Final score: 83.3
```

A.6 Sample Policy Adaptation

As seen in Figure 6, we compare a policy adapted with LoRA via `Meta-Rubrics` vs the base model. Here, we find that the response after fine-tuning captures more than generic positivity: the large format, illustrations, and suitability for reading together all match the target's reasons for preferring this edition.

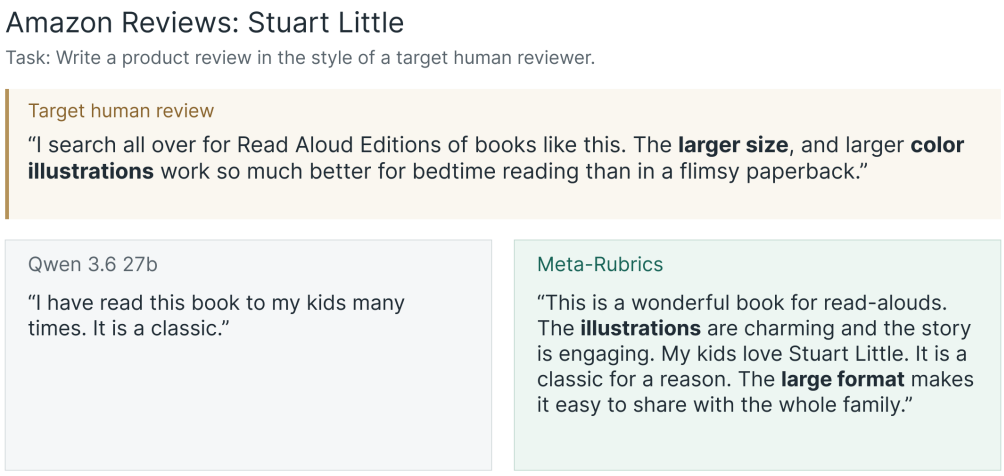

Figure 6: **Sample response after policy adaptation with `Meta-Rubrics` on Amazon Reviews** Meta-Rubrics identifies some salient details such as the colored illustrations and large format.

A.7 Alignment Objectives for Reward Function Fitting

Given predicted rewards $\hat{r}_{b,d}$ and gold rewards $r^*_{b,d}$ for D candidate responses to each of B problems, several objectives can be used to fit the rubric writer and judge models. Each captures a different notion of agreement with the gold reward, which we can use as a reward function for RL optimization.

The first set of objectives measure the per-candidate deviation from the gold reward. These objectives include mean squared error (MSE)

$$\mathcal{R}_{\text{MSE}} = -\frac{1}{BD}\sum_{b=1}^{B}\sum_{d=1}^{D}\left(\hat{r}_{b,d} - r^*_{b,d}\right)^2, \tag{2}$$

and soft-label cross entropy (CE)

$$\mathcal{R}_{\text{CE}} = \frac{1}{BD}\sum_{b=1}^{B}\sum_{d=1}^{D}\left[r^*_{b,d}\log\hat{r}_{b,d} + \left(1 - r^*_{b,d}\right)\log(1-\hat{r}_{b,d})\right]. \tag{3}$$

These objectives encourage the proxy to be numerically calibrated to the gold reward function, with CE penalizing confident mistakes more sharply and MSE treating all deviations symmetrically.

Policy adaptation primarily depends on relative credit assignment among candidates, so another natural objective is judging how well the proxy reward preserves the ordering of the candidate set.

Writing $\hat{\mathbf{r}}_b = (\hat{r}_{b,1}, \ldots, \hat{r}_{b,D})$ and $\mathbf{r}^*_b = (r^*_{b,1}, \ldots, r^*_{b,D})$ for the predicted and gold reward vectors of problem b, we define the objective as

$$\mathcal{R}_{\text{corr}} = \frac{1}{B} \sum_{b=1}^{B} \rho(\hat{\mathbf{r}}_b, \mathbf{r}^*_b)\,, \tag{4}$$

where ρ is a correlation objective such as Spearman, Kendall-τ (pairwise ranking), or Pearson.

A third family of objectives measures the efficacy of the proxy reward in the way it is ultimately used, namely selecting among candidates. For a fixed candidate-set size $n = D$, let $\mathcal{A}_b = \arg\max_d \hat{r}_{b,d}$ be the set of candidates attaining the highest predicted reward and let $\bar{r}^*_b = \frac{1}{D}\sum_d r^*_{b,d}$. We define the normalized best-of-n reward as

$$\mathcal{R}_{\text{BoK}}(D) = \frac{1}{B} \sum_{b=1}^{B} \frac{\frac{1}{|\mathcal{A}_b|}\sum_{d\in\mathcal{A}_b} r^*_{b,d} - \bar{r}^*_b}{\max_d r^*_{b,d} - \bar{r}^*_b}. \tag{5}$$

We average the gold reward over the argmax set for tie-breaking. The normalization places 0 at random selection in expectation and 1 at oracle selection, so problems of differing difficulty and reward range contribute comparably. To encourage strong selection performance across candidate-set sizes, we use the harmonic best-of-n objective,

$$\mathcal{R}_{\text{HBoK}} = \sum_{n=2}^{D} \frac{1}{n}\, \mathcal{R}_{\text{BoK}}(n), \tag{6}$$

where $\mathcal{R}_{\text{BoK}}(n)$ is the same normalized objective averaged over uniformly sampled subsets of n candidates per problem, with the baseline and oracle rewards computed within each subset. The harmonic weights emphasize smaller candidate sets while retaining contributions from larger ones. For each of the relative objectives, candidate sets with no variation in their gold rewards carry no relative credit-assignment information and are omitted or assigned zero weight; this also excludes singleton sets.

A.8 Dense Credit Assignment via Iterative Rubric Construction

Thus far, we have treated rubric generation as a single-shot process: the rubric writer generates a complete set of criteria, after which the judge scores the candidate responses and receives a single terminal reward. Although this objective measures the quality of the resulting reward function, it provides limited information about which individual criteria contributed to its success. A useful criterion and a redundant or misleading criterion receive the same trajectory-level return. Can we instead assign credit to each criterion as the rubric is constructed, shaping the reward function (Ng et al., 1999) for denser credit assignment and easier optimization?

Sequential Rubric Generation. We formulate rubric writing as a sequential decision process. At step t, the rubric writer generates a new criterion $c_{b,t}$, conditioned on the support information Ψ_b and the existing rubric prefix for problem b. Appending the criterion produces

$$\mathcal{C}_{b,t} = \mathcal{C}_{b,t-1} \cup \{c_{b,t}\}, \qquad \mathcal{C}_{b,0} = \emptyset. \tag{7}$$

Each criterion contains a natural-language description, grading guidance, and an optional importance weight. The rubric writer may additionally emit a stopping action when it determines that the current rubric sufficiently captures the task objective.

After each criterion is added, the judge evaluates the query candidates using the current rubric prefix. Let

$$\hat{\mathbf{r}}_b^{,t} = \left(\hat{r}_{b,1}^{,t}, \ldots, \hat{r}_{b,D}^{,t}\right) \tag{8}$$

denote the scores induced by $\mathcal{C}_{b,t}$, and let $\mathbf{r}^*_b$ denote the corresponding gold reward vector. We define the potential of a partial rubric as its agreement with the underlying reward:

$$\Phi_b\left(\mathcal{C}_{b,t}\right) = \rho\left(\hat{\mathbf{r}}_b^{,t}, \mathbf{r}^*_b\right), \tag{9}$$

where ρ may be Spearman, Kendall-Tau, or another downstream-aligned reward-learning objective. We define the potential of the empty rubric as zero, or as the performance of the unconditioned judge when such a baseline is available.

Criterion-Level Reward Shaping. We assign each newly generated criterion a reward equal to its marginal contribution to the quality of the induced reward function:

$$\mathcal{R}_t^{\text{dense}} = \frac{1}{B}\sum_{b=1}^{B}\left[\Phi_b\left(\mathcal{C}_{b,t}\right) - \Phi_b\left(\mathcal{C}_{b,t-1}\right)\right]. \tag{10}$$

A criterion receives positive reward when it improves agreement with the underlying objective, near-zero reward when it is redundant, and negative reward when it introduces an incorrect distinction or degrades credit assignment. This provides substantially denser supervision than applying the final correlation reward uniformly to every criterion in the completed rubric.

Importantly, this shaped reward preserves the original terminal objective. For a rubric containing T criteria, the intermediate rewards telescope:

$$\sum_{t=1}^{T}\mathcal{R}_t^{\text{dense}} = \frac{1}{B}\sum_{b=1}^{B}\left[\Phi_b\left(\mathcal{C}_{b,T}\right) - \Phi_b\left(\mathcal{C}_{b,0}\right)\right]. \tag{11}$$

Thus, when the initial potential is fixed, maximizing the cumulative criterion-level reward is equivalent to maximizing the quality of the final rubric.

During meta-training, the potential is computed using oracle rewards on the held-out query set. At test time, oracle rewards are no longer required: the learned rubric writer generates criteria sequentially and applies its learned stopping behavior before the final rubric-conditioned judge scores candidate trajectories. When reward-labeled support examples are available, the same potential can additionally be evaluated on the support set to rerank alternative criteria online.

Sequential Rubric Construction Provides Dense Credit Assignment.

By evaluating every rubric prefix, we reward each criterion according to its marginal improvement in reward agreement. These shaped rewards telescope to the original final-rubric objective while providing a denser and more localized learning signal for the rubric writer.

A.9 Reward Inference versus Direct Policy Adaptation in Synthetic Bandits

Motivation and Problem Setup As discussed in Section 2, we would like to consider how a policy should adapt given access to support information Ψ_τ for an unknown task. In this section, we defined a task as a markov decision process (MDP), but simplified this to a setting where only the reward function for a given state s is unknown. We define a simpler version of the task τ as a set of states S and its corresponding reward function $R_\tau(s), s \in S$. This reward function is unknown at deployment time (e.g encountering a new user) and we gain access to a support dataset Ψ_τ. Our goal is to optimize a policy $\pi(a|s, \Psi_\tau)$ that maximizes the reward function R_τ.

We would like to show that amplifying the verification-generation (VG) gap by inferring the target environment's reward function R_τ is more sample-efficient adaptation strategy than direct predicting the optimal action through policy adaptation, learning $\hat{\pi}(a|s, \Psi_\tau)$. We isolate and evaluate this underlying claim in a controlled, one step decision making problem in the form of a distribution of bandit tasks. Here, each bandit task τ represents a distinct environment (analogous to a specific user in the user simulation setting), and its arms represent candidate trajectories. At deployment, the learner receives a support set Ψ_τ comprising p arms and their noisy scalar rewards $\{(s_i, R(s_i)\}_{i=1}^{p}, s_i \in S$. The goal is to select the optimal arm from the full set of k candidates. In this simplified setting, direct policy adaptation reduces to predicting an optimal arm distribution directly from the support set Ψ_τ. Conversely, reward inference, the numerical analogue to our natural-language rubric paradigm, reconstructs a reward function for all k arms before selecting the maximizer.

Our task can be viewed as an extension of the best arm identification problem from Audibert et al. (2010), where we consider a multi-task bandit setting and aim to identify the best arm for an unseen task, given access to the support set Ψ_τ.

Task Distribution and Rewards. We design the task distribution to exhibit two complementary forms of structure: regularity shared across tasks and task-specific variation, which must be inferred from the support set at deployment. This prevents either source of information from being sufficient on its own and creates a controlled setting in which we can study how efficiently different adaptation

strategies use support observations. If we consider the user simulation domain, this structure is readily present with humans maybe sharing a set of common values as a population (exemplified by RLHF having consistent agreement) but a set of personal as values that define their character.

We use $k = 20$ equally spaced arm coordinates $x_i \in [-1, 1]$. We then define the latent mean reward for each arm i in task τ as

$$\mu_\tau(i) = \lambda f_{\text{shared}}(x_i) + (1 - \lambda) f_\tau(x_i), \lambda = 0.5 \tag{12}$$

The shared component is sampled once for each experimental seed and captures structure that can be transferred across tasks, whereas every task receives an independent task-specific component that must be identified from Ψ_τ. For simplicity, we choose a weighting of 0.5 in the convex combination to balance these two sources of variation.

We instantiate both components as one-hidden-layer random-feature networks,

$$f(x) = \sum_{m=1}^{h} a_m \tanh(w_m x + b_m), \qquad h = 32, \tag{13}$$

where $w_m \sim \mathcal{N}(0, 3^2)$, $b_m \sim \mathcal{N}(0, 1)$, and $a_m \sim \mathcal{N}(0, 1/h)$. This construction produces a diverse family of smooth, nonlinear reward landscapes.

Observed rewards are stochastic:

$$r_{\tau i} = \mu_\tau(i) + \epsilon_{\tau i}, \qquad \epsilon_{\tau i} \sim \mathcal{N}(0, \sigma_{\tau i}^2), \qquad \log \sigma_{\tau i} \sim \mathcal{N}(\log 0.1, 0.25^2). \tag{14}$$

The observation noise prevents the learner from treating individual support rewards as exact function values. Allowing the noise level to vary across task-arm pairs also models the fact that evaluations of different candidate trajectories may have different levels of uncertainty.

For each task, the arms are randomly permuted, and the first p arms form the support set $\Psi_\tau = \{(x_i, r_{\tau i})\}_{i=1}^{p}$. Thus, support arms are sampled uniformly without replacement.

Each seed contains 4,000 training tasks, 1,000 validation tasks, and 4,000 held-out test tasks. Together, these choices produce a controlled but nontrivial benchmark in which success requires both cross-task transfer and task-specific adaptation, while the support-set size p directly controls the amount of deployment-time information available.

Evaluated Methods. Next we discuss different methods that we compare with as a baseline in our evaluation for this task.

Random: Selects an arm uniformly, $\pi_{\text{rand}}(i) = 1/k$.

Population Reward: Pools the partial observations from the training tasks to estimate an arm-wise mean,

$$\widehat{\mu}_{\text{pop}}(i) = \frac{\sum_{\tau \in \mathcal{T}_{\text{train}}} \sum_{(j,r) \in \mathcal{D}_\tau} \mathbb{I}[j = i] r}{\sum_{\tau \in \mathcal{T}_{\text{train}}} \sum_{(j,r) \in \mathcal{D}_\tau} \mathbb{I}[j = i]}. \tag{15}$$

This baseline transfers population-level reward structure but does not condition on feedback from the held-out target task.

Task Specific Reward Fitting: This approach frames multi-task reward modeling through the lens of meta-learning, directly mirroring how we leverage support sets to infer specific user rubrics in Section 2. Learning a shared prior from task populations to accelerate reward inference has been broadly explored in multi-task and pre-trained Gaussian processes (Swersky et al., 2013; Wang et al., 2024b; Rasmussen & Williams, 2006) as well as meta-learning priors for bandits and regret minimization (Kveton et al., 2021; Azizi et al., 2023; Simchowitz et al., 2021; Hsieh et al., 2023).

For recovering the reward function, we fit a Gaussian prior over the distribution of rewards for each arm for a bandit task using a fixed support set Φ in which each task contributes $q = 10$ observed arms. We estimate the prior mean as a pooled arm-wise estimate $\widehat{\boldsymbol{\mu}}_{\text{pop}}$. We estimate the residual covariance $\widehat{C}$ from all pairs of arms that are jointly observed in the training tasks. To ensure this empirical estimate is well-conditioned, we regularize it toward a structured radial basis function (RBF) kernel to produce our final prior covariance $\widetilde{C}$. The regularization weight, kernel hyperparameters, and scalar observation-noise variance σ_ϵ^2 are selected exclusively on validation tasks by leave-one-out

Gaussian predictive likelihood. The resulting mean, covariance, and noise estimate are then frozen for the entire test evaluation.

For a held-out task with observed support set Ψ_τ, Gaussian conditioning gives

$$\widehat{\boldsymbol{\mu}}_\tau = \widehat{\boldsymbol{\mu}}_{\text{pop}} + \widetilde{\boldsymbol{C}}_{:,\Psi_\tau} \left(\widetilde{\boldsymbol{C}}_{\Psi_\tau,\Psi_\tau} + \sigma_\epsilon^2 \boldsymbol{I} \right)^{-1} \left(\boldsymbol{r}_{\tau,\Psi_\tau} - \widehat{\boldsymbol{\mu}}_{\text{pop},\Psi_\tau} \right). \tag{16}$$

The same frozen prior is used for every test-time support size p, including $p = 1$; the method is not refit or otherwise specialized for individual points on the evaluation curve.

Test-Time-Only Reward Fitting: As a reward-inference baseline without population transfer, we perform zero-mean Gaussian-process regression using only the held-out task's support set and a fixed RBF kernel,

$$k_{\text{RBF}}(x, x') = \exp\left(-\frac{(x - x')^2}{2\ell^2} \right), \qquad \ell = 0.45, \tag{17}$$

with ridge 0.01. This baseline imposes smoothness but learns neither a population mean nor a covariance from the training tasks.

Direct Prediction: A permutation-invariant set encoder maps the same test-time support set Ψ_τ of $(x_i, r_{\tau i})$ pairs directly to logits over the k arms. We train a separate predictor for each support size p. The *seen-arm* variant is supervised by the arm with the highest noisy reward among the observed arms. This matches the optimization procedure of the policy gradient algorithm ReST (Gulcehre et al., 2023). We also define a stronger *oracle* variant, which is supervised by the true optimum arm (arm with highest expected reward),

$$i_\tau^\star = \arg\max_{i \in [k]} \mu_\tau(i), \tag{18}$$

and therefore receives privileged labels during training. At test time, both variants receive only the same p noisy rewards as the reward-based methods. We retain dropout for 20 forward passes and average their argmax votes.

Evaluation. Reward estimates are converted to policies by a tie-weighted argmax, emulating the test-time inference strategy of best-of-n sampling (Snell et al., 2024). We report simple pseudo-regret,

$$R_\tau(\pi) = \max_{i \in [k]} \mu_\tau(i) - \mathbb{E}_{\hat{i} \sim \pi(\cdot|\Psi_\tau)} \left[\mu_\tau(\hat{i}) \right], \tag{19}$$

and its range-normalized counterpart. Both metrics are computed from latent mean rewards rather than noisy observations.

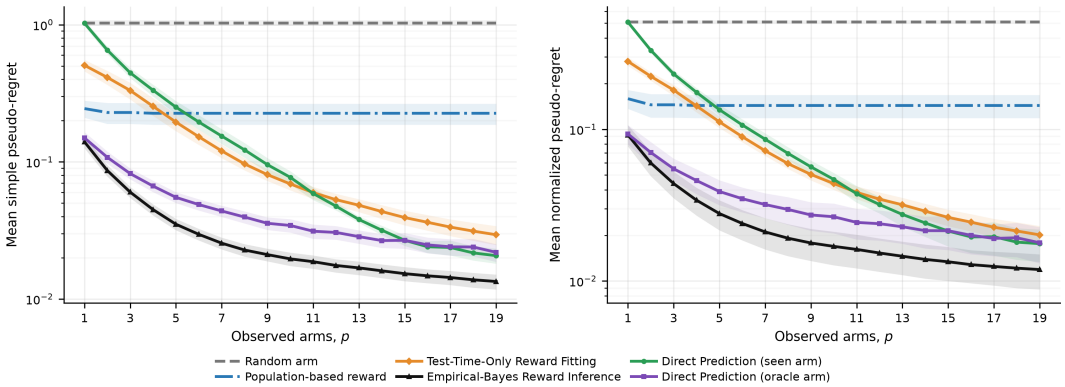

Figure 7: **Reward inference (empirical Bayes) improves adaptation to unseen tasks in a multi-task bandit setting.** Simple pseudo-regret (left) and range-normalized pseudo-regret (right) as the number of observed arms p increases; lower is better.

Across the number of observed arms, task-specific reward inference achieves lower raw and normalized pseudo-regret than both direct predictors. This comparison is deliberately favorable to direct

prediction: the oracle model is trained with the true optimal arm for every training task, which is not provided to the task-specific reward fitting. The reward model nevertheless benefits from a more informative intermediate representation of the objective. Rather than compressing the limited support set Ψ_τ immediately into a singular action policy, it uses each observed reward to update the estimated values of all related arms.

The population and test-time-only baselines isolate the two ingredients of this advantage. The population mean transfers shared structure but cannot adapt to the current task, whereas the test-time only reward fitting function adapts without learning the population's underlying reward statistics. The task-specific reward fitting model combines both: it learns a reusable function-space prior and conditions that prior on the current support set. Because the true rewards are generated by random neural networks rather than by the RBF model, the result does not rely on knowing the correct parametric family. This result is consistent with our hypothesis in Section 2: an explicit reward estimate yields lower regret than direct action prediction.

The posterior over task rewards after observing the task data the action minimizing the expected simple regret is given by the Bayes decision rule:

$$\hat{a}^* = \arg\max_a \mathbb{E}[R_\tau(a)|\Psi_\tau] \tag{20}$$

Thus, in the ideal infinite-data/infinite-capacity limit, there is no fundamental reason that explicitly reconstructing the reward vector must beat direct policy inference. So reward inference and direct action prediction are not fundamentally different in the Bayes-optimal limit; both can represent the same decision rule. Our hypothesis is that explicitly modeling the task reward provides a better finite-sample inductive bias when rewards possess reusable cross-task structure. Figure 7 shows evidence for this finite-data hypothesis.

Reward inference yields lower regret than direct action prediction.

In this controlled bandit setting, optimizing for an inferred reward function yields lower regret than predicting the optimal action directly from the task's support set, even when the direct predictor receives privileged training labels.

A.10 Understanding the role of the desiderata via a didactic example

We identify three key properties of a desirable rubric based reward functions that steers the downstream policy to producing high quality candidate responses: (1) accurate credit assignment, (2) generalization of the reward function beyond the train set, and (3) a dense reward signal.

To make these desiderata concrete, consider Wordle (Bertsimas & Paskov, 2022), where players have six guesses to identify a hidden five-letter word using feedback on letter presence and position (indicated by color). The oracle reward function would evaluate a candidate guess based on how efficiently it uses this feedback to narrow down the remaining pool of valid hidden words. This oracle perfectly satisfies our three requirements: it correctly orders guesses by their information gain or the number of valid candidates it eliminates. This applies universally to any valid board state.

We can clearly see the downstream failure modes when we ablate each desideratum in isolation as seen in Figure 8: *(1) Without accurate credit assignment*, we might instead reward a guess purely based on the number of green (correct letter + position) or yellow (correct letter + incorrect position) tiles it produces. While dense and extrapolatable, this misattributes quality. A redundant guess reusing known letters might score higher than a strategic guess that tests five new letters and eliminates thousands of candidate words, ultimately encouraging the wrong policy behavior. *(2) Without generalization*, we might use a lookup-table judge that provides the oracle score for common early-game openings but outputs a default score for unfamiliar states. It performs strongly on the initial distribution but as the policy improves and reaches novel, complex late-game scenarios, the reward model fails to generalize, preventing genuine improvement beyond the initial training data. *(3) Without a dense reward signal*, the reward becomes strictly binary: 1 for guessing the exact hidden word, and 0 otherwise. While accurate and generalizable, this approach assigns the exact same value to a strong guess that leaves only one possible word remaining and a terrible guess that reveals nothing. Thus, the agent has to expend compute to guess words until it gets the word correct.

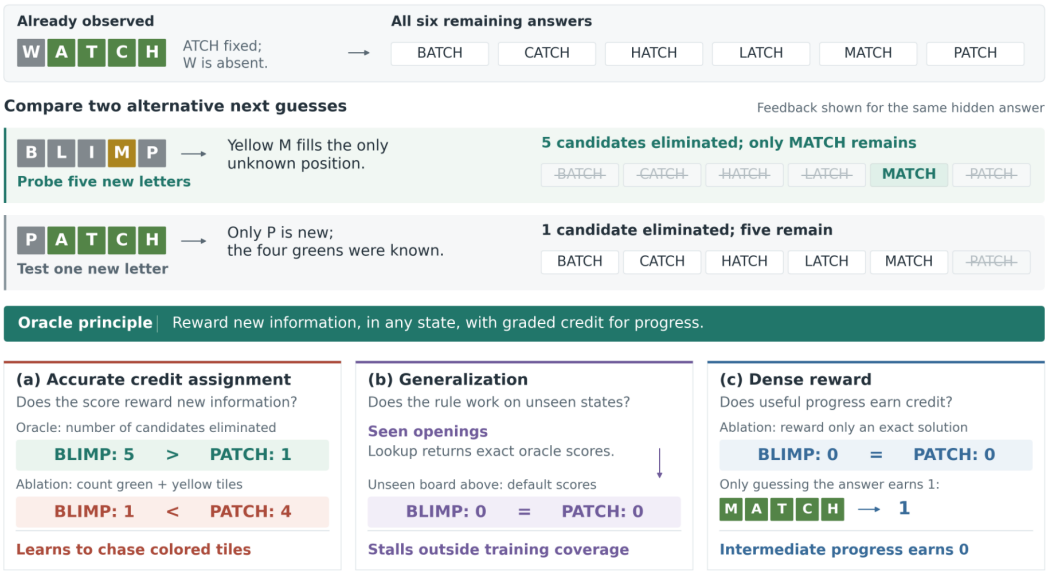

Figure 8: **Wordle Didactic Example.** We study a wordle game, where the target word is `MATCH`. Here, we show the benefit of accurate credit assignment, generalization, and a dense reward function.

A.11 Dataset Details

We evaluate Meta-Rubrics on two deployment-time adaptation domains: user simulation and data curation. In both domains, a task corresponds to an unseen environment μ with limited support information Ψ_μ and held-out query instances. The support set is used to infer a task-specific rubric, while query instances are used only for candidate scoring, reward-learning supervision, and final evaluation. All evaluation is done using held out instances. The training candidates are constructed by the base model by either directly sampling or via candidate augmentation and are labeled with the training judge model.

Amazon Reviews user simulation. Our user-simulation domain builds on the HumanLM Amazon review setting (Wu et al., 2026). We extend the original book-focused setup by applying the HumanLM preprocessing pipeline to a broader subset of Amazon Reviews 2023 (Hou et al., 2024), including categories such as fashion, appliances, electronics, beauty, and books. After filtering, the paper split contains 2,160 users: 2,000 for training, 60 for validation, and 100 for test. Each task is a single reviewer. The support set contains 8 prior reviews from that reviewer, including sanitized product context, review text, and rating metadata; the query consists of a held-out product for which the model must generate the review that the same user would likely write. Evaluation follows HumanLM and uses an LLM similarity judge as a proxy for agreement with the hidden target review. User and target-product identities are disjoint across the held-out split, so performance measures adaptation to new reviewer preferences rather than of observed products or users.

Data curation. In the data-curation domain, each task specifies a target downstream benchmark or domain together with a fixed fine-tuning token budget. The agent selects training examples from a candidate source pool, and the selected examples are used to fine-tune a small target model, `HuggingFaceTB/SmolLM2-135M` (Allal et al., 2025). Gold rewards are computed from the downstream improvement in held-out negative log-likelihood or perplexity after fine-tuning. The source pool is derived primarily from the Nemotron Cascade mid-training corpus (Yang et al., 2026b). We curate a full set of 10,000 tasks, with 100 evaluation tasks. Evaluation tasks are drawn from 21 suites held out during training. The evaluation suites include the MMLU family (MMLU (Hendrycks et al., 2021), Global MMLU (Singh et al., 2025), MMLU-Redux in Spanish (Gema et al., 2025), MMLU-ProX (Xuan et al., 2025), and MMLU-SR (Wang et al., 2024a)); BIG-bench (BIG-bench authors, 2023), including BIG-Bench Hard (BBH) (Suzgun et al., 2023); LAMBADA and its multilingual variants (Paperno et al., 2016); MGSM (Shi et al., 2023) and AfriMGSM (Adelani et al., 2025); AfroBench (Ojo et al., 2025); the AI2 Reasoning Challenge (ARC) (Clark et al., 2018); MLQA (Lewis

et al., 2020); MultiBLiMP (Jumelet et al., 2026); Okapi (Lai et al., 2023); and XQuAD (Artetxe et al., 2020). Candidate selections are truncated to the task token budget before fine-tuning, and target evaluation examples are kept disjoint from the selectable source pool. The support set contains 3 prior curation samples from the benchmark with their corresponding reward score (relative perplexity reduction). The model has access to a search tool, which it can use to search the curation corpus backed by the Mini-LM embedding model. We measure two representative degenerate baselines MiniLM embedding top-2048 selection 0.5445 ± 0.0780 and random selection 0.2478 ± 0.0526, which the support-conditioned policy exceeds with a single sample (0.68), while the base policy roughly matches it (0.55). We include an example task and curation sample below.

Representative Data-Curation Task: Python Code Understanding

```
[system]
You are a **training-data curator**. A small base language model needs to do
better on a **target task**, and you must choose which examples from a large
pool of candidate training examples it should be fine-tuned on.

The base model will be LoRA-fine-tuned on exactly the examples you submit,
then its **perplexity on held-out target data** is measured. Your reward is how
much that perplexity improves. Choosing examples that match the target's
language, domain, format, and difficulty helps; wasting the budget on
irrelevant examples does not.

You can explore the pool with tools:
- `search(query, k)` -- find up to `k` pool examples relevant to a query;
  returns each example's `[id] (label) snippet`.
- `show(id)` -- print the full content of one example.
- `token_count(ids, budget)` -- how many **training tokens** a set of IDs would
  use, so you can stay within your token budget.

Guidance:
- Use `search` and `show` to explore the pool and `token_count` to check sizes.
  Your selection must fit the **token budget** stated in the task.
- Do not solve the target task yourself -- only select training data for it.

When you are done, output your final selection as a single line of
space-separated pool IDs wrapped in tags, e.g.:

`<submit>1234 5678 9012</submit>`

Only the IDs inside the last `<submit>...</submit>` count, and the selection
must fit the token budget.

[user]

Examples of the target data:
--- example 1 ---
User:
Python code:
def natural_sum_recursive(n):
    if n <= 1:
        return n
    else:
        return n + natural_sum_recursive(n-1)

num = 16
print(natural_sum_recursive(num))

  choice: prints the first 16 natural numbers
  choice: returns natural_sum_recursive
  choice: prints string natural
  choice: prints the sum of first 16 natural numbers

English language description:
Assistant:
prints the sum of first 16 natural numbers
--- example 2 ---
User:
Python code:
fruits = ['Apple','Orange','Guava','Banana']
fruits.insert(1, 'Grape')

  choice: inserts 1 in list fruits
  choice: returns a list of fruits
  choice: creates a list of apples
  choice: inserts Grape in list fruits in second position
```

```
English language description:
Assistant:
inserts Grape in list fruits in second position
--- example 3 ---
User:
Python code:
print([x for x in range(1,531)])

  choice: prints 'hello world' to the terminal
  choice: prints values from 1 to 10
  choice: computes first 10 prime numbers
  choice: prints a list of first 530 natural numbers

English language description:
Assistant:
prints a list of first 530 natural numbers

Token budget: your fine-tuning set may use at most ~65536 training tokens
(call token_count to check). Explore the pool with search/show, then submit
your final curation as described in the instructions.
```

Sample Curation Rollout (Abridged)

```
[assistant/analysis]
Let me understand the target task. Looking at the examples, this is a
multiple-choice question task where the goal is to describe what a piece of
Python code does. The format is:
- User gives Python code and multiple choices
- Assistant provides the correct description

So I need to find training examples that teach the model to:
1. Understand Python code
2. Generate a concise natural language description of what the code does
3. Choose from multiple options

Let me search for relevant examples in the pool. I should search for things
related to Python code descriptions, code understanding, or similar tasks.

[assistant -> search]
{"k": 20, "query": "Python code description what does it do"}
{"k": 20, "query": "code explanation natural language description"}
{"k": 20, "query": "Python function algorithm explanation"}

[tool/search; excerpt]
[56580] (code:python) assi: def f(n): """Implement the function f that takes n
as a parameter, and returns a list of size n ...
[56576] (code:python) assi: def choose_num(x, y): """This function takes two
positive numbers x and y and returns the biggest even integer number that is in
the range [x, y] inclusive ...
[2784] (nemo:science) ... Consider the following Python code snippet ...
[2424] (nemo:science) ... Consider the following x86 assembly code snippet ...

[... additional search calls and results omitted ...]

[assistant/analysis]
I'm not finding examples that match the exact target task format (with User,
Python code, choice: lines, English language description, Assistant). Let me
try to get examples directly from IDs that might match the target task format.

[assistant -> show]
{"id": "56516"}
{"id": "9485"}
{"id": "2784"}
{"id": "2424"}

[tool/show; id 56516]
[assistant]
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [(e + 1) for e in l]

[... three show results omitted ...]

[user/evaluator]
```

```
The tool phase is now closed. Do not emit, describe, or request any tool call.
Using only pool IDs already visible in the trajectory, output your final
selection now as exactly one <submit>id1 id2 ...</submit> line. Put the
strongest IDs first and include a ranked reserve beyond the token budget.
Overage is safe: training truncates at the exact token cap and ignores the
remaining tail.

[assistant/final; first submission abridged]
<submit>56516 56534 56537 56564 56576 ... 56187 56361</submit>

[rollout statistics]
valid: true
candidate_nll_delta: 0.3654105137
submitted IDs: 621 unique (622 positions)
effective IDs consumed at the token cap: 510
training tokens: 65,536 / 65,536 (100.0%)
tool calls: 22
trajectory events: 35
```

A.12 Teacher Student Alignment

Because our method includes a distillation phase from a listwise teacher to a pointwise student, we measure their consistency after training on a held-out set of user–book pairs. As shown in Figure 9, the teacher and student achieve a spearman correlation of 0.95 and rmse of 0.06, indicating strong rank alignment after training. In this phase, the rubric writer and judge models are held frozen and sampled on the same set of training examples.

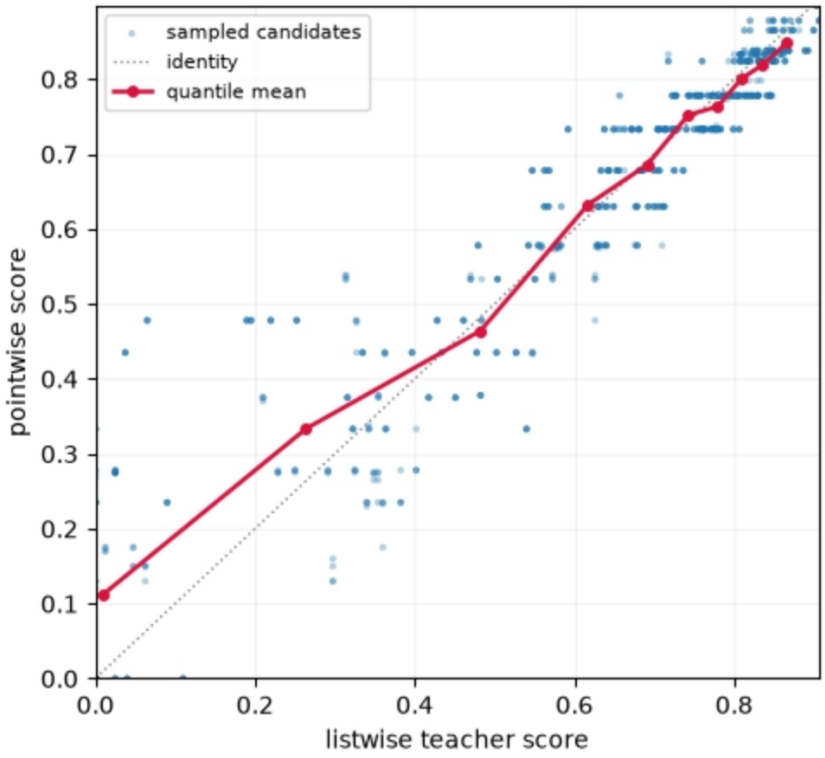

Figure 9: **Teacher–student consistency.** Rank consistency between the listwise teacher and the distilled pointwise student on held-out user–book pairs.

A.13 Rubric Shuffling Ablation

To validate that the rubric carries task-specific information, we setup an ablation where we score candidates using another user's rubric to confirm the efficacy of the rubric writer representation. We find a drop in normalized similarity score when shuffling rubrics from a different task. We used the same cached rubrics for evaluating both `Meta-Rubrics` and the shuffled variant.

Table 5: **Rubric Shuffling on Amazon Reviews.** We use `Meta-Rubrics` to generate rubrics for one (user, product) pair and use it review another (user, product) pair.

Method	1	2	4	8	16	32
Ours (Support)	0.35 (0.022)	0.38 (0.025)	0.39 (0.027)	0.40 (0.028)	0.41 (0.028)	**0.43** (0.021)
Ours (Support) - Shuffled	0.35 (0.022)	0.37 (0.024)	0.37 (0.025)	0.37 (0.026)	0.37 (0.026)	0.37 (0.026)
Oracle (Support)	0.35 (0.022)	0.43 (0.025)	0.49 (0.027)	0.54 (0.027)	0.58 (0.026)	0.62 (0.025)

A.14 Correlation of Baselines during Fine-Tuning

Table 6: **Correlation between training and oracle scores during multi-LoRA at step 10.** Both score vectors are centered within each eight-response group and pooled across 512 responses from eight pairs per method. Oracle correlation is 1 by construction.

Method	LLM Judge	LLM Rubric	Rubric Dropout	RLER	Ours	Oracle Rewards
Pearson r	-0.03	0.15	0.19	0.16	**0.41**	1.00
Spearman ρ	-0.06	0.12	0.15	0.13	**0.36**	1.00

We also see that the rubric (untrained) has a slightly better spearman correlation than the direct judge, indicating the benefit of the rubric based decomposition.