Themis: Training Robust Multilingual Code Reward Models for Flexible Multi-Criteria Scoring

Anonymous authors
Paper under double-blind review

Abstract

Reward models (RMs) have become an indispensable fixture of the language model (LM) post-training playbook, enabling policy alignment and test-time scaling. Research on the application of RMs in code generation, however, has been comparatively sparse, with existing work largely focusing on execution feedback. This choice constrains post-training to optimizing functional correctness over self-contained executable code. In this work, we examine the training and evaluation of multilingual, multi-criteria code RMs. To this end, we first compile **Themis-CodeRewardBench**, a benchmark to evaluate code RMs across five preference dimensions (i.e., criteria) and eight programming languages, on which we profile 50+ code, math, and general-purpose RMs. Observing the limited proficiency of current RMs beyond scoring for functional correctness, we develop **Themis-CodePreference**, the largest open-source collection of code preferences to date (more than 350k preference pairs), and use it to train **Themis-RM**, a suite of multilingual code reward models for flexible multi-criteria scoring, ranging in size from 600M to 32B parameters. Our experiments and ablations demonstrate positive scaling trends, strong cross-lingual transfer when training on diverse preferences, and the importance of multi-criteria training for reliable code reward modeling.

1 Introduction

Language model (LM) post-training recipes commonly rely on extrinsic reward signals to score LM responses. Access to such a reward enables model developers to align model outputs to end-user preferences (Ziegler et al., 2019; Askell et al., 2021; Ouyang et al., 2022), improve the reliability of human data labeling (Yang et al., 2024e), and efficiently allocate compute via test-time scaling (Snell et al., 2024; Hassid et al., 2024). Further, leveraging extrinsic rewards for on-policy training pushes the performance frontier (Ahmadian et al., 2024; Guo et al., 2024b; Song et al., 2024; Tajwar et al., 2024) while unlocking superior generalization (Hayes et al., 2025; Kirk et al., 2024b; Chu et al., 2025) and enabling learning from negative outcomes (Feng et al., 2025c; Zhu et al., 2025b; Singh et al., 2024; Dong et al., 2023a).

In general-domain settings, extrinsic rewards are usually obtained from a specialized reward model (RM) (Su et al., 2025; Gunjal et al., 2025), owing to their flexibility across disparate criteria (Kim et al., 2024a) and their implicit regularizing effect in noisy and underspecified settings (Xiong et al., 2024; Swamy et al., 2025). Despite measures of code quality spanning multiple attributes such as correctness, wall time, memory efficiency, and security, post-training for code generation (He et al., 2025a; Lambert et al., 2024; Chen et al., 2025c) avoids RMs and resorts to runtime rewards grounded in execution (Liu et al., 2023a; Gehring et al., 2025; Le et al., 2022; Shojaee et al., 2023). In this work, we posit that existing code reward sources are inadequate and that robust multi-criteria code RMs are crucial for improved code generation post-training that accounts for non-functional code quality criteria and removes the bottleneck of procuring readily executable code.

The limitations of execution feedback. Input-output matching via test-case execution is the dominant paradigm for sourcing rewards in code generation post-training (Liu et al., 2025g; Shi et al., 2022; Cheng et al., 2025a). However, it is limited to scenarios where model developers can procure their own test-cases. A common strategy to sidestep this bottleneck involves synthetically generating test-cases using LMs (Lee et al., 2025; Zhou et al., 2025c). However, despite positive scaling characteristics (Ma et al., 2025b), test-case generation approaches often rely on circular consensus mechanisms (To et al., 2024; Chen et al., 2023b; Liu et al., 2024g; Xu et al., 2025d; Zhang et al., 2025d; Zeng et al., 2025; Wang et al., 2025g) that result in brittle

or erroneous test-cases (Wang et al., 2025e; Cao et al., 2025). What is more, test-case feedback is only possible with self-contained executable code: this is highly restrictive as it limits models' post-training exposure to code devoid of external dependencies, which is very rare (Yu et al., 2024a), i.e., it is unrepresentative of the natural code distribution "in the wild". It also prevents the use of (most) code in programming languages with module-level compilation units, such as `Rust`, `Go`, and `Swift`. Attempts at mitigation via synthetic repository-level test-case generation are limited owing to the considerable difficulty of the task (Mündler et al., 2024; Jain et al., 2025). Finally, test-case verification conventionally relies on well-formed inputs derived from user-specified input preconditions and can thus miss subtle interface contract violations (Lim et al., 2025).

Alternatively, prior work has sought to expand runtime reward acquisition via auxiliary information such as execution duration and data flow (Duan et al., 2023; Nichols et al., 2024; Shojaee et al., 2023). However, such rewards can be noisy unless the code execution occurs in a tightly controlled emulated environment (Shypula et al., 2024). Additionally, such rewards are often noisy proxies for an underlying objective and prone to reward hacking (Pan et al., 2022; Lehman et al., 2020; Gao et al., 2023). For instance, logic invoked in a compile-time macro can throw off the use of a compilation reward as a proxy for syntactic correctness. In contrast, RMs derived from massively pre-trained LLMs, owing to their extensive knowledge, can complement code execution signals by providing extrinsic rewards across a wide range of programming languages in underspecified scenarios (Everitt et al., 2017), and are not limited to functional correctness evaluation.

The drawbacks of LMs as surrogate code executors. Recent work has adopted the LM-as-a-judge approach (Liu et al., 2023b; Li et al., 2024b) to evaluate code via specialized prompting techniques (Aggarwal et al., 2024; Tong & Zhang, 2024; Zhuo, 2024), but general-purpose LMs tend to be poorly calibrated code evaluators in reference-free settings (Yadavally et al., 2025; Lyu et al., 2025). In parallel, attempts have been made to train (i) specialized LM scorers as code execution surrogates (Bruches et al., 2026), (ii) quality estimators (Ni et al., 2023; Shi et al., 2022), and (iii) re-rankers (Zhang et al., 2023b; Inala et al., 2022). However, these efforts primarily target self-contained `Python` code, a setting in which the execution-based feedback excels. Furthermore, prior work shows that when applied to low-resource programming languages, surrogate code execution models suffer from non-trivial performance drops (Lyu et al., 2025). We thus argue that the impact of post-training in code generation can be expanded via robust RMs capable of scoring code outputs across diverse scenarios, programming languages, and over disparate user-specified criteria.

The need for multilingual and multi-criteria code RM benchmarks. RM training losses are poor predictors of test-time regret (Fluri et al., 2025), necessitating the creation of challenging and realistic RM preference evaluation benchmarks (Lambert et al., 2025; Malik et al., 2025; Liu et al., 2025f; Tan et al., 2025; Frick et al., 2025). Exhaustive evaluation enables the flagging of weak RMs before deployment, thereby preventing downstream reward hacking (Xie et al., 2025). Yet the diversity and difficulty of code preferences in RM evaluation datasets are extremely limited, consisting entirely of valid-buggy `Python` solution pairs sourced from a select few elementary and dated competitive coding datasets (Austin et al., 2021; Chen et al., 2021; Muennighoff et al., 2024). Existing attempts at dedicated evaluations for general-domain code RMs are similarly limited in programming language and scenario coverage (Zhang et al., 2025a; Zhao et al., 2025; Jiang et al., 2025; Ficek et al., 2025). There is thus an acute need for code RM evaluation benchmarks that assess scoring across a varied mix of scenarios, in multiple programming languages, and on criteria beyond (just) functional correctness.

Contributions. We address the abovementioned research gaps and make the following contributions:

- We compile `Themis-CodeRewardBench`, an extensive code RM evaluation benchmark comprising ≈`8.9k` diverse code preferences across eight programming languages and five scoring dimensions. Subsequently, we study the failure modes of `45` existing code RMs (ref. Section 3).
- We develop `Themis-GeneralPreference` and `Themis-CodePreference`, two extensive preference datasets containing `110k+` general-domain and `350k+` code-domain preferences, respectively; the latter covers five different dimensions/criteria of code quality: functional correctness, execution efficiency, memory efficiency, readability and maintainability, and security.
- We use the above preference datasets to train `Themis-RM`, a suite of multilingual open-source code RMs ranging from `600M` to `32B` parameters in size, capable of flexibly scoring code across arbitrary (i.e., user-defined) subsets of the five quality criteria (ref. Section 4).

- We conduct an extensive investigation into the performance trends of the `Themis-RM` suite across model sizes, including on unseen criteria and programming languages (ref. Section 5). Our work constitutes the first test of the viability of using code RMs to score open-domain code under adversarial settings, i.e., on unseen criteria and out-of-distribution programming languages.

We structure our investigation into training state-of-the-art RMs for code post-training in Section 5 around the following set of research questions:

RQ1: How accurately can RMs score code across a set of diverse real-world code quality preferences?

RQ2: What is the best approach to minimize cross-criteria interference in multi-criteria code RMs?

RQ3: How well does preference learning for code RMs transfer to unseen programming languages?

RQ4: Do RMs trained on multi-criteria preferences exhibit the requisite adversarial robustness and listwise re-ranking competence necessary for downstream applications such as post-training and search?

2 Related Work

We briefly summarize four lines of relevant existing research: **1.)** surrogate code execution modeling for alignment and quality estimation, **2.)** extrinsic rewards for code beyond functional correctness, **3.)** multi-criteria reward models, and **4.)** reward model evaluation benchmarks.

Surrogate code execution modeling for alignment and quality estimation. LM developers seeking to ensure quality control of model-generated code have traditionally relied on test-case feedback. However, in addition to restrictive executability requirements, existing approaches in this mold require users to either bring their own test-cases (Liu et al., 2023a; Gehring et al., 2025; Tao et al., 2025), generate synthetic test-cases (To et al., 2024; Chen et al., 2023b; Zhang et al., 2025d; Liu et al., 2024g; Wei et al., 2024a) or source reference solutions (Wang et al., 2024d). Consequently, recent work has looked to repurpose neural surrogates (a.k.a. verifiers) for code execution simulation (Hesthaven & Ubbiali, 2018; Lu & Ricciuto, 2019; Yan et al., 2020; Rodionov & Prokhorenkova, 2023; Lyu et al., 2025) as a means of scoring code that is not readily executable. Early work in this vein trained quality estimators to simulate test-case execution (Ni et al., 2023; Zhou et al., 2023; Inala et al., 2022), but the resultant models can be poorly calibrated with respect to program semantics in reference-free settings (Shi et al., 2022).

Subsequent work has leveraged pairwise preference learning (Ouyang et al., 2022; Ziegler et al., 2019) to improve robustness to noisy signals. Approaches in this paradigm have bootstrapped scalar RMs from execution feedback (Zeng et al., 2025; Zhu et al., 2026) as well as synthetic feedback (Weyssow et al., 2024). Pairwise RMs (Jiang et al., 2023; Whitehouse et al., 2025; Chen et al., 2025a; Yu et al., 2025; Wang et al., 2025f; Liu et al., 2025h; 2024b) evolve this approach to its logical end by operating on pairs of code responses at both training and inference time. However, pairwise RMs can produce inconsistent and intransitive preferences (Liu et al., 2024f), which demand specialized prompting techniques in listwise re-ranking tasks (Zhu et al., 2024; Qin et al., 2024). In parallel, attempts have been made to leverage the commonsense knowledge of frontier LMs (Zhao et al., 2023) to provide free-text feedback that reasons about code semantics (Dong et al., 2025; Zhuo, 2024; Moon et al., 2023). The range of such textual feedback spans iterative self-reflection (Shinn et al., 2023; Madaan et al., 2023), refinement proposals (Zhou et al., 2025a; Zhang et al., 2023a; Chen et al., 2023a), deliberating compiler outputs (Gupta et al., 2020; Gou et al., 2024; Wang et al., 2022; Chen et al., 2024a) and studying execution traces (Ni et al., 2024a). However, as we illustrate in Section 5.1, generation-based surrogate code executors can exhibit low scoring fidelity, impeding their ability to resolve fine-grained preferences. Our experiments with `Themis-RM` show that scalar RMs trained on diverse code preferences can accurately score code across a variety of domains (ref. Section 5.1), while maintaining high scoring fidelity, thus enabling listwise re-ranking in challenging settings (ref. Section 5.4).

Extrinsic rewards for code beyond functional correctness. Owing to the difficulty of multi-objective optimization of LMs (Nguyen et al., 2025; Yang et al., 2024a;d; Zhou et al., 2024; Li et al., 2021; Dai et al., 2024) and its occasionally unclear tradeoffs for code generation (Coignion et al., 2025), LM post-training has largely avoided improving code quality along non-functional axes. However, as LM-generated code proliferates (Daniotti et al., 2025), the potentially dire security implications of synthetically generated code smells (Paul et al., 2025; Pearce et al., 2025; Kharma et al., 2025) have led to a renewed interest. Prior

work on enhancing the security of LM code outputs has incorporated static analyzer feedback (Blyth et al., 2025; Siddiq & Santos, 2022; He & Vechev, 2023; Sijwali & Saha, 2026). But static analyzers broadly rely on predefined surface-level patterns in code and cannot catch vulnerabilities that require dependency context and runtime information (Cui et al., 2024). Accordingly, recent work incorporated LM-based feedback to mine security preferences (Weyssow et al., 2025; Xu et al., 2025a), constrain inference-time decoding (Qu et al., 2025; Wang et al., 2025b; Fu et al., 2024; Li et al., 2024a), or directly score fine- (Quan et al., 2026) and coarse-grained (Wu et al., 2026; Ding et al., 2025; Li et al., 2025d; Islam et al., 2024) security hardness. In tandem, the reasoning capabilities of LMs have been leveraged towards iterative code hardening (Liu et al., 2025b; Zhang et al., 2024a; Le et al., 2024), security-focused synthetic environment generation (Zhuo et al., 2025), and in-context learning of secure coding principles (Zhang et al., 2024b; He et al., 2024).

Similarly, the scarcity of work optimizing wall-clock metrics has led to LM-generated code lagging behind developer-written code in efficiency (He et al., 2025b; Fan et al., 2025; Ma et al., 2025a). Attempts to address this gap via post-training have resorted to retrieval-augmented generation (Wu et al., 2025b), compiler feedback (Lamouri et al., 2025), or test-case execution feedback (Feng et al., 2025b; Waghjale et al., 2024). These efforts are, however, impeded by the need for executable code, pre-existing test-cases, and/or reference solutions. More recent approaches derive extrinsic rewards by generating synthetic test-cases (Ye et al., 2025), using scalar reward models (Nichols et al., 2024), or directly from single- or multi-step improvements via textual reasoning (Huang et al., 2025a; Yang et al., 2025b; Peng et al., 2025; Du et al., 2025; Huang et al., 2024). The use of extrinsic signals to improve code memory efficiency (Rajput et al., 2026), maintainability (Zhang et al., 2025b; Nunes et al., 2025), and specification following (Ma et al., 2024) is even more sporadic. Our analyses with `Themis-RM` show that non-functional requirements do not necessarily interfere with functional correctness in reward modeling, and that specific design choices during training can enable scalar RMs to model preferences across multiple axes (ref. Section 5.2).

Multi-criteria reward models. Preference learning for RMs has typically been framed as the Nash equilibrium of an implicit two-player game (Jacob et al., 2024; Wang et al., 2023; Munos et al., 2024). This formulation implies that training scalar RMs via naive multi-task learning (Vu et al., 2024; Yang et al., 2024b; Wu et al., 2025a) across multilingual, multi-task, and personalized preferences is bound to incur interference (Chakraborty et al., 2024) and distributional biases (Christian et al., 2025). Prior attempts to mitigate such interference involve model merging (Yang et al., 2025a; Jang et al., 2023; Kim et al., 2024b), factorized representations (Shenfeld et al., 2025; Calderon et al., 2025), steering vectors (Lin et al., 2025a), Bayesian optimization (Winata et al., 2025) and specialized neural layers (Xie et al., 2026; Wang et al., 2024c). Subsequent work has demonstrated that detailing diverse preference scenarios as textual contexts allows scalar (Lee et al., 2024; Sun et al., 2024) and implicit RMs (Wang et al., 2024b; Dong et al., 2023b; Wang et al., 2025c) to effectively disentangle them (Zhang et al., 2025f).

Additionally, the body of work on generative (Kwon et al., 2023; Bai et al., 2022; Zhu et al., 2025a; Zhang et al., 2025c; Zhong et al., 2022; Liu et al., 2024d; Jiang et al., 2024) and reasoning RMs (Chen et al., 2025a; Guo et al., 2025; Liu et al., 2025h) also demonstrates the benefits of incorporating textual evaluation criteria via rubrics, towards both preference learning and downstream optimization Anugraha et al. (2025); Yu et al. (2025); He et al. (2025c); Liu et al. (2025d); Huang et al. (2025c). More recent work outlines how the general knowledge of LMs can be used to self-propose evaluation criteria in underspecified scoring settings (Chen et al., 2025b; Saha et al., 2025; Saad-Falcon et al., 2025; Zhou & Tan, 2026; Ankner et al., 2024). Our evaluations with `Themis-RM` show that modeling evaluation criteria via constituting principles enables scalar RMs to effectively model multi-dimensional preferences with minimal interference (ref. Section 5.2).

Reward model evaluation benchmarks. Identifying coverage shortcomings (Wang et al., 2024a) and distributional biases (Li et al., 2025e; Kirk et al., 2024a) of RMs before their downstream deployment is crucial for minimizing reward hacking at deployment time (Xie et al., 2025; Pan et al., 2022). Early work in this direction outlined the strong correlation between RM accuracy in modeling preferences and downstream post-training performance (Frick et al., 2025). Accordingly, a significant body of work evaluates RMs on general-domain preferences (Tan et al., 2025; Zhou et al., 2025b; Lambert et al., 2025) as well as for pairwise rankings on a host of specialized domains such as instruction following (Wen et al., 2026), computer use (Lin et al., 2025b), multi-modal (Li et al., 2025b), and multilingual (Gureja et al., 2025) settings. More recent efforts benchmark the pairwise re-ranking abilities of code RMs by repurposing existing competition code

Criteria	Dataset	Short Description	Programming Language								Provenance	
			C	C++	C#	Go	Java	JS	Python	Ruby	Prompts	Responses
Functional Correctness (FC)	Commit Preference Correctness	Multi-LLM verified functionality bug-fixes expressed in GitHub commits.	11	25	42	59	134	241	216	97	LLM Generated	Human Written
	HumanEvalPack Muennighoff et al. (2024)	Manually injected bugs in translated HumanEval solutions.	✗	147	✗	117	132	113	119	✗	Human Written	Human Written
	MBPPPlusFix-Hard Prasad et al. (2025)	Discerning LLM generations that satisfy a high proportion, but not all the testcases.	✗	✗	✗	✗	✗	✗	37	✗	Human Written	Mixed
	MDEval Liu et al. (2024e)	Manually annotated buggy-fixed pairs inspired by GitHub code.	12	11	✗	24	✗	30	22	35	Human Written	Human Written
	DebugEval Yang et al. (2025c)	DebugBench and LiveCodeBench buggy-fixed pairs (LeetCode).	✗	211	✗	✗	194	✗	319	✗	Human Written	Human Written
	RunBugRun-V1 Prenner & Robbes (2023)	User grouped fast-slow pairs from CodeNet (AtCoder and Aizu).	279	504	✗	352	453	164	305	376	Human Written	Human Written
Execution Efficiency (EE)	Commit Preference Runtime	Multi-LLM verified runtime improving code changes expressed in GitHub commits.	4	6	21	15	55	39	56	42	LLM Generated	Human Written
	Pie4Perf Shypula et al. (2024)	Emulator verified user-grouped curated pairs of slow-fast solutions from CodeNet.	✗	460	✗	✗	✗	✗	✗	✗	Human Written	Human Written
	ECCO Waghjale et al. (2024)	Execution verified curated pairs of slow-fast solutions from CodeNet.	✗	✗	✗	✗	✗	✗	399	✗	Human Written	Human Written
	EvalPerf Liu et al. (2024c)	LLM completions of varying efficiency to EvalPlus.	✗	✗	✗	✗	✗	✗	212	✗	Human Written	LLM Generated
Memory Efficiency (ME)	Commit Preference Memory	Multi-LLM verified memory usage improving code changes expressed in GitHub commits.	73	43	8	12	52	28	26	10	LLM Generated	Human Written
	NoFunEval Memory Singhal et al. (2024)	Human validated mined memory efficiency-enhancing commits.	2	✗	✗	✗	35	✗	✗	✗	LLM Generated	Human Written
Readability And Maintainability (R&M)	Commit Preference CodeStyle	Multi-LLM verified code stylistic improvements expressed in GitHub commits.	13	24	80	101	257	325	365	206	LLM Generated	Human Written
	NoFunEval Maintain Singhal et al. (2024)	CodeQL-verified maintainability fixes linked to GitHub commits.	✗	✗	✗	✗	✗	✗	128	✗	LLM Generated	Human Written
Security Hardness (SH)	Commit Preference Security	Multi-LLM verified vulnerability fixes expressed in GitHub commits.	31	42	37	46	143	144	172	206	LLM Generated	Human Written
	CodePrefBench Security Liu et al. (2024b)	Synthetically fixed CyberSecEval samples using LLMs.	✗	✗	✗	✗	✗	✗	173	✗	LLM Generated	Mixed
	Vul4J Bui et al. (2022)	Human and test validated commits mined from SAP's Project KB.	✗	✗	✗	✗	8	✗	✗	✗	LLM Generated	Human Written
	SecBench Jing et al. (2024)	Fusion of multiple security-fix commit datasets mined from CWEs.	2	1	✗	✗	4	✗	2	5	LLM Generated	Human Written
	NoFunEval Security Singhal et al. (2024)	Asleep at the Keyboard completions by CoPilot that trigger CodeQL rules.	12	✗	✗	✗	✗	✗	15	✗	LLM Generated	LLM Generated

Table 1: Criteria-level dataset composition of the Themis-CodeRewardBench reward model evaluation benchmark introduced in Section 3, broken down across the eight constituent programming languages. For detailed evaluation results on this benchmark, refer to Section 5.1 and Section 5.3.

generation benchmarks (Ficek et al., 2025; Zhao et al., 2025; Jiang et al., 2025; Ni et al., 2024b; Zhang et al., 2025a). Such benchmarks, however, risk testing RMs on a narrow distribution of prompts in a select few high-resource programming languages, and may be contaminated (Matton et al., 2024). We contribute Themis-CodeRewardBench (ref. Section 3), a diverse, multi-criteria, and multilingual code RM evaluation benchmark that enables us to uncover critical gaps in the coverage of current code RMs (ref. Section 5.1). Further work has outlined the importance of holistic profiling of reward models on adversarial (Venkatkrishna et al., 2026; Liu et al., 2025f; Moon et al., 2025) and listwise ranking settings (Malik et al., 2025; Wen et al., 2025; Kim et al., 2025) to detect over-optimization failure modes that preference-based accuracy evaluation can occasionally miss. Our robust evaluations on Themis-RM show how the advantages of open-domain code preference learning extend to such scenarios (ref. Section 5.4).

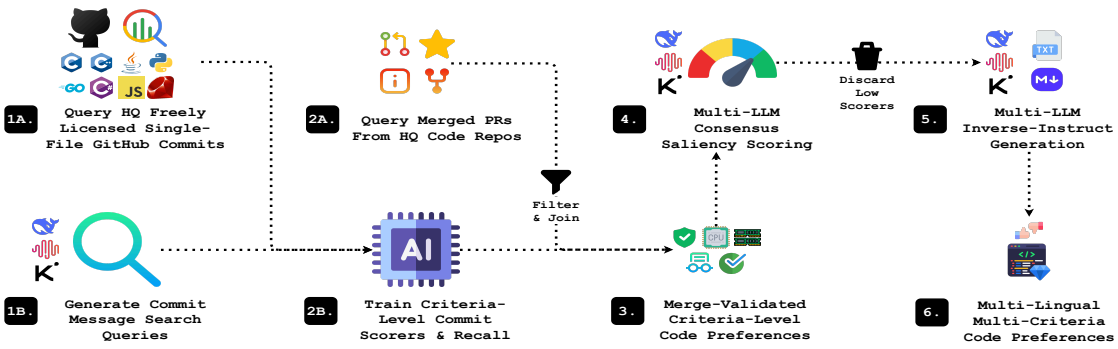


Figure 1: Overview of our pipeline for mining multi-programming-language multi-criteria code preferences from single-file merged GitHub commits. Used both in `Themis-CodeRewardBench` and for `Themis-CodePreference`.

3 `Themis-CodeRewardBench`: Multilingual And Multi-Criteria Code RM Evaluation

With the aim of thoroughly evaluating RMs across multilingual and multi-dimensional code preferences, we first curate the `Themis-CodeRewardBench` benchmark, a collection comprising `13` distinct pre-existing as well as newly constructed code preference datasets. Our multi-criteria benchmark evaluates RMs **(1)** on five code quality dimensions: `Functional Correctness (FC)`, `Execution Efficiency (EE)`, `Memory Efficiency (ME)`, `Readability And Maintainability (R&M)`, and `Security Hardness (SH)`, and (2) for eight high- and medium-resource programming languages: `C`, `C#`, `C++`, `Go`, `Java`, `JavaScript`, `Python`, and `Ruby`. Deferring to prior findings (Frick et al., 2025) and the strong precedent in existing RM evaluation, we use preference accuracy as the evaluation metric on `Themis-CodeRewardBench`.

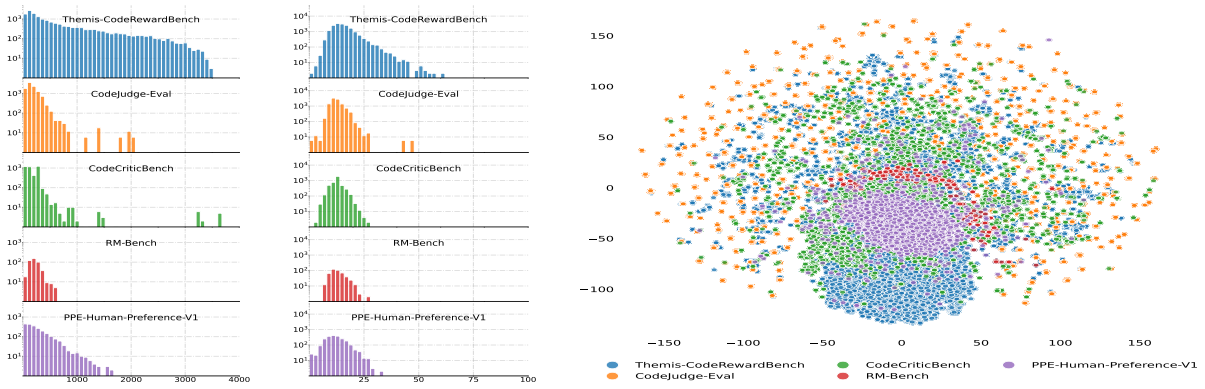
We start by procuring the test splits of diverse existing datasets that contain explicit or implicit expressions of code preference. Such datasets range from human-annotated code changes to execution-validated code metrics computed for human- or model-generated code. This curation yields pairs of code with disparate functional correctness (Muennighoff et al., 2024; Prasad et al., 2025; Liu et al., 2024e; Yang et al., 2025c; Prenner & Robbes, 2023), execution efficiency (Shypula et al., 2024; Waghjale et al., 2024; Liu et al., 2024c), security (Liu et al., 2024b; Jing et al., 2024; Bui et al., 2022), and code-style compliance or maintainability (Singhal et al., 2024). For datasets lacking explicit queries or instructions, we synthetically generate inverse instructions (see Section C.2 for the prompt). The resulting collection built from existing datasets, albeit varied in provenance, is rather thinly spread across most combinations of programming languages and evaluation criteria (especially for non-functional criteria). To address this limitation, we develop a multi-step data-collection pipeline to extract implicit code preferences from GitHub commits, as described next.

We begin our commit mining by querying the public GitHub archives[1] using a modified version of the pipeline detailed in Muennighoff et al. (2024) (see Section B.1 for the consolidated query), searching for single-file code changes in openly-licensed repositories. We then generate search terms for each of our five preference criteria (detailed in Section B.2) using frontier open-LMs (Kimi-Team, 2026; DeepSeek-AI, 2025b; MiniMax, 2025). We leverage samples retrieved with these search terms to train criteria-specialized ModernBERT (Warner et al., 2025) commit classifiers, which we use to recall high-confidence code changes corresponding to the criteria. Subsequently, we incorporate pull request data from reputable repositories[2] using `GHTorrent`[3] and only retain non-reverted commits authored between `June 2019` and `January 2021`, that are part of merged pull-requests, ensuring implicit human validation of intents. Next, we solicit a consensus of multiple frontier open-LMs to weed out multi-purpose code changes or changes without a clear intent (see Section C.1 for annotation prompt). Finally, we synthetically generate realistic inverse instructions across a multitude of styles for the remaining code-change pairs (we provide the prompt in Section C.2). Figure 1 illustrates this commit-mining workflow. We provide the complete dataset-, criteria-, and language-level make-up of `Themis-CodeRewardBench` in Table 1. Figure 2 compares the `Themis-CodeRewardBench` against code subsets

[1] `github/github-repos`

[2] We deem repositories with `15+` GitHub stars, `5+` contributors, and `10+` GitHub issues as a reasonable proxy for reputability and commitment to good software engineering practices.

[3] `ghtorrent/ghtorrent.org`



(a) Response code seq. length distribution.

(b) Response code AST depth distribution.

(c) 2D t-SNE plot of prompt embeddings, as generated by 🤗 `Alibaba-NLP/gte-Qwen2-1.5B-instruct` Li et al. (2023).

Figure 2: Comparison of `Themis-CodeRewardBench` against the code subsets of popular existing RM evaluation benchmarks. `Themis-CodeRewardBench` judges RMs on (a) longer and (b) more complex code responses, over a (c) largely novel distribution of prompts.

of existing RM benchmarks, showing that `Themis-CodeRewardBench` introduces a largely novel distribution of code preferences (Figure 2c), for code of increased complexity (Figure 2a and Figure 2b).

4 `Themis-RM`: Training Multilingual Criteria-Following Code RMs

To establish the benefits of diverse code-specific reward modeling, we train the `Themis-RM` suite of RMs. We train `Themis-RM` models in two stages: (1) a preference model pre-training (PT) stage, followed by (2) a preference modeling (PM) stage. In this section, we detail the training data and exact setup for both stages.

4.1 Training Data Mixture

Preference Model Pre-Training (PT): Scoring code on non-functional and stylistic axes demands a general understanding of human preferences and conventions. Hence, we first pre-train the `Themis-RM` suite to instill common human-inspired notions of preference evaluation such as relevance, helpfulness, and harmlessness. To this end, we curate `Themis-GeneralPreference`, a `110k+` sample mix of natural language and code preferences curated from popular existing preference and retrieval datasets (ref. Section B.3 for the detailed composition). This approach mirrors prior work, which demonstrated the benefits of training LMs for approximating open-domain human preferences (Askell et al., 2021; Wang et al., 2025a; Korbak et al., 2023).

Preference Modeling (PM): Our second training stage centers on `Themis-CodePreference`, a `350k+` preference dataset comprising preferences across the same five dimensions (i.e., criteria) of code evaluation and the eight programming languages that our `Themis-CodeRewardBench` benchmark evaluates. In the interest of training on diverse and complex scenarios, we source GitHub commit preferences and synthetic prompts using the pipeline described in Section 3. We ensure that all commits in our training data are pushed before `March 2019` and come from a disjoint set of repositories, vis-à-vis the commit data in `Themis-CodeRewardBench`. Having single-intent single-file-changing commits enables us to source preference pairs for each of the criteria of interest, which curtails training noise (Yang et al., 2024b; Shen et al., 2024; D'Oosterlinck et al., 2025) and impedes reward hacking (Kim et al., 2025). Additionally, acquiring criteria-specific commit preferences allows us to train `Themis-RM` models on non-functional preferences that are otherwise rare in traditional instruction-tuning datasets (Xu et al., 2025a; Koukoumidis, 2023).

Aiming to deliver complete and well-rounded RM models for code, we also, naturally, train `Themis-RM` on diverse multi-dimensional preferences—functional correctness, wall time, and memory usage preferences—sourced from a range of existing code-contest datasets (Puri et al., 2021; Waghjale et al., 2024; Du et al., 2025; Prenner & Robbes, 2023). Finally, we source a mix of functional and non-functional code preferences from LM-generated responses to user prompts. Training on synthetic data improves the utility of the `Themis-RM` suite in downstream post-training settings (Tajwar et al., 2024). We source model-generated preferences by synthetically introducing algorithmic and syntactic errors (i.e., bugs) into responses in existing instruction-

tuning datasets (Xu et al., 2025a; Zhu et al., 2026). Section B.3 details the criteria- and language-level composition of `Themis-CodePreference` and the data filtering steps.

4.2 Model Training

`Themis-RM` represents a suite of code-specialized RMs ranging from `0.6B` to `32B` parameters in size that we derive from Qwen3 (Team, 2025) dense models using the aforementioned two-phase training. Our training incorporates optional system prompts p that allow the specification of custom evaluation criteria, with `15%` of the training samples unaccompanied by any criteria (i.e., without p), a further `20%` accompanied by a generic formulation that outlines all the criteria, and the rest of the samples coupled with the evaluation instruction that captures exactly one criterion (ref. Section C.3 for the evaluation instructions). Our experimental results in Section 5.2 show that this prompt strategy effectively disentangles multi-criteria preferences, eliminating the need for criteria-specific modules (Wang et al., 2024c), ensembling (Coste et al., 2024), model-merging (Jang et al., 2023; Ramé et al., 2023), or data augmentation (Liu et al., 2025e; Shen et al., 2023; Wu et al., 2025d).

`Themis-RM` models are trained using the Bradley-Terry (Zhang et al., 2025f) reward modeling objective on preference tuples $(p, x, y_c, y_r) \in \mathcal{D}$ sourced from the aforementioned datasets. Owing to the largely offline nature of our training data acquisition, and the raised risk of over-optimization (Xu et al., 2025b), we adopt additional regularization for the RM hidden states. Prior work has regularized RM representations via a range of auxiliary information, including uncertainty (Lou et al., 2024; Shekhar et al., 2025; Cai et al., 2024), distributional data (Dorka, 2024), and adversarial generators (Bukharin et al., 2025). We opt for a minimally invasive conditional language modeling loss (Yang et al., 2024c; Hejna et al., 2023) over the preferred response y_c as our regularizer. Finally, owing to the poor calibration of scalar RMs beyond their normal scoring range (Bays & Dowding, 2017), we adopt a reward magnitude regularizer (Eisenstein et al., 2023). Our overall training objective is specified as follows:

$$\mathcal{L} = -\mathbb{E}_{([p],\, x,\, y_c,\, y_r)\sim\mathcal{D}}\Big[\log\sigma(r_\theta(p,x,y_c) - r_\theta(p,x,y_r)) + \lambda\cdot\log p_\theta(y_c \mid p, x) + \mu\cdot\left(r_\theta(p,x,y_c) + r_\theta(p,x,y_r)\right)^2\Big]$$

where $\sigma(\cdot)$: sigmoid function; r_θ: scalar RM; p: criteria prompt; x: task prompt; y_c: chosen response; y_r: rejected response.

For simplicity, we utilize the above objective in both training phases, which we verify is extremely effective (Yu et al., 2024b) (ref. Section 5.2) and removes the need for bespoke objectives for the PT phase (Dou et al., 2025; Askell et al., 2021). We train on `Themis-GeneralPreference` for two epochs during the PT phase and on `Themis-CodePreference` for one epoch during the PM phase (we train using a modified fork of the `trl`[4] framework). We leverage the AdamW optimizer (Loshchilov & Hutter, 2019) and a cosine scheduler with a `5%` warmup in each of the phases. We outline the complete training configuration and architectural heritage of the `Themis-RM` suite in Appendix A.

5 Experimental Research Questions And Results

5.1 RQ1: Judging Reward Models On Multilingual Multi-Criteria Code Preferences

We begin our inquiry with a rigorous evaluation of the competence of existing RMs in scoring code. To this end, we assess a diverse mix of the most competitive existing general-purpose, code-specialized, scalar, generative, and reasoning RMs and compare them against `Themis-RM` on `Themis-CodeRewardBench` (ref. Table 2). Our assessments render current RMs largely unusable for scoring code along non-functional axes such as efficiency and security, often degenerating to purely random scoring. `Themis-RM`, in contrast, displays a strong performance on all axes across all model sizes. Our smallest model, `Themis-RM-0.6B`, outscores multiple `>100x` larger general-purpose RMs, whereas our largest model, `Themis-RM-32B`, sets itself as the clear state-of-the-art (including on functional correctness). Within each evaluated model family (including `Themis-RM`), we observe a strong positive scaling trend in scoring for functional correctness. This trend is in line with prior findings on the correlation between RMs' verification and generation abilities (Zhou et al., 2025d), both conforming to the well-known scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022). Profiling the results for the functional correctness scoring (see Table 7), which is the usual operating mode for existing code RMs, across individual datasets uncovers glaring limitations of existing RMs. Concretely, most RMs fare poorly when required to grade code outside the common distribution of code contest data, trailing

[4] `huggingface/trl`

Auxiliary Training Objectives | Code RM | Criteria-Following RM | Math RM | Generative RM | Reasoning RM

	Model		Size	Average	Criteria-Level Accuracy				
					Functional Correctness (FC)	Execution Efficiency (EE)	Memory Efficiency (ME)	Readability And Maintainability (R&M)	Security Hardness (SH)
XL	Qwen/Qwen2.5-Math-RM-72B		72B	69.06	79.89	55.92	57.09	57.44	55.30
XL	nvidia/AceMath-72B-RM		72B	72.83	84.76	61.73	53.29	59.24	56.21
XL	Qwen/WorldPM-72B-RLHFLow		72B	76.96	86.02	60.73	63.67	70.38	68.52
XL	ContextualAI/LMUnit-qwen2.5-72b		72B	27.89	39.10	20.17	6.92	11.81	14.53
XL	nvidia/Llama-3.3-Nemotron-70B-Reward		70B	78.39	88.76	61.30	66.44	65.78	73.26
XL	infly/INF-ORM-Llama3.1-70B		70B	74.84	82.88	62.03	62.63	68.18	66.60
XL	allenai/Llama-3.1-70B-Instruct-RM-RB2		70B	78.23	86.23	64.02	64.71	71.25	72.96
XL	allenai/Llama-3.1-Tulu-3-70B-SFT-RM-RB2		70B	78.96	86.23	65.16	67.13	73.58	73.76
XL	Nexusflow/Athene-RM-70B		70B	81.19	87.69	67.07	77.16	75.45	78.30
XL	Nexusflow/Starling-RM-34B		34B	70.63	77.75	54.55	63.67	64.58	68.72
XL	**Themis-RM 32B**		32B	91.82	94.27	84.95	95.16	87.59	94.55
XL	TIGER-Lab/AceCodeRM-32B		32B	62.95	73.67	57.91	40.48	46.36	49.55
XL	nvidia/Qwen3-Nemotron-32B-GenRM-Principle		32B	71.67	79.34	66.77	68.17	57.17	64.08
L	nicolinho/QRM-Gemma-2-27B		27B	52.48	53.89	52.71	47.06	50.57	48.85
L	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1		27B	75.57	84.28	65.47	62.63	67.31	63.17
L	Skywork/Skywork-Reward-Gemma-2-27B-v0.2		27B	54.16	56.74	56.74	49.13	50.77	51.36
L	internlm/internlm2-20b-reward		20B	73.13	81.38	60.20	65.74	64.11	66.26
L	**Themis-RM 14B**		14B	91.19	92.74	86.10	94.46	87.32	95.36
L	rubricreward/R3-Qwen3-14B-14k		14B	45.41	56.8	41.79	31.83	24.95	30.17
L	openbmb/UltraRM-13b		13B	70.43	74.17	55.23	66.44	71.18	72.45
M	**Themis-RM 8B**		8B	89.78	91.75	83.65	92.04	86.06	93.34
M	LARK-Lab/CodeScaler-8B		8B	79.12	87.42	61.27	72.32	73.45	73.26
M	rubricreward/R3-Qwen3-8B-14k		8B	43.23	52.91	39.88	29.07	27.22	29.36
M	Skywork/Skywork-Reward-V2-Qwen3-8B		8B	79.97	87.25	64.63	71.63	75.05	74.97
M	RLHFlow/ArmoRM-Llama3-8B-v0.1		8B	71.76	79.59	61.96	58.13	64.71	61.55
M	nicolinho/QRM-Llama3.1-8B-v2		8B	71.81	80.16	59.74	66.78	62.98	62.36
M	allenai/Llama-3.1-8B-Base-RM-RB2		8B	75.77	82.57	61.42	69.90	70.78	71.24
M	Ray2333/GRM-llama3-8B-sftreg		8B	71.81	79.05	58.75	57.44	66.11	67.00
M	LxzGordon/URM-LLaMa-3.1-8B		8B	71.62	79.91	60.73	65.05	62.31	62.16
M	NCSOFT/Llama-3-OffsetBias-RM-8B		8B	71.76	79.80	59.05	59.17	63.91	65.29
M	sfairXC/FsfairX-LLaMA3-RM-v0.1		8B	73.52	81.02	59.74	67.82	67.31	66.60
M	Nexusflow/Athene-RM-8B		8B	76.58	82.08	63.18	70.24	73.18	74.77
M	TIGER-Lab/AceCodeRM-7B		7B	71.11	82.48	55.54	60.55	59.44	57.62
M	eth-dl-rewards/internlm2-7b-reward-code-100k		7B	70.48	79.82	56.07	59.17	64.91	60.54
M	eth-dl-rewards/internlm2-7b-reward-math-100k		7B	70.30	77.75	56.15	57.79	66.18	62.97
M	reciprocate/mistral-7b-gsm8k-code-rm		7B	63.60	71.68	48.51	53.29	58.91	54.69
M	nvidia/AceMath-7B-RM		7B	67.41	77.40	59.28	55.63	53.24	55.50
M	openbmb/Eurus-RM-7b		7B	67.37	74.09	54.93	58.13	61.44	63.07
M	internlm/internlm2-7b-reward		7B	71.73	79.30	57.37	62.28	67.38	63.57
S	**Themis-RM 4B**		4B	88.39	89.72	83.65	92.39	84.52	92.94
S	LARK-Lab/CodeScaler-4B		4B	77.97	85.58	63.94	71.28	72.05	70.74
S	PKU-ONELab/CE-RM-4B		4B	57.16	66.64	52.79	39.10	44.16	42.18
S	rubricreward/R3-Qwen3-4B-14k		4B	41.10	50.52	41.10	21.80	23.95	27.25
S	Skywork/Skywork-Reward-V2-Qwen3-4B		4B	79.27	86.40	64.78	72.66	74.38	73.36
S	Ray2333/GRM-llama3.2-3B-sftreg		3B	71.28	78.30	57.07	62.98	65.98	66.70
XS	internlm/internlm2-1_8b-reward		1.8B	63.84	67.06	52.56	62.28	64.71	62.36
XS	**Themis-RM 1.7B**		1.7B	83.04	82.82	81.89	86.85	80.05	89.00
XS	LARK-Lab/CodeScaler-1.7B		1.7B	73.75	80.85	61.65	69.90	67.38	66.30
XS	Skywork/Skywork-Reward-V2-Qwen3-1.7B		1.7B	75.60	82.36	64.17	69.20	68.78	70.33
XXS	**Themis-RM 0.6B**		0.6B	79.26	77.00	79.98	86.85	78.79	87.69
XXS	Skywork/Skywork-Reward-V2-Qwen3-0.6B		0.6B	72.77	78.51	62.80	62.98	69.45	66.20
Ablations	**Themis-RM 8B w/o AuxLoss**		8B	87.23	89.43	81.98	88.96	82.55	90.21
Ablations	**Themis-RM 8B w/o AuxLoss And PT**		8B	85.14	86.43	81.94	88.79	80.58	88.98
Ablations	**Themis-RM 8B w/ All-Criteria Prompt (Inference-Time Only)**		8B	88.16	90.94	81.13	92.04	82.52	91.42
Ablations	**Themis-RM 8B w/o Criteria System Prompt**		8B	87.88	90.60	80.83	91.00	82.05	91.93
Ablations	**Themis-RM 8B w/ Criteria-Level Model-Merge**		8B	77.07	83.58	70.03	73.18	62.36	78.41
Ablations	**Themis-RM 8B w/ FC Prefs. Only**		8B	78.61	90.31	58.76	70.11	62.34	75.58
Ablations	**Themis-RM 8B w/ EE Prefs. Only**		8B	72.67	74.91	82.67	66.48	60.23	69.32
Ablations	**Themis-RM 8B w/ ME Prefs. Only**		8B	67.65	69.97	67.79	83.74	56.31	68.72
Ablations	**Themis-RM 8B w/ R&M Prefs. Only**		8B	72.04	76.56	60.19	70.61	71.11	67.71
Ablations	**Themis-RM 8B w/ SH Prefs. Only**		8B	73.35	78.67	60.09	73.12	58.77	87.32

Table 2: Detailed criteria-level preference accuracy of extant RMs and the `Themis-RM` suite on `Themis-CodeRewardBench` (**RQ1**: ref. to Section 5.1) followed by `Themis-RM-8B` training ablations (**RQ2**: ref. to Section 5.2). For detailed dataset-level scores per RM, consult Tables 7 to 9 in Appendix D.

substantially on code sourced from commit preferences. Similarly, a look at the scores on the `MBPP+Fix (Hard)` dataset (Prasad et al., 2025) shows that RMs largely cannot discern between varying degrees of partial correctness; only larger `Themis-RM` models and a handful of existing RMs can capture these subtle differences. Existing RM benchmarks (Lambert et al., 2025; Liu et al., 2025f) fail to uncover these weaknesses because they primarily rely on a small set of simple and dated datasets such as `HumanEvalPack` (Muennighoff et al., 2024), which, as our results show, are too saturated to serve as a useful indicator of RMs' scoring ability.

We further observe that generative (Mahan et al., 2024; Zhang et al., 2025e) and reasoning-enabled (Anugraha et al., 2025) RMs are unable to leverage inference-time compute effectively for pointwise reference-free code evaluation and often suffer due to the low resolution of text-based scoring. This effect is even more pronounced for generative RMs that self-propose evaluation rubrics (Hu et al., 2026; Saad-Falcon et al., 2025). Conversely, we find that scalar reward modeling is well-suited to reference-free code evaluation. Besides the clear dominance of the `Themis-RM` models, the most competitive existing RMs in our assessment are *all* scalar RMs (Liu et al., 2025a; Zhu et al., 2026). This, we believe, is in line with the findings that sequence-level representations are effective for both functional and non-functional code understanding (Lin et al., 2026; Ribeiro et al., 2025; Akhauri et al., 2025). Overall, these results show that training on diverse, multi-criteria code preferences is crucial for a strong code RM. Our results indicate an above-average performance from math RMs, which confirms prior findings of positive transfer (Cheng et al., 2025b; Yu et al., 2024b): this transfer, however, is limited only to judging functional correctness in algorithmic code.

5.2 RQ2: Training Choices To Minimize Multi-Criteria Interference In Code RMs

Multi-criteria RMs require minimization of cross-criteria interference. We thus perform a series of ablations on `Themis-RM-8B` to assess the effects of components of the `Themis-RM` training recipe and to investigate how the various code evaluation criteria interact. Our results, shown at the bottom of Table 2, indicate that both priming the RM with PT and including auxiliary training losses improve performance across the board. The PT phase enables RMs to learn general notions of human preference (Yu et al., 2024b) and mitigate value biases learned in LM pre-training (Christian et al., 2026). Training with auxiliary losses helps mitigate generator-validator inconsistencies (Li et al., 2024c), thereby improving scoring accuracy.

Given the consistently weak performance of existing RMs on non-functional preferences (Section 5.1), we next investigate the role of criteria prompting in disentangling multi-dimensional preferences. To this end, we train a multi-task baseline model *without* any criteria-defining prompts. We observe that, while learning multi-dimensional preferences via multi-task learning is viable, including explicit criteria prompts has a markedly positive effect: this is in line with results from rubric-enabled evaluators (Gunjal et al., 2025; Zhang et al., 2025c; Liu et al., 2025d). The main benefit of RM training with criterion prompts (see §4.2), however, emerges with criterion-specific prompting at inference time, as inference with a prompt that lists all criteria only marginally improves over the multi-task baseline (88.16 vs. 87.88%). We next test cross-criteria transfer, where we train using single-criteria preferences ({`FC`, `EE`, `ME`, `R&M`, `SH`} `Prefs. Only` in Table 2) and evaluate for all other criteria. The results show strong generalization of `Themis-RM`: even in zero-shot cross-criteria transfer, it outperforms most existing RMs (cf. the "M" section of Table 2). We find that training on FC preferences transfers better to non-functional requirements than vice versa, owing to non-functional code improvements frequently coinciding with functional code changes. None of the criteria-specific models match the full `Themis-RM`, not even for their respective criterion: this suggests positive transfer between the quality criteria. Finally, we find that merging (Yadav et al., 2023) of criterion-specific models is surprisingly ineffective, with the merged RM underperforming the multi-task baseline by over 10 points. This finding is likely due to the under-specified nature of the Bradley-Terry training objective, which can lead to individual RMs with very different reward scales (and our weak regularization is unable to mitigate this effect fully).[5]

5.3 RQ3: Cross-Lingual Transfer In Code RMs

We next examine the multilingual and cross-lingual characteristics of code RMs. Multilingual interference is a well-documented phenomenon (Wang et al., 2020; Conneau et al., 2020; Arivazhagan et al., 2019; Lauscher et al., 2020), but prior work on RMs has largely focused on the natural language, with mixed findings ranging from strong interference (Lai et al., 2023; Gureja et al., 2025) to positive transfer (Wu et al., 2024; Hong et al., 2025; Dang et al., 2024). The question is arguably even more pertinent for code LMs, which often exhibit large performance disparities even across high-resource programming languages (Athiwaratkun et al., 2023; Paul et al., 2024; Rashid et al., 2025). We first compare per-programming-language accuracy of `Themis-RM` with that of existing RMs on `Themis-CodeRewardBench` in Table 3. Barring the rare poor performance of some RMs in a few specific languages (attributable to respective training distributions), we find that most RMs exhibit modest performance differences between high- and mid-resource programming

[5]We thus experimented with higher values of $\mu \in \{0.1, 0.5\}$, but that overly constrained reward variance and led to lower accuracy, particularly harming the RMs' performance in listwise re-ranking (Chen et al., 2024b; Razin et al., 2025).

Auxiliary Training Objectives | Code RM | Criteria-Following RM | Math RM | Generative RM | Reasoning RM

	Model	Size	Average	Programming-Language-Level Accuracy							
				C	C#	C++	Go	Java	JS	Python	Ruby
XL	Qwen/Qwen2.5-Math-RM-72B	72B	69.06	74.94	60.11	70.7	75.62	71.23	66.51	65.94	68.54
XL	nvidia/AceMath-72B-RM	72B	72.83	77.45	61.17	77.7	78.93	76.21	68.17	69.33	70.27
XL	Qwen/WorldPM-72B-RLHFLow	72B	76.96	79.73	73.40	78.72	85.12	81.12	75.00	72.37	75.57
XL	ContextualAI/LMUnit-qwen2.5-72b	72B	27.89	15.95	7.98	31.41	21.76	34.70	23.89	32.46	18.05
XL	nvidia/Llama-3.3-Nemotron-70B-Reward	70B	78.39	82.69	70.21	79.61	84.99	83.57	77.12	73.23	78.49
XL	infly/INF-ORM-Llama3.1-70B	70B	74.84	76.77	67.55	74.85	79.34	78.73	73.52	72.76	72.97
XL	allenai/Llama-3.1-70B-Instruct-RM-RB2	70B	78.23	80.64	75.00	78.25	81.13	81.80	77.95	75.68	77.19
XL	allenai/Llama-3.1-Tulu-3-70B-SFT-RM-RB2	70B	78.96	79.04	75.00	77.91	83.47	83.30	79.61	75.92	78.70
XL	Nexusflow/Athene-RM-70B	70B	81.19	81.78	76.60	78.86	86.36	85.48	83.30	78.37	80.00
XL	Nexusflow/Starling-RM-34B	34B	70.63	75.17	61.70	69.20	73.55	73.89	71.40	67.07	74.05
XL	**Themis-RM 32B**	32B	91.82	92.94	94.15	89.67	93.94	95.16	91.42	90.34	91.89
XL	TIGER-Lab/AceCodeRM-32B	32B	62.95	58.54	47.87	68.05	64.88	66.05	57.56	61.46	64.00
XL	nvidia/Qwen3-Nemotron-32B-GenRM-Principle	32B	71.67	71.07	60.11	75.80	74.79	75.19	66.79	71.43	66.05
L	nicolinho/QRM-Gemma-2-27B	27B	52.48	52.16	46.81	52.48	51.10	54.87	51.38	53.59	49.30
L	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1	27B	75.57	76.99	69.15	77.57	80.58	79.35	72.51	72.53	75.14
L	Skywork/Skywork-Reward-Gemma-2-27B-v0.2	27B	54.16	57.63	50.53	54.49	56.34	53.10	52.03	54.72	53.51
L	internlm/internlm2-20b-reward	20B	73.13	77.45	70.05	74.29	77.10	78.94	72.11	69.29	69.41
L	**Themis-RM 14B**	14B	91.19	94.53	92.02	89.60	92.56	93.73	91.05	89.63	91.35
L	rubricreward/R3-Qwen3-14B-14k	14B	45.41	47.84	29.26	55.61	49.45	49.69	38.93	43.76	33.51
L	openbmb/UltraRM-13b	13B	70.43	68.79	74.47	68.12	72.04	74.78	74.45	66.95	70.81
M	**Themis-RM 8B**	8B	89.78	90.21	90.43	88.24	91.60	92.43	89.94	88.82	88.76
M	LARK-Lab/CodeScaler-8B	8B	79.12	84.74	78.19	76.55	84.85	83.98	78.14	76.46	77.08
M	rubricreward/R3-Qwen3-8B-14k	8B	43.23	38.72	29.79	46.91	46.83	48.19	38.84	43.92	34.81
M	Skywork/Skywork-Reward-V2-Qwen3-8B	8B	79.97	84.74	79.79	77.50	81.62	84.46	79.89	77.71	79.46
M	RLHFlow/ArmoRM-Llama3-8B-v0.1	8B	71.76	73.58	62.23	76.00	72.31	74.64	70.20	69.06	70.38
M	nicolinho/QRM-Llama3.1-8B-v2	8B	71.81	75.40	68.09	71.31	72.87	76.07	71.49	70.30	68.65
M	allenai/Llama-3.1-8B-Base-RM-RB2	8B	75.77	78.82	72.34	74.17	79.06	78.94	77.40	78.31	73.51
M	Ray2333/GRM-llama3-8B-sftreg	8B	71.81	71.30	68.09	73.96	72.59	75.94	71.86	69.33	69.08
M	LxzGordon/URM-LLaMa-3.1-8B	8B	71.62	76.08	73.62	65.96	73.00	74.98	71.96	68.32	69.84
M	NCSOFT/Llama-3-OffsetBias-RM-8B	8B	71.76	71.75	63.83	74.10	76.03	75.53	73.25	67.81	69.51
M	sfairXC/FsfairX-LLaMA3-RM-v0.1	8B	73.52	77.45	71.28	74.44	75.90	77.03	75.18	69.60	72.11
M	Nexusflow/Athene-RM-8B	8B	76.58	77.90	77.66	74.03	79.06	80.44	80.26	74.12	74.27
M	TIGER-Lab/AceCodeRM-7B	7B	71.11	76.77	62.77	72.47	75.90	77.78	68.54	67.23	67.46
M	eth-dl-rewards/internlm2-7b-reward-code-100k	7B	70.48	70.84	67.02	72.60	76.45	75.12	72.42	66.45	64.54
M	eth-dl-rewards/internlm2-7b-reward-math-100k	7B	70.30	70.62	69.68	70.84	77.00	74.71	73.71	65.28	67.14
M	reciprocate/mistral-7b-gsm8k-code-rm	7B	63.60	63.78	63.30	65.87	65.15	67.83	63.84	59.31	63.68
M	nvidia/AceMath-7B-RM	7B	67.41	68.56	53.72	74.58	71.49	70.07	62.64	65.98	60.43
M	openbmb/Eurus-RM-7b	7B	67.37	67.88	68.09	68.52	69.56	72.67	65.68	63.37	68.11
M	internlm/internlm2-7b-reward	7B	71.73	74.03	68.62	72.13	78.37	75.39	74.72	66.91	69.51
S	**Themis-RM 4B**	4B	88.39	86.79	88.83	88.17	90.36	90.66	90.50	86.94	85.84
S	LARK-Lab/CodeScaler-4B	4B	77.97	81.78	71.28	77.63	83.20	82.75	78.97	74.36	74.27
S	PKU-ONELab/CE-RM-4B	4B	57.16	57.63	43.09	64.58	58.40	61.76	53.84	55.53	49.08
S	rubricreward/R3-Qwen3-4B-14k	4B	41.10	36.67	22.87	48.33	42.56	46.08	32.93	40.84	36.65
S	Skywork/Skywork-Reward-V2-Qwen3-4B	4B	79.27	81.09	81.91	78.72	84.85	83.71	79.70	76.23	75.24
S	Ray2333/GRM-llama3.2-3B-sftreg	3B	71.28	74.72	60.11	71.72	73.55	75.26	71.68	69.21	68.43
XS	internlm/internlm2-1_8b-reward	1.8B	63.84	64.24	59.57	63.77	69.70	64.89	66.70	61.57	61.30
XS	**Themis-RM 1.7B**	1.7B	83.04	80.64	82.45	83.82	84.99	84.66	83.49	81.80	81.84
XS	LARK-Lab/CodeScaler-1.7B	1.7B	73.75	76.31	75.00	73.01	81.27	77.30	73.34	71.47	68.76
XS	Skywork/Skywork-Reward-V2-Qwen3-1.7B	1.7B	75.60	76.54	77.13	74.92	80.30	78.19	75.83	73.85	72.76
XXS	**Themis-RM 0.6B**	0.6B	79.26	79.73	77.63	78.31	75.90	82.07	80.07	79.35	77.84
XXS	Skywork/Skywork-Reward-V2-Qwen3-0.6B	0.6B	72.77	74.26	72.87	72.26	73.97	76.01	73.52	71.63	68.08
Ablation	**Themis-RM 8B w/ 50k All Language Prefs. Only**	8B	85.31	84.63	82.91	83.57	87.95	87.11	83.58	85.24	86.19
Ablation	**Themis-RM 8B w/ 50k High Resource Prefs. Only**	8B	84.87	82.46	81.04	82.25	83.84	88.42	85.79	85.29	83.94
Ablation	**Themis-RM 8B w/ 50k Python Prefs. Only**	8B	78.62	72.09	69.52	69.75	72.56	73.51	83.44	86.01	84.43
Ablation	**Themis-RM 8B w/ 50k Java Prefs. Only**	8B	81.42	80.32	78.51	79.55	75.36	87.91	82.11	81.40	79.31

Table 3: Detailed programming-language-level preference accuracy of extant RMs and the `Themis-RM` suite on `Themis-CodeRewardBench` (**RQ3**: ref. to Section 5.3) followed by `Themis-RM-8B` cross-lingual transfer ablations (**RQ3**: ref. to Section 5.3). For detailed dataset-level scores per RM, consult Tables 7 to 9 in Appendix D.

languages. We next perform controlled cross-lingual transfer experiments with `Themis-RM-8B` (see bottom of Table 3), mirroring the setup presented in Dang et al. (2024). We run size-controlled comparisons between all-language, high-resource-language, and Python- and Java-only training runs. We observe the following transfer patterns. Python RMs transfer better to other dynamically typed languages, whereas Java RMs transfer better to statically typed ones. Training on all languages yields the best performance, suggesting net-positive cross-lingual transfer effects. Interestingly, all four models show small performance differences across languages, suggesting that training code RMs on diverse, multi-criteria preferences yields stable multilingual reward modeling. We believe this is due to (1) the extensive oversampling of multilingual code data in the training of modern LMs Chen et al. (2026); Liu et al. (2025c) and (2) situational coverage in high-resource languages being more beneficial than data volume in low-resource ones (Son et al., 2024).

5.4 RQ4: Downstream Robustness Evaluation

The most prevalent downstream applications of RMs, such as post-training (DeepSeek-AI, 2025a; Dong et al., 2023a; Singh et al., 2024; Peng et al., 2019) and score-guided inference-time search (Wang et al., 2024e;

Auxiliary Training Objectives | Code RM | Criteria-Following RM | Math RM | Generative RM | Reasoning RM

	Model	Size	Re-Ranking						Adversarial		
			Hits@10 (Mean)			Rank Corr.@40 (Median)			Accuracy		
			C++	Java	Python	C++	Java	Python	C++	Java	Python
XL	Qwen/WorldPM-72B-RLHFLow	72B	43.12	40.14	45.76	0.0591	0.0656	0.1644	57.88	56.45	59.88
	nvidia/Llama-3.3-Nemotron-70B-Reward	70B	71.91	73.14	78.59	0.2568	0.2644	0.3012	63.48	62.12	55.91
	Nexusflow/Athene-RM-70B	70B	24.88	27.78	29.34	-0.0567	-0.0245	0.1281	53.19	52.44	54.47
	Themis-RM 32B	32B	97.65	98.59	97.44	0.5067	0.5352	0.5018	81.43	82.09	83.02
	TIGER-Lab/AceCodeRM-32B	32B	21.76	19.23	22.34	0.0256	0.067	0.3415	62.41	65.33	64.76
L	**Themis-RM 14B**	14B	95.88	95.91	96.84	0.4649	0.4711	0.4619	76.48	77.56	79.29
M	**Themis-RM 8B**	8B	95.29	94.75	96.26	0.4448	0.4404	0.4387	72.04	73.48	74.54
	LARK-Lab/CodeScaler-8B	8B	94.65	94.87	96.43	0.4431	0.4486	0.4403	73.28	71.7	75.43
	Skywork/Skywork-Reward-V2-Qwen3-8B	8B	91.45	90.45	91.41	0.3765	0.3734	0.3922	56.81	56.78	57.65
	Nexusflow/Athene-RM-8B	8B	33.97	34.82	34.98	0.3492	0.3522	0.3411	50.89	44.5	47.04
	TIGER-Lab/AceCodeRM-7B	7B	34.21	32.09	35.46	-0.0426	0.0398	-0.0131	62.21	63.14	63.02
S	**Themis-RM 4B**	4B	93.17	93.45	95.1	0.4148	0.4192	0.4197	68.73	71.18	69.45
	LARK-Lab/CodeScaler-4B	4B	93.33	93.11	93.78	0.4255	0.4214	0.4216	71.33	71.06	74.19
	Skywork/Skywork-Reward-V2-Qwen3-4B	4B	89.45	88.32	90.19	0.3677	0.3594	0.3687	55.96	54.76	57.16
XS	**Themis-RM 1.7B**	1.7B	89.45	88.98	90.77	0.3978	0.4067	0.4092	61.39	56.92	64.26
	LARK-Lab/CodeScaler-1.7B	1.7B	90.86	89.85	91.72	0.4046	0.4095	0.4113	63.06	60.54	64.97
	Skywork/Skywork-Reward-V2-Qwen3-1.7B	1.7B	84.52	82.89	84.54	0.2219	0.2106	0.2384	51.12	49.69	52.69
XXS	**Themis-RM 0.6B**	0.6B	84.65	83.98	86.76	0.1769	0.1878	0.2171	53.11	50.16	53.14
	Skywork/Skywork-Reward-V2-Qwen3-0.6B	0.6B	79.53	78.55	79.87	0.0971	0.1042	0.1065	47.07	42.69	44.26

Table 4: Re-ranking and adversarial robustness evaluation of the `Themis-RM` suite (**RQ4**: ref. to Section 5.4). We benchmark against a selection of the highest scoring extant RMs on `Themis-CodeRewardBench`.

Zuo et al., 2025) demand accurate listwise re-ranking of LM responses. Lists of model-generated responses can typically contain model-specific artifacts (Dou et al., 2025; Orel et al., 2025) and attempts at verifier-hacking (Moon et al., 2026), to which the RMs with evaluation criteria prompts are more susceptible (Li et al., 2025c). To verify the downstream effectiveness of `Themis-RM`, we directly evaluate its listwise re-ranking performance and adversarial preference accuracy on code.

For listwise re-ranking, we source `40` solutions each in `C++`, `Java`, and `Python` to problems in `CodeContests+` (Wang et al., 2025g) from a mix of open-source LMs (Team, 2025; 2024; Rozière et al., 2023; Guo et al., 2024a). We then execute these solutions against the predefined test-cases using a code sandbox[6], and observe how well RMs can retrieve completely correct solutions (`Hits@10`) and how well their listwise scores correlate with the ground-truth (`Rank Corr.@40`). We then evaluate robustness to adversarial perturbations using accuracy on the `Aletheia-Adv` (Venkatkrishna et al., 2026) suite, which tests RM accuracy on correct-incorrect code pairs modified using a variety of judge-hacking behaviors (Wang et al., 2025d; Lam et al., 2025). Overall, the results (in Table 4) demonstrate the competitiveness of `Themis-RM` models in realistic downstream scenarios. Our models (i) achieve state-of-the-art adversarial robustness and (ii) match the performance of the best RMs specialized for correctness of contest-type code in re-ranking, a task proven to predict downstream post-training utility (Feng et al., 2025a; Wen et al., 2025; Kim et al., 2025). Most encouragingly, and in contrast to prior work (Bartoldson et al., 2024), we find that `Themis-RM` exhibits better scaling trends in downstream robustness than in our pairwise accuracy tests.

6 Conclusion

We present the `Themis` project, an investigation into the viability of model-based scoring of LM-generated code for multi-criteria alignment, quality estimation, and re-ranking. We first construct `Themis-CodeRewardBench`, an exhaustive preference evaluation suite that tests RMs' scoring competence across eight typologically diverse programming languages and their ability to accurately score code preferences across different functional and non-functional criteria. Subsequently, we outline the creation of `Themis-GeneralPreference` and `Themis-CodePreference`, two diverse multi-dimensional preference datasets that we use to train `Themis-RM`, a suite of state-of-the-art multilingual multi-criteria code RMs. Our opening evaluations demonstrate that current RMs struggle to model code preferences beyond a very narrow mode of operation, namely, judging the functional correctness of code contest solutions. In subsequent experiments, we illustrate that code RMs trained on a variety of code preferences and evaluation criteria exhibit clear-cut positive scaling trends with minimal cross-criteria interference, non-trivial cross-lingual transfer, and robust performance in adversarial settings, establishing them as a viable reward source for code post-training. We hope that our findings, coupled with open-source artifacts, will expand the use of RMs in post-training beyond executable code.

[6] `bytedance/sandboxfusion`

References

Julien Abadji, Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. Towards a cleaner document-oriented multilingual crawled corpus. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Thirteenth Language Resources and Evaluation Conference, LREC 2022, Marseille, France, 20-25 June 2022*, pp. 4344–4355. European Language Resources Association, 2022. URL https://aclanthology.org/2022.lrec-1.463.

Pooja Aggarwal, Oishik Chatterjee, Ting Dai, Prateeti Mohapatra, Brent Paulovicks, Brad Blancett, and Arthur De Magalhaes. Codesift: An llm-based reference-less framework for automatic code validation. In Rong N. Chang, Carl K. Chang, Jingwei Yang, Nimanthi L. Atukorala, Zhi Jin, Michael Sheng, Jing Fan, Kenneth Fletcher, Qiang He, Tevfik Kosar, Santonu Sarkar, Sreekrishnan Venkateswaran, Shangguang Wang, Xuanzhe Liu, Seetharami Seelam, Chandra Narayanaswami, and Ziliang Zong (eds.), *17th IEEE International Conference on Cloud Computing, CLOUD 2024, Shenzhen, China, July 7-13, 2024*, pp. 404–410. IEEE, 2024. doi: 10.1109/CLOUD62652.2024.00052. URL https://doi.org/10.1109/CLOUD62652.2024.00052.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.662. URL https://doi.org/10.18653/v1/2024.acl-long.662.

Yash Akhauri, Xingyou Song, Arissa Wongpanich, Bryan Lewandowski, and Mohamed S. Abdelfattah. Regression language models for code. *CoRR*, abs/2509.26476, 2025. doi: 10.48550/ARXIV.2509.26476. URL https://doi.org/10.48550/arXiv.2509.26476.

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D. Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models. *CoRR*, abs/2408.11791, 2024. doi: 10.48550/ARXIV.2408.11791. URL https://doi.org/10.48550/arXiv.2408.11791.

David Anugraha, Zilu Tang, Lester James V. Miranda, Hanyang Zhao, Mohammad Rifqi Farhansyah, Garry Kuwanto, Derry Wijaya, and Genta Indra Winata. R3: robust rubric-agnostic reward models. *CoRR*, abs/2505.13388, 2025. doi: 10.48550/ARXIV.2505.13388. URL https://doi.org/10.48550/arXiv.2505.13388.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George F. Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. Massively multilingual neural machine translation in the wild: Findings and challenges. *CoRR*, abs/1907.05019, 2019. URL http://arxiv.org/abs/1907.05019.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment. *CoRR*, abs/2112.00861, 2021. URL https://arxiv.org/abs/2112.00861.

Ben Athiwaratkun, Sanjay Krishna Gouda, Zijian Wang, Xiaopeng Li, Yuchen Tian, Ming Tan, Wasi Uddin Ahmad, Shiqi Wang, Qing Sun, Mingyue Shang, Sujan Kumar Gonugondla, Hantian Ding, Varun Kumar, Nathan Fulton, Arash Farahani, Siddhartha Jain, Robert Giaquinto, Haifeng Qian, Murali Krishna Ramanathan, and Ramesh Nallapati. Multi-lingual evaluation of code generation models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=Bo7eeXm6An8.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr,

Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073, 2022. doi: 10.48550/ARXIV.2212.08073. URL https://doi.org/10.48550/arXiv.2212.08073.

Brian R. Bartoldson, James Diffenderfer, Konstantinos Parasyris, and Bhavya Kailkhura. Adversarial robustness limits via scaling-law and human-alignment studies. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 3046–3072. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/bartoldson24a.html.

Paul M. Bays and Ben A. Dowding. Fidelity of the representation of value in decision-making. *PLoS Comput. Biol.*, 13(3), 2017. doi: 10.1371/JOURNAL.PCBI.1005405. URL https://doi.org/10.1371/journal.pcbi.1005405.

Scott Blyth, Sherlock A. Licorish, Christoph Treude, and Markus Wagner. Static analysis as a feedback loop: Enhancing llm-generated code beyond correctness. *CoRR*, abs/2508.14419, 2025. doi: 10.48550/ARXIV.2508.14419. URL https://doi.org/10.48550/arXiv.2508.14419.

Andrei Z. Broder. On the resemblance and containment of documents. In Bruno Carpentieri, Alfredo De Santis, Ugo Vaccaro, and James A. Storer (eds.), *Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings*, pp. 21–29. IEEE, 1997. doi: 10.1109/SEQUEN.1997.666900. URL https://doi.org/10.1109/SEQUEN.1997.666900.

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. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Elena Bruches, Daniil Grebenkin, Mikhail Klementev, Vadim Alperovich, Roman Derunets, Dari Baturova, Georgy Mkrtchyan, Oleg Sedukhin, Ivan Bondarenko, Nikolay Bushkov, and Stanislav Moiseev. RM -rf: Reward model for run-free unit test evaluation. *CoRR*, abs/2601.13097, 2026. doi: 10.48550/ARXIV.2601.13097. URL https://doi.org/10.48550/arXiv.2601.13097.

Quang-Cuong Bui, Riccardo Scandariato, and Nicolás E. Díaz Ferreyra. Vul4j: A dataset of reproducible java vulnerabilities geared towards the study of program repair techniques. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*, pp. 464–468. ACM, 2022. doi: 10.1145/3524842.3528482. URL https://doi.org/10.1145/3524842.3528482.

Alexander Bukharin, Haifeng Qian, Shengyang Sun, Adithya Renduchintala, Soumye Singhal, Zhilin Wang, Oleksii Kuchaiev, Olivier Delalleau, and Tuo Zhao. Adversarial training of reward models. *CoRR*, abs/2504.06141, 2025. doi: 10.48550/ARXIV.2504.06141. URL https://doi.org/10.48550/arXiv.2504.06141.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu,

Yingtong Xiong, Xiaomeng Zhao, and et al. Internlm2 technical report. *CoRR*, abs/2403.17297, 2024. doi: 10.48550/ARXIV.2403.17297. URL https://doi.org/10.48550/arXiv.2403.17297.

Nitay Calderon, Liat Ein-Dor, and Roi Reichart. Multi-domain explainability of preferences. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 14542–14575. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.736. URL https://doi.org/10.18653/v1/2025.emnlp-main.736.

Yuhan Cao, Zian Chen, Kun Quan, Ziliang Zhang, Yu Wang, Xiaoning Dong, Yeqi Feng, Guanzhong He, Jingcheng Huang, Jianhao Li, Yixuan Tan, Jiafu Tang, Yilin Tang, Junlei Wu, Qianyu Xiao, Can Zheng, Shouchen Zhou, Yuxiang Zhu, Yiming Huang, Tian Xie, and Tianxing He. Can llms generate reliable test case generators? A study on competition-level programming problems. *CoRR*, abs/2506.06821, 2025. doi: 10.48550/ARXIV.2506.06821. URL https://doi.org/10.48550/arXiv.2506.06821.

Linzheng Chai, Shukai Liu, Jian Yang, Yuwei Yin, Ke Jin, Jiaheng Liu, Tao Sun, Ge Zhang, Changyu Ren, Hongcheng Guo, Noah Wang, Boyang Wang, Xianjie Wu, Bing Wang, Tongliang Li, Liqun Yang, Sufeng Duan, Zhaoxiang Zhang, and Zhoujun Li. Mceval: Massively multilingual code evaluation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=UunCPtPOlZ.

Souradip Chakraborty, Jiahao Qiu, Hui Yuan, Alec Koppel, Dinesh Manocha, Furong Huang, Amrit S. Bedi, and Mengdi Wang. Maxmin-rlhf: Alignment with diverse human preferences. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6116–6135. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/chakraborty24b.html.

Angelica Chen, Jérémy Scheurer, Tomasz Korbak, Jon Ander Campos, Jun Shern Chan, Samuel R. Bowman, Kyunghyun Cho, and Ethan Perez. Improving code generation by training with natural language feedback. *CoRR*, abs/2303.16749, 2023a. doi: 10.48550/ARXIV.2303.16749. URL https://doi.org/10.48550/arXiv.2303.16749.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/forum?id=ktrw68Cmu9c.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Mayee F. Chen, Tyler Murray, David Heineman, Matt Jordan, Hannaneh Hajishirzi, Christopher Ré, Luca Soldaini, and Kyle Lo. Olmix: A framework for data mixing throughout LM development. *CoRR*, abs/2602.12237, 2026. doi: 10.48550/ARXIV.2602.12237. URL https://doi.org/10.48550/arXiv.2602.12237.

Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. Judgelrm: Large reasoning models as a judge. *CoRR*, abs/2504.00050, 2025a. doi: 10.48550/ARXIV.2504.00050. URL https://doi.org/10.48550/arXiv.2504.00050.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=KuPixIqPiq.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. RM-R1: reward modeling as reasoning. *CoRR*, abs/2505.02387, 2025b. doi: 10.48550/ARXIV.2505.02387. URL https://doi.org/10.48550/arXiv.2505.02387.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *CoRR*, abs/2505.16400, 2025c. doi: 10.48550/ARXIV.2505.16400. URL https://doi.org/10.48550/arXiv.2505.16400.

Yanjun Chen, Dawei Zhu, Yirong Sun, Xinghao Chen, Wei Zhang, and Xiaoyu Shen. The accuracy paradox in RLHF: when better reward models don't yield better language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 2980–2989. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.EMNLP-MAIN.174. URL https://doi.org/10.18653/v1/2024.emnlp-main.174.

Zhoujun Cheng, Richard Fan, Shibo Hao, Taylor W. Killian, Haonan Li, Suqi Sun, Hector Ren, Alexander Moreno, Daqian Zhang, Tianjun Zhong, Yuxin Xiong, Yuanzhe Hu, Yutao Xie, Xudong Han, Yuqi Wang, Varad Pimpalkhute, Yonghao Zhuang, Aaryamonvikram Singh, Xuezhi Liang, Anze Xie, Jianshu She, Desai Fan, Chengqian Gao, Liqun Ma, Mikhail Yurochkin, John Maggs, Xuezhe Ma, Guowei He, Zhiting Hu, Zhengzhong Liu, and Eric P. Xing. K2-think: A parameter-efficient reasoning system. *CoRR*, abs/2509.07604, 2025a. doi: 10.48550/ARXIV.2509.07604. URL https://doi.org/10.48550/arXiv.2509.07604.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, Taylor W. Killian, Mikhail Yurochkin, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. Revisiting reinforcement learning for LLM reasoning from A cross-domain perspective. *CoRR*, abs/2506.14965, 2025b. doi: 10.48550/ARXIV.2506.14965. URL https://doi.org/10.48550/arXiv.2506.14965.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 8359–8388. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/chiang24b.html.

Brian Christian, Hannah Rose Kirk, Jessica A. F. Thompson, Christopher Summerfield, and Tsvetomira Dumbalska. Reward model interpretability via optimal and pessimal tokens. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency, FAccT 2025, Athens, Greece, June 23-26, 2025*, pp. 1048–1059. ACM, 2025. doi: 10.1145/3715275.3732068. URL https://doi.org/10.1145/3715275.3732068.

Brian Christian, Jessica A. F. Thompson, Elle Michelle Yang, Vincent Adam, Hannah Rose Kirk, Christopher Summerfield, and Tsvetomira Dumbalska. Reward models inherit value biases from pretraining. *CoRR*, abs/2601.20838, 2026. doi: 10.48550/ARXIV.2601.20838. URL https://doi.org/10.48550/arXiv.2601.20838.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/chu25c.html.

Tristan Coignion, Clément Quinton, and Romain Rouvoy. When faster isn't greener: The hidden costs of llm-based code optimization. In *40th IEEE/ACM International Conference on Automated Software Engineering, ASE 2025, Seoul, Korea, Republic of, November 16-20, 2025*, pp. 1655–1666. IEEE, 2025. doi: 10.1109/ASE63991.2025.00139. URL https://doi.org/10.1109/ASE63991.2025.00139.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8440–8451. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.747. URL https://doi.org/10.18653/v1/2020.acl-main.747.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=dcjtMYkpXx.

Han Cui, Menglei Xie, Ting Su, Chengyu Zhang, and Shin Hwei Tan. An empirical study of false negatives and positives of static code analyzers from the perspective of historical issues. *CoRR*, abs/2408.13855, 2024. doi: 10.48550/ARXIV.2408.13855. URL https://doi.org/10.48550/arXiv.2408.13855.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe RLHF: safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=TyFrPOKYXw.

John Dang, Arash Ahmadian, Kelly Marchisio, Julia Kreutzer, Ahmet Üstün, and Sara Hooker. RLHF can speak many languages: Unlocking multilingual preference optimization for llms. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 13134–13156. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.729. URL https://doi.org/10.18653/v1/2024.emnlp-main.729.

Simone Daniotti, Johannes Wachs, Xiangnan Feng, and Frank Neffke. Who is using AI to code? global diffusion and impact of generative AI. *CoRR*, abs/2506.08945, 2025. doi: 10.48550/ARXIV.2506.08945. URL https://doi.org/10.48550/arXiv.2506.08945.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025a. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *CoRR*, abs/2512.02556, 2025b. doi: 10.48550/ARXIV.2512.02556. URL https://doi.org/10.48550/arXiv.2512.02556.

Yangruibo Ding, Yanjun Fu, Omniyyah Ibrahim, Chawin Sitawarin, Xinyun Chen, Basel Alomair, David A. Wagner, Baishakhi Ray, and Yizheng Chen. Vulnerability detection with code language models: How far are we? In *47th IEEE/ACM International Conference on Software Engineering, ICSE 2025, Ottawa, ON, Canada, April 26 - May 6, 2025*, pp. 1729–1741. IEEE, 2025. doi: 10.1109/ICSE55347.2025.00038. URL https://doi.org/10.1109/ICSE55347.2025.00038.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: reward ranked finetuning for generative foundation model alignment. *Trans. Mach. Learn. Res.*, 2023, 2023a. URL https://openreview.net/forum?id=m7p5O7zblY.

Yi Dong, Zhilin Wang, Makesh Narsimhan Sreedhar, Xianchao Wu, and Oleksii Kuchaiev. Steerlm: Attribute conditioned SFT as an (user-steerable) alternative to RLHF. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, volume EMNLP 2023 of *Findings of ACL*, pp. 11275–11288. Association for Computational Linguistics, 2023b. doi: 10.18653/V1/2023.FINDINGS-EMNLP.754. URL https://doi.org/10.18653/v1/2023.findings-emnlp.754.

Yihong Dong, Jiazheng Ding, Xue Jiang, Ge Li, Zhuo Li, and Zhi Jin. Codescore: Evaluating code generation by learning code execution. *ACM Trans. Softw. Eng. Methodol.*, 34(3):77:1–77:22, 2025. doi: 10.1145/3695991. URL https://doi.org/10.1145/3695991.

Karel D'Oosterlinck, Winnie Xu, Chris Develder, Thomas Demeester, Amanpreet Singh, Christopher Potts, Douwe Kiela, and Shikib Mehri. Anchored preference optimization and contrastive revisions: Addressing underspecification in alignment. *Trans. Assoc. Comput. Linguistics*, 13:442–460, 2025. doi: 10.1162/TACL_A_00748. URL https://doi.org/10.1162/tacl_a_00748.

Nicolai Dorka. Quantile regression for distributional reward models in RLHF. *CoRR*, abs/2409.10164, 2024. doi: 10.48550/ARXIV.2409.10164. URL https://doi.org/10.48550/arXiv.2409.10164.

Shihan Dou, Shichun Liu, Yuming Yang, Yicheng Zou, Yunhua Zhou, Shuhao Xing, Chenhao Huang, Qiming Ge, Demin Song, Haijun Lv, Songyang Gao, Chengqi Lv, Enyu Zhou, Honglin Guo, Zhiheng Xi, Wenwei Zhang, Qipeng Guo, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Tao Gui, and Kai Chen. Pre-trained policy discriminators are general reward models. *CoRR*, abs/2507.05197, 2025. doi: 10.48550/ARXIV.2507.05197. URL https://doi.org/10.48550/arXiv.2507.05197.

Mingzhe Du, Luu Tuan Tuan, Yue Liu, Yuhao Qing, Dong Huang, Xinyi He, Qian Liu, Zejun Ma, and See-Kiong Ng. Afterburner: Reinforcement learning facilitates self-improving code efficiency optimization. *CoRR*, abs/2505.23387, 2025. doi: 10.48550/ARXIV.2505.23387. URL https://doi.org/10.48550/arXiv.2505.23387.

Shukai Duan, Nikos Kanakaris, Xiongye Xiao, Heng Ping, Chenyu Zhou, Nesreen K. Ahmed, Guixiang Ma, Mihai Capotă, Theodore L. Willke, Shahin Nazarian, and Paul Bogdan. Perfrl: A small language model framework for efficient code optimization. 2023. URL https://api.semanticscholar.org/CorpusID:266163427.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, Dj Dvijotham, Adam Fisch, Katherine A. Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *CoRR*, abs/2312.09244, 2023. doi: 10.48550/ARXIV.2312.09244. URL https://doi.org/10.48550/arXiv.2312.09244.

Yanai Elazar, Akshita Bhagia, Ian Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hannaneh Hajishirzi, Noah A. Smith, and Jesse Dodge. What's in my big data? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=RvfPnOkPV4.

Tom Everitt, Victoria Krakovna, Laurent Orseau, and Shane Legg. Reinforcement learning with a corrupted reward channel. In Carles Sierra (ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 4705–4713. ijcai.org, 2017. doi: 10.24963/IJCAI.2017/656. URL https://doi.org/10.24963/ijcai.2017/656.

Zhiyu Fan, Kirill Vasilevski, Dayi Lin, Boyuan Chen, Yihao Chen, Zhiqing Zhong, Jie M. Zhang, Pinjia He, and Ahmed E. Hassan. Swe-effi: Re-evaluating software AI agent system effectiveness under resource constraints. *CoRR*, abs/2509.09853, 2025. doi: 10.48550/ARXIV.2509.09853. URL https://doi.org/10.48550/arXiv.2509.09853.

Yuanning Feng, Sinan Wang, Zhengxiang Cheng, Yao Wan, and Dongping Chen. Are we on the right way to assessing llm-as-a-judge? *CoRR*, abs/2512.16041, 2025a. doi: 10.48550/ARXIV.2512.16041. URL https://doi.org/10.48550/arXiv.2512.16041.

Yunlong Feng, Yang Xu, Xiao Xu, Binyuan Hui, and Junyang Lin. Towards better correctness and efficiency in code generation. *CoRR*, abs/2508.20124, 2025b. doi: 10.48550/ARXIV.2508.20124. URL https://doi.org/10.48550/arXiv.2508.20124.

Yunzhen Feng, Parag Jain, Anthony Hartshorn, Yaqi Duan, and Julia Kempe. Don't waste mistakes: Leveraging negative rl-groups via confidence reweighting. *CoRR*, abs/2510.08696, 2025c. doi: 10.48550/ARXIV.2510.08696. URL https://doi.org/10.48550/arXiv.2510.08696.

Aleksander Ficek, Somshubra Majumdar, Vahid Noroozi, and Boris Ginsburg. Scoring verifiers: Evaluating synthetic verification in code and reasoning. *ArXiv*, abs/2502.13820, 2025. URL https://api.semanticscholar.org/CorpusID:277502118.

Lukas Fluri, Leon Lang, Alessandro Abate, Patrick Forré, David Krueger, and Joar Max Viktor Skalse. The perils of optimizing learned reward functions: Low training error does not guarantee low regret. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/fluri25a.html.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios Nikolas Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to evaluate reward models for RLHF. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=cbttLtO94Q.

Yanjun Fu, Ethan Baker, and Yizheng Chen. Constrained decoding for secure code generation. *CoRR*, abs/2405.00218, 2024. doi: 10.48550/ARXIV.2405.00218. URL https://doi.org/10.48550/arXiv.2405.00218.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. RLEF: grounding code llms in execution feedback with reinforcement learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/gehring25a.html.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *CoRR*, abs/2507.17746, 2025. doi: 10.48550/ARXIV.2507.17746. URL https://doi.org/10.48550/arXiv.2507.17746.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. *CoRR*, abs/2401.14196, 2024a. doi: 10.48550/ARXIV.2401.14196. URL https://doi.org/10.48550/arXiv.2401.14196.

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Reward reasoning model. *CoRR*, abs/2505.14674, 2025. doi: 10.48550/ARXIV.2505.14674. URL https://doi.org/10.48550/arXiv.2505.14674.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Ramé, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. Direct language model alignment from online AI feedback. *CoRR*, abs/2402.04792, 2024b. doi: 10.48550/ARXIV.2402.04792. URL https://doi.org/10.48550/arXiv.2402.04792.

Kavi Gupta, Peter Ebert Christensen, Xinyun Chen, and Dawn Song. Synthesize, execute and debug: Learning to repair for neural program synthesis. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/cd0f74b5955dc87fd0605745c4b49ee8-Abstract.html.

Srishti Gureja, Lester James Validad Miranda, Shayekh Bin Islam, Rishabh Maheshwary, Drishti Sharma, Gusti Triandi Winata, Nathan Lambert, Sebastian Ruder, Sara Hooker, and Marzieh Fadaee. M-rewardbench: Evaluating reward models in multilingual settings. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 43–58. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.3/.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet,

Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/0f69b4b96a46f284b726fbd70f74fb3b-Abstract-Datasets_and_Benchmarks_Track.html.

Michael Hassid, Tal Remez, Jonas Gehring, Roy Schwartz, and Yossi Adi. The larger the better? improved LLM code-generation via budget reallocation. *CoRR*, abs/2404.00725, 2024. doi: 10.48550/ARXIV.2404.00725. URL https://doi.org/10.48550/arXiv.2404.00725.

Jamie Hayes, Ilia Shumailov, William P. Porter, and Aneesh Pappu. Measuring memorization in RLHF for code completion. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=Tg8RLxpMDu.

Jingxuan He and Martin T. Vechev. Large language models for code: Security hardening and adversarial testing. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda (eds.), *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pp. 1865–1879. ACM, 2023. doi: 10.1145/3576915.3623175. URL https://doi.org/10.1145/3576915.3623175.

Jingxuan He, Mark Vero, Gabriela Krasnopolska, and Martin T. Vechev. Instruction tuning for secure code generation. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18043–18062. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/he24k.html.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *CoRR*, abs/2505.22312, 2025a. doi: 10.48550/ARXIV.2505.22312. URL https://doi.org/10.48550/arXiv.2505.22312.

Xinyi He, Qian Liu, Mingzhe Du, Lin Yan, Zhijie Fan, Yiming Huang, Zejian Yuan, and Zejun Ma. Swe-perf: Can language models optimize code performance on real-world repositories? *CoRR*, abs/2507.12415, 2025b. doi: 10.48550/ARXIV.2507.12415. URL https://doi.org/10.48550/arXiv.2507.12415.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G. Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Awadalla, and Manaal Faruqui. Advancedif: Rubric-based benchmarking and reinforcement learning for advancing LLM instruction following. *CoRR*, abs/2511.10507, 2025c. doi: 10.48550/ARXIV.2511.10507. URL https://doi.org/10.48550/arXiv.2511.10507.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan (eds.), *Proceedings of the Sixth Workshop on Statistical Machine Translation, WMT@EMNLP 2011, Edinburgh, Scotland, UK, July 30-31, 2011*, pp. 187–197. Association for Computational Linguistics, 2011. URL https://aclanthology.org/W11-2123/.

Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W. Bradley Knox, and Dorsa Sadigh. Contrastive preference learning: Learning from human feedback without RL. *CoRR*, abs/2310.13639, 2023. doi: 10.48550/ARXIV.2310.13639. URL https://doi.org/10.48550/arXiv.2310.13639.

Jan S. Hesthaven and Stefano Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.*, 363:55–78, 2018. doi: 10.1016/J.JCP.2018.02.037. URL https://doi.org/10.1016/j.jcp.2018.02.037.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556. URL https://doi.org/10.48550/arXiv.2203.15556.

Jiwoo Hong, Noah Lee, Rodrigo Martínez-Castaño, César Rodríguez, and James Thorne. Cross-lingual transfer of reward models in multilingual alignment. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational*

Linguistics: Human Language Technologies, NAACL 2025 - Volume 2: Short Papers, Albuquerque, New Mexico, April 29 - May 4, 2025, pp. 82–94. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-SHORT.8. URL https://doi.org/10.18653/v1/2025.naacl-short.8.

Xinyu Hu, Yancheng He, Weixun Wang, Tao Feng, Li Lin, Jiashun Liu, Wenbo Su, Bo Zheng, and Xiaojun Wan. CE-RM: A pointwise generative reward model optimized via two-stage rollout and unified criteria. *CoRR*, abs/2601.20327, 2026. doi: 10.48550/ARXIV.2601.20327. URL https://doi.org/10.48550/arXiv.2601.20327.

Dong Huang, Guangtao Zeng, Jianbo Dai, Meng Luo, Han Weng, Yuhao Qing, Heming Cui, Zhijiang Guo, and Jie M. Zhang. Effi-code: Unleashing code efficiency in language models. *CoRR*, abs/2410.10209, 2024. doi: 10.48550/ARXIV.2410.10209. URL https://doi.org/10.48550/arXiv.2410.10209.

Dong Huang, Guangtao Zeng, Jianbo Dai, Meng Luo, Han Weng, Yuhao Qing, Heming Cui, Zhijiang Guo, and Jie Zhang. Efficoder: Enhancing code generation in large language models through efficiency-aware fine-tuning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025a. URL https://proceedings.mlr.press/v267/huang25as.html.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Xianzhen Luo, Qiufeng Wang, YuanTao Fan, Qingfu Zhu, Zhaoxiang Zhang, Yang Gao, Jie Fu, Qian Liu, Houyi Li, Ge Zhang, Yuan Qi, Yinghui Xu, Wei Chu, and Zili Wang. Opencoder: The open cookbook for top-tier code large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 33167–33193. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.acl-long.1591/.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors. *CoRR*, abs/2508.12790, 2025c. doi: 10.48550/ARXIV.2508.12790. URL https://doi.org/10.48550/arXiv.2508.12790.

Jeevana Priya Inala, Chenglong Wang, Mei Yang, Andrés Codas, Mark Encarnación, Shuvendu K. Lahiri, Madanlal Musuvathi, and Jianfeng Gao. Fault-aware neural code rankers. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/5762c579d09811b7639be2389b3d07be-Abstract-Conference.html.

Nafis Tanveer Islam, Mohammad Bahrami Karkevandi, and Peyman Najafirad. Code security vulnerability repair using reinforcement learning with large language models. *CoRR*, abs/2401.07031, 2024. doi: 10.48550/ARXIV.2401.07031. URL https://doi.org/10.48550/arXiv.2401.07031.

Athul Paul Jacob, Yikang Shen, Gabriele Farina, and Jacob Andreas. The consensus game: Language model generation via equilibrium search. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=n9xeGcI4Yg.

Kush Jain, Gabriel Synnaeve, and Baptiste Rozière. Testgeneval: A real world unit test generation and test completion benchmark. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=7o6SG5gVev.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *CoRR*, abs/2310.11564, 2023. doi: 10.48550/ARXIV.2310.11564. URL https://doi.org/10.48550/arXiv.2310.11564.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.),

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, pp. 14165–14178. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.792. URL https://doi.org/10.18653/v1/2023.acl-long.792.

Dongfu Jiang, Yishan Li, Ge Zhang, Wenhao Huang, Bill Yuchen Lin, and Wenhu Chen. Tigerscore: Towards building explainable metric for all text generation tasks. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=EE1CBKC0SZ.

Hongchao Jiang, Yiming Chen, Yushi Cao, Hung-yi Lee, and Robby T. Tan. Codejudgebench: Benchmarking llm-as-a-judge for coding tasks. *CoRR*, abs/2507.10535, 2025. doi: 10.48550/ARXIV.2507.10535. URL https://doi.org/10.48550/arXiv.2507.10535.

Pengfei Jing, Mengyun Tang, Xiaorong Shi, Xing Zheng, Sen Nie, Shi Wu, Yong Yang, and Xiapu Luo. Secbench: A comprehensive multi-dimensional benchmarking dataset for llms in cybersecurity. *CoRR*, abs/2412.20787, 2024. doi: 10.48550/ARXIV.2412.20787. URL https://doi.org/10.48550/arXiv.2412.20787.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Amir Hossein Kargaran, Ayyoob Imani, François Yvon, and Hinrich Schütze. Glotlid: Language identification for low-resource languages. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Findings of ACL, pp. 6155–6218. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.410. URL https://doi.org/10.18653/v1/2023.findings-emnlp.410.

Mohammed Kharma, Soohyeon Choi, Mohammed AlKhanafseh, and David Mohaisen. Security and quality in llm-generated code: A multi-language, multi-model analysis. *CoRR*, abs/2502.01853, 2025. doi: 10.48550/ARXIV.2502.01853. URL https://doi.org/10.48550/arXiv.2502.01853.

Seungone Kim, Jamin Shin, Yejin Choi, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=8euJaTveKw.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 4334–4353. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.EMNLP-MAIN.248. URL https://doi.org/10.18653/v1/2024.emnlp-main.248.

Sunghwan Kim, Dongjin Kang, Taeyoon Kwon, Hyungjoo Chae, Dongha Lee, and Jinyoung Yeo. Rethinking reward model evaluation through the lens of reward overoptimization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 13252–13280. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.649/.

Kimi-Team. Kimi K2.5: visual agentic intelligence. *CoRR*, abs/2602.02276, 2026. doi: 10.48550/ARXIV.2602.02276. URL https://doi.org/10.48550/arXiv.2602.02276.

Hannah Rose Kirk, Alexander Whitefield, Paul Röttger, Andrew M. Bean, Katerina Margatina, Rafael Mosquera Gómez, Juan Ciro, Max Bartolo, Adina Williams, He He, Bertie Vidgen, and Scott Hale. The PRISM alignment dataset: What participatory, representative and individualised human feedback reveals about the subjective and multicultural alignment of large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/be2e1b68b44f2419e19f6c35a1b8cf35-Abstract-Datasets_and_Benchmarks_Track.html.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation and diversity. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=PXD3FAVHJT.

Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Vinayak Bhalerao, Christopher L. Buckley, Jason Phang, Samuel R. Bowman, and Ethan Perez. Pretraining language models with human preferences. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pp. 17506–17533. PMLR, 2023. URL https://proceedings.mlr.press/v202/korbak23a.html.

Emmanouil Koukoumidis. Using Code Review Repositories and Changelists to Train Large Language Models for Code Generation — tdcommons.org. https://www.tdcommons.org/dpubs_series/6027/, 2023.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=10uNUgI5Kl.

Viet Dac Lai, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In Yansong Feng and Els Lefever (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023 - System Demonstrations, Singapore, December 6-10, 2023*, pp. 318–327. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-DEMO.28. URL https://doi.org/10.18653/v1/2023.emnlp-demo.28.

Man Ho Lam, Chaozheng Wang, Jen-tse Huang, and Michael R. Lyu. CODECRASH: stress testing LLM reasoning under structural and semantic perturbations. *CoRR*, abs/2504.14119, 2025. doi: 10.48550/ARXIV.2504.14119. URL https://doi.org/10.48550/arXiv.2504.14119.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124, 2024. doi: 10.48550/ARXIV.2411.15124. URL https://doi.org/10.48550/arXiv.2411.15124.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James V. Miranda, Bill Yuchen Lin, Khyathi Raghavi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, volume NAACL 2025 of *Findings of ACL*, pp. 1755–1797. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.FINDINGS-NAACL.96. URL https://doi.org/10.18653/v1/2025.findings-naacl.96.

Djamel Rassem Lamouri, Iheb Nassim Aouadj, Smail Kourta, and Riyadh Baghdadi. Pearl: Automatic code optimization using deep reinforcement learning. In *Proceedings of the 39th ACM International Conference on Supercomputing, ICS 2025, Salt Lake City, UT, USA, June 8-11, 2025*, pp. 959–974. ACM, 2025. doi: 10.1145/3721145.3725766. URL https://doi.org/10.1145/3721145.3725766.

Anne Lauscher, Vinit Ravishankar, Ivan Vulic, and Goran Glavas. From zero to hero: On the limitations of zero-shot cross-lingual transfer with multilingual transformers. *CoRR*, abs/2005.00633, 2020. URL https://arxiv.org/abs/2005.00633.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu-Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/8636419dea1aa9fbd25fc4248e702da4-Abstract-Conference.html.

Hung Le, Doyen Sahoo, Yingbo Zhou, Caiming Xiong, and Silvio Savarese. INDICT: code generation with internal dialogues of critiques for both security and helpfulness. In Amir Globersons, Lester Mackey,

Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/9b812ee4b831c21e14156ced8659197c-Abstract-Conference.html.

Dongjun Lee, Changho Hwang, and Kimin Lee. Learning to generate unit test via adversarial reinforcement learning. *CoRR*, abs/2508.21107, 2025. doi: 10.48550/ARXIV.2508.21107. URL https://doi.org/10.48550/arXiv.2508.21107.

Seongyun Lee, Sue Hyun Park, Seungone Kim, and Minjoon Seo. Aligning to thousands of preferences via system message generalization. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/86c9df30129f7663ad4d429b6f80d461-Abstract-Conference.html.

Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J. Bentley, Samuel Bernard, Guillaume Beslon, David M. Bryson, Nick Cheney, Patryk Chrabaszcz, Antoine Cully, Stéphane Doncieux, Fred C. Dyer, Kai Olav Ellefsen, Robert Feldt, Stephan Fischer, Stephanie Forrest, Antoine Frénoy, Christian Gagné, Léni K. Le Goff, Laura M. Grabowski, Babak Hodjat, Frank Hutter, Laurent Keller, Carole Knibbe, Peter Krcah, Richard E. Lenski, Hod Lipson, Robert MacCurdy, Carlos Maestre, Risto Miikkulainen, Sara Mitri, David E. Moriarty, Jean-Baptiste Mouret, Anh Nguyen, Charles Ofria, Marc Parizeau, David P. Parsons, Robert T. Pennock, William F. Punch, Thomas S. Ray, Marc Schoenauer, Eric Schulte, Karl Sims, Kenneth O. Stanley, François Taddei, Danesh Tarapore, Simon Thibault, Richard A. Watson, Westley Weimer, and Jason Yosinski. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *Artif. Life*, 26(2): 274–306, 2020. doi: 10.1162/ARTL_A_00319. URL https://doi.org/10.1162/artl_a_00319.

Chaofan Li, Jianlyu Chen, Yingxia Shao, Defu Lian, and Zheng Liu. Towards A generalist code embedding model based on massive data synthesis. *CoRR*, abs/2505.12697, 2025a. doi: 10.48550/ARXIV.2505.12697. URL https://doi.org/10.48550/arXiv.2505.12697.

Dong Li, Meng Yan, Yaosheng Zhang, Zhongxin Liu, Chao Liu, Xiaohong Zhang, Ting Chen, and David Lo. Cosec: On-the-fly security hardening of code llms via supervised co-decoding. In Maria Christakis and Michael Pradel (eds.), *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2024, Vienna, Austria, September 16-20, 2024*, pp. 1428–1439. ACM, 2024a. doi: 10.1145/3650212.3680371. URL https://doi.org/10.1145/3650212.3680371.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. Generative judge for evaluating alignment. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=gtkFw6sZGS.

Kaiwen Li, Tao Zhang, and Rui Wang. Deep reinforcement learning for multiobjective optimization. *IEEE Trans. Cybern.*, 51(6):3103–3114, 2021. doi: 10.1109/TCYB.2020.2977661. URL https://doi.org/10.1109/TCYB.2020.2977661.

Lei Li, Yuancheng Wei, Zhihui Xie, Xuqing Yang, Yifan Song, Peiyi Wang, Chenxin An, Tianyu Liu, Sujian Li, Bill Yuchen Lin, Lingpeng Kong, and Qi Liu. Vl-rewardbench: A challenging benchmark for vision-language generative reward models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 24657–24668. Computer Vision Foundation / IEEE, 2025b. doi: 10.1109/CVPR52734.2025.02296. URL https://openaccess.thecvf.com/content/CVPR2025/html/Li_VL-RewardBench_A_Challenging_Benchmark_for_Vision-Language_Generative_Reward_Models_CVPR_2025_paper.html.

Weiyuan Li, Xintao Wang, Siyu Yuan, Rui Xu, Jiangjie Chen, Qingqing Dong, Yanghua Xiao, and Deqing Yang. Curse of knowledge: When complex evaluation context benefits yet biases LLM judges. *CoRR*, abs/2509.03419, 2025c. doi: 10.48550/ARXIV.2509.03419. URL https://doi.org/10.48550/arXiv.2509.03419.

Xiang Lisa Li, Vaishnavi Shrivastava, Siyan Li, Tatsunori Hashimoto, and Percy Liang. Benchmarking and improving generator-validator consistency of language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024c. URL

https://openreview.net/forum?id=phBS6YpTzC.

Youpeng Li, Weiliang Qi, Xuyu Wang, Fuxun Yu, and Xinda Wang. Revisiting pre-trained language models for vulnerability detection. *CoRR*, abs/2507.16887, 2025d. doi: 10.48550/ARXIV.2507.16887. URL https://doi.org/10.48550/arXiv.2507.16887.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *CoRR*, abs/2308.03281, 2023. doi: 10.48550/ARXIV.2308.03281. URL https://doi.org/10.48550/arXiv.2308.03281.

Zihao Li, Feihao Fang, Xitong Zhang, Jiaru Zou, Zhining Liu, Wei Xiong, Ziwei Wu, Baoyu Jing, and Jingrui He. Not all voices are rewarded equally: Probing and repairing reward models across human diversity. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 3426–3455. Association for Computational Linguistics, 2025e. URL https://aclanthology.org/2025.findings-emnlp.183/.

Soohan Lim, Joonghyuk Hahn, Hyunwoo Park, Sang-Ki Ko, and Yo-Sub Han. Contracteval: A benchmark for evaluating contract-satisfying assertions in code generation. 2025. URL https://api.semanticscholar.org/CorpusID:282064465.

Baijiong Lin, Weisen Jiang, Yuancheng Xu, Hao Chen, and Ying-Cong Chen. PARM: multi-objective test-time alignment via preference-aware autoregressive reward model. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025a. URL https://proceedings.mlr.press/v267/lin25h.html.

Haojia Lin, Xiaoyu Tan, Yulei Qin, Zihan Xu, Yuchen Shi, Zongyi Li, Gang Li, Shaofei Cai, Siqi Cai, Chaoyou Fu, Ke Li, and Xing Sun. Cuarewardbench: A benchmark for evaluating reward models on computer-using agent. *CoRR*, abs/2510.18596, 2025b. doi: 10.48550/ARXIV.2510.18596. URL https://doi.org/10.48550/arXiv.2510.18596.

Jiayi Lin, Yanlin Wang, Yibiao Yang, Lei Zhang, and Yutao Xie. Towards better code understanding in decoder-only models with contrastive learning. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 32006–32014. AAAI Press, 2026. doi: 10.1609/AAAI.V40I38.40471. URL https://doi.org/10.1609/aaai.v40i38.40471.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *CoRR*, abs/2410.18451, 2024a. doi: 10.48550/ARXIV.2410.18451. URL https://doi.org/10.48550/arXiv.2410.18451.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *CoRR*, abs/2507.01352, 2025a. doi: 10.48550/ARXIV.2507.01352. URL https://doi.org/10.48550/arXiv.2507.01352.

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. RLTF: reinforcement learning from unit test feedback. *Trans. Mach. Learn. Res.*, 2023, 2023a. URL https://openreview.net/forum?id=hjYmsV6nXZ.

Jiawei Liu, Thanh Nguyen, Mingyue Shang, Hantian Ding, Xiaopeng Li, Yu Yu, Varun Kumar, and Zijian Wang. Learning code preference via synthetic evolution. *CoRR*, abs/2410.03837, 2024b. doi: 10.48550/ARXIV.2410.03837. URL https://doi.org/10.48550/arXiv.2410.03837.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation. *CoRR*, abs/2408.06450, 2024c. doi: 10.48550/ARXIV.2408.06450. URL https://doi.org/10.48550/arXiv.2408.06450.

Jiawei Liu, Nirav Diwan, Zhe Wang, Haoyu Zhai, Xiaona Zhou, Kiet A. Nguyen, Tianjiao Yu, Muntasir Wahed, Yinlin Deng, Hadjer Benkraouda, Yuxiang Wei, Lingming Zhang, Ismini Lourentzou, and Gang Wang. Purpcode: Reasoning for safer code generation. *CoRR*, abs/2507.19060, 2025b. doi: 10.48550/

ARXIV.2507.19060. URL https://doi.org/10.48550/arXiv.2507.19060.

Minqian Liu, Ying Shen, Zhiyang Xu, Yixin Cao, Eunah Cho, Vaibhav Kumar, Reza Ghanadan, and Lifu Huang. X-eval: Generalizable multi-aspect text evaluation via augmented instruction tuning with auxiliary evaluation aspects. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 8560–8579. Association for Computational Linguistics, 2024d. doi: 10.18653/V1/2024.NAACL-LONG.473. URL https://doi.org/10.18653/v1/2024.naacl-long.473.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025c. URL https://openreview.net/forum?id=5BjQOUXq7i.

Shukai Liu, Linzheng Chai, Jian Yang, Jiajun Shi, He Zhu, Liran Wang, Ke Jin, Wei Zhang, Hualei Zhu, Shuyue Guo, Tao Sun, Jiaheng Liu, Yunlong Duan, Yu Hao, Liqun Yang, Guanglin Niu, Ge Zhang, and Zhoujun Li. Mdeval: Massively multilingual code debugging. *CoRR*, abs/2411.02310, 2024e. doi: 10.48550/ARXIV.2411.02310. URL https://doi.org/10.48550/arXiv.2411.02310.

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. *CoRR*, abs/2510.07743, 2025d. doi: 10.48550/ARXIV.2510.07743. URL https://doi.org/10.48550/arXiv.2510.07743.

Tianqi Liu, Wei Xiong, Jie Ren, Lichang Chen, Junru Wu, Rishabh Joshi, Yang Gao, Jiaming Shen, Zhen Qin, Tianhe Yu, Daniel Sohn, Anastasia Makarova, Jeremiah Zhe Liu, Yuan Liu, Bilal Piot, Abe Ittycheriah, Aviral Kumar, and Mohammad Saleh. RRM: robust reward model training mitigates reward hacking. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025e. URL https://openreview.net/forum?id=88AS5MQnmC.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 2511–2522. Association for Computational Linguistics, 2023b. doi: 10.18653/V1/2023.EMNLP-MAIN.153. URL https://doi.org/10.18653/v1/2023.emnlp-main.153.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Rm-bench: Benchmarking reward models of language models with subtlety and style. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025f. URL https://openreview.net/forum?id=QEHrmQPBdd.

Yifei Liu, Li Lyna Zhang, Yi Zhu, Bingcheng Dong, Xudong Zhou, Ning Shang, Fan Yang, and Mao Yang. rstar-coder: Scaling competitive code reasoning with a large-scale verified dataset. *CoRR*, abs/2505.21297, 2025g. doi: 10.48550/ARXIV.2505.21297. URL https://doi.org/10.48550/arXiv.2505.21297.

Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulic, Anna Korhonen, and Nigel Collier. Aligning with human judgement: The role of pairwise preference in large language model evaluators. *CoRR*, abs/2403.16950, 2024f. doi: 10.48550/ARXIV.2403.16950. URL https://doi.org/10.48550/arXiv.2403.16950.

Zhihan Liu, Shenao Zhang, and Zhaoran Wang. DSTC: direct preference learning with only self-generated tests and code to improve code lms. *CoRR*, abs/2411.13611, 2024g. doi: 10.48550/ARXIV.2411.13611. URL https://doi.org/10.48550/arXiv.2411.13611.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *CoRR*, abs/2504.02495, 2025h. doi: 10.48550/ARXIV.2504.02495. URL https://doi.org/10.48550/arXiv.2504.02495.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Xingzhou Lou, Dong Yan, Wei Shen, Yuzi Yan, Jian Xie, and Junge Zhang. Uncertainty-aware reward model: Teaching reward models to know what is unknown. *CoRR*, abs/2410.00847, 2024. doi: 10.48550/ARXIV.

2410.00847. URL https://doi.org/10.48550/arXiv.2410.00847.

Dan Lu and Daniel M. Ricciuto. Efficient surrogate modeling methods for large-scale earth system models based on machine learning techniques. *CoRR*, abs/1901.05125, 2019. URL http://arxiv.org/abs/1901.05125.

Bohan Lyu, Siqiao Huang, Zichen Liang, Qian Sun, and Jiaming Zhang. Surge: On the potential of large language models as general-purpose surrogate code executors. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 3268–3308. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.162. URL https://doi.org/10.18653/v1/2025.emnlp-main.162.

Jeffrey Jian Ma, Milad Hashemi, Amir Yazdanbakhsh, Kevin Swersky, Ofir Press, Enhui Li, Vijay Janapa Reddi, and Parthasarathy Ranganathan. Swe-fficiency: Can language models optimize real-world repositories on real workloads? *CoRR*, abs/2511.06090, 2025a. doi: 10.48550/ARXIV.2511.06090. URL https://doi.org/10.48550/arXiv.2511.06090.

Lezhi Ma, Shangqing Liu, Lei Bu, Shangru Li, Yida Wang, and Yang Liu. Speceval: Evaluating code comprehension in large language models via program specifications. *CoRR*, abs/2409.12866, 2024. doi: 10.48550/ARXIV.2409.12866. URL https://doi.org/10.48550/arXiv.2409.12866.

Zeyao Ma, Xiaokang Zhang, Jing Zhang, Jifan Yu, Sijia Luo, and Jie Tang. Dynamic scaling of unit tests for code reward modeling. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 6917–6935. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.acl-long.343/.

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. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Dakota Mahan, Duy Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *CoRR*, abs/2410.12832, 2024. doi: 10.48550/ARXIV.2410.12832. URL https://doi.org/10.48550/arXiv.2410.12832.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. *CoRR*, abs/2506.01937, 2025. doi: 10.48550/ARXIV.2506.01937. URL https://doi.org/10.48550/arXiv.2506.01937.

Alexandre Matton, Tom Sherborne, Dennis Aumiller, Elena Tommasone, Milad Alizadeh, Jingyi He, Raymond Ma, Maxime Voisin, Ellen Gilsenan-McMahon, and Matthias Gallé. On leakage of code generation evaluation datasets. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, volume EMNLP 2024 of *Findings of ACL*, pp. 13215–13223. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-EMNLP.772. URL https://doi.org/10.18653/v1/2024.findings-emnlp.772.

MiniMax. MiniMax M2.5: Built for Real-World Productivity. — minimax.io. https://www.minimax.io/news/minimax-m25, 2025.

Jiwon Moon, Yerin Hwang, Dongryeol Lee, Taegwan Kang, Yongil Kim, and Kyomin Jung. Don't judge code by its cover: Exploring biases in LLM judges for code evaluation. *CoRR*, abs/2505.16222, 2025. doi: 10.48550/ARXIV.2505.16222. URL https://doi.org/10.48550/arXiv.2505.16222.

Jiwon Moon, Yerin Hwang, Dongryeol Lee, Taegwan Kang, Yongil Kim, and Kyomin Jung. Don't judge code by its cover: Exploring biases in LLM judges for code evaluation. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026, Rabat, Morocco, March 24-29, 2026*, pp. 1364–1389. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-eacl.70/.

Seungjun Moon, Yongho Song, Hyungjoo Chae, Dongjin Kang, Taeyoon Kwon, Kai Tzu-iunn Ong, Seung-won Hwang, and Jinyoung Yeo. Coffee: Boost your code llms by fixing bugs with feedback. *CoRR*, abs/2311.07215, 2023. doi: 10.48550/ARXIV.2311.07215. URL https://doi.org/10.48550/arXiv.2311.07215.

Niklas Muennighoff, Qian Liu, Armel Randy Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=mw1PWNSWZP.

Niels Mündler, Mark Niklas Müller, Jingxuan He, and Martin T. Vechev. Swt-bench: Testing and validating real-world bug-fixes with code agents. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/94f093b41fc2666376fb1f667fe282f3-Abstract-Conference.html.

Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Côme Fiegel, Andrea Michi, Marco Selvi, Sertan Girgin, Nikola Momchev, Olivier Bachem, Daniel J. Mankowitz, Doina Precup, and Bilal Piot. Nash learning from human feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 36743–36768. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/munos24a.html.

Duy Nguyen, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Multi-attribute steering of language models via targeted intervention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 20619–20634. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1007/.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-Tau Yih, Sida I. Wang, and Xi Victoria Lin. LEVER: learning to verify language-to-code generation with execution. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26106–26128. PMLR, 2023. URL https://proceedings.mlr.press/v202/ni23b.html.

Ansong Ni, Miltiadis Allamanis, Arman Cohan, Yinlin Deng, Kensen Shi, Charles Sutton, and Pengcheng Yin. Next: Teaching large language models to reason about code execution. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37929–37956. PMLR / OpenReview.net, 2024a. URL https://proceedings.mlr.press/v235/ni24a.html.

Ansong Ni, Pengcheng Yin, Yilun Zhao, Martin Riddell, Troy Feng, Rui Shen, Stephen Yin, Ye Liu, Semih Yavuz, Caiming Xiong, Shafiq Joty, Yingbo Zhou, Dragomir Radev, and Arman Cohan. L2CEval: Evaluating language-to-code generation capabilities of large language models. *Trans. Assoc. Comput. Linguistics*, 12: 1311–1329, 2024b. doi: 10.1162/TACL_A_00705. URL https://doi.org/10.1162/tacl_a_00705.

Daniel Nichols, Pranav Polasam, Harshitha Menon, Aniruddha Marathe, Todd Gamblin, and Abhinav Bhatele. Performance-aligned llms for generating fast code. *CoRR*, abs/2404.18864, 2024. doi: 10.48550/ARXIV.2404.18864. URL https://doi.org/10.48550/arXiv.2404.18864.

Henrique Gomes Nunes, Eduardo Figueiredo, Larissa Rocha Soares, Sarah Nadi, Fischer Ferreira, and Geanderson E. dos Santos. Evaluating the effectiveness of llms in fixing maintainability issues in real-world projects. In *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2025, Montreal, QC, Canada, March 4-7, 2025*, pp. 669–680. IEEE, 2025. doi: 10.1109/SANER64311.2025.00069. URL https://doi.org/10.1109/SANER64311.2025.00069.

Daniil Orel, Indraneil Paul, Iryna Gurevych, and Preslav Nakov. Droid: A resource suite for ai-generated code detection. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.),

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025, pp. 31263–31289. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.1593. URL https://doi.org/10.18653/v1/2025.emnlp-main.1593.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=JYtwGwIL7ye.

Junsoo Park, Seungyeon Jwa, Meiying Ren, Daeyoung Kim, and Sanghyuk Choi. Offsetbias: Leveraging debiased data for tuning evaluators. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Findings of ACL, pp. 1043–1067. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-EMNLP.57. URL https://doi.org/10.18653/v1/2024.findings-emnlp.57.

Debalina Ghosh Paul, Hong Zhu, and Ian Bayley. Investigating the smells of LLM generated code. *CoRR*, abs/2510.03029, 2025. doi: 10.48550/ARXIV.2510.03029. URL https://doi.org/10.48550/arXiv.2510.03029.

Indraneil Paul, Goran Glavas, and Iryna Gurevych. Ircoder: Intermediate representations make language models robust multilingual code generators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 15023–15041. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.802. URL https://doi.org/10.18653/v1/2024.acl-long.802.

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of github copilot's code contributions. *Commun. ACM*, 68(2):96–105, 2025. doi: 10.1145/3610721. URL https://doi.org/10.1145/3610721.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019. URL http://arxiv.org/abs/1910.00177.

Yun Peng, Akhilesh Deepak Gotmare, Michael R. Lyu, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Perfcodegen: Improving performance of LLM generated code with execution feedback. In *IEEE/ACM Second International Conference on AI Foundation Models and Software Engineering, Forge@ICSE 2025, Ottawa, ON, Canada, April 27-28, 2025*, pp. 1–13. IEEE, 2025. doi: 10.1109/FORGE66646.2025.00008. URL https://doi.org/10.1109/Forge66646.2025.00008.

Archiki Prasad, Elias Stengel-Eskin, Justin Chih-Yao Chen, Zaid Khan, and Mohit Bansal. Learning to generate unit tests for automated debugging. *CoRR*, abs/2502.01619, 2025. doi: 10.48550/ARXIV.2502.01619. URL https://doi.org/10.48550/arXiv.2502.01619.

Julian Aron Prenner and Romain Robbes. Runbugrun - an executable dataset for automated program repair. *CoRR*, abs/2304.01102, 2023. doi: 10.48550/ARXIV.2304.01102. URL https://doi.org/10.48550/arXiv.2304.01102.

Ruchir Puri, David S. Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir R. Choudhury, Lindsey Decker, Veronika Thost, Luca Buratti, Saurabh Pujar, Shyam Ramji, Ulrich Finkler, Susan Malaika, and Frederick Reiss. Codenet: A large-scale AI for code dataset for learning a diversity of coding tasks. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.

neurips.cc/paper/2021/hash/a5bfc9e07964f8dddeb95fc584cd965d-Abstract-round2.html.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, volume NAACL 2024 of *Findings of ACL*, pp. 1504–1518. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.97. URL https://doi.org/10.18653/v1/2024.findings-naacl.97.

Muzi Qu, Jie Liu, Liangyi Kang, Shuyi Ling, Shuai Wang, Dan Ye, and Tao Huang. Scodegen: A real-time trustworthy constrained decoding framework for secure code generation with llms. In *24th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2025, Guiyang, China, November 14-17, 2025*, pp. 492–503. IEEE, 2025. doi: 10.1109/TRUSTCOM66490.2025.00061. URL https://doi.org/10.1109/Trustcom66490.2025.00061.

Jiazheng Quan, Xiaodong Li, Bin Wang, Guo An, Like Liu, Degen Huang, Lingfeng Liu, and Chengbin Hou. Learning to generate secure code via token-level rewards. 2026. URL https://api.semanticscholar.org/CorpusID:286171293.

Prateek Rajput, Yewei Song, Abdoul Aziz Bonkoungou, Iyiola E. Olatunji, Abdoul Kader Kaboré, Jacques Klein, and Tegawendé F. Bissyandé. Correctness isnt efficiency: Runtime memory divergence in llm-generated code. *CoRR*, abs/2601.01215, 2026. doi: 10.48550/ARXIV.2601.01215. URL https://doi.org/10.48550/arXiv.2601.01215.

Alexandre Ramé, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/e12a3b98b67e8395f639fde4c2b03168-Abstract-Conference.html.

Muhammad Shihab Rashid, Christian Bock, Yuan Zhuang, Alexander Buchholz, Timothy B. Esler, Simon Valentin, Luca Franceschi, Martin Wistuba, Prabhu Teja Sivaprasad, Woo Jung Kim, Anoop Deoras, Giovanni Zappella, and Laurent Callot. Swe-polybench: A multi-language benchmark for repository level evaluation of coding agents. *CoRR*, abs/2504.08703, 2025. doi: 10.48550/ARXIV.2504.08703. URL https://doi.org/10.48550/arXiv.2504.08703.

Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D. Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective. *CoRR*, abs/2503.15477, 2025. doi: 10.48550/ARXIV.2503.15477. URL https://doi.org/10.48550/arXiv.2503.15477.

Francisco Ribeiro, Claudio Spiess, Premkomar Thomas Devanbu, and Sarah Nadi. On llms'internal representation of code correctness. 2025. URL https://api.semanticscholar.org/CorpusID:283693749.

Gleb Rodionov and Liudmila Prokhorenkova. Neural algorithmic reasoning without intermediate supervision. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a2370db7c99791ad5d9f3ef48ad6d464-Abstract-Conference.html.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023. doi: 10.48550/ARXIV.2308.12950. URL https://doi.org/10.48550/arXiv.2308.12950.

Jon Saad-Falcon, Rajan Vivek, William Berrios, Nandita Shankar Naik, Matija Franklin, Bertie Vidgen, Amanpreet Singh, Douwe Kiela, and Shikib Mehri. LMUNIT: fine-grained evaluation with natural language unit tests. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet

Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 3303–3324. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.176/.

Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason E. Weston, and Tianlu Wang. Learning to plan & reason for evaluation with thinking-llm-as-a-judge. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/saha25b.html.

Shivanshu Shekhar, Shreyas Singh, and Tong Zhang. SEE-DPO: self entropy enhanced direct preference optimization. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=xQbRFHfgGL.

Jiaming Shen, Ran Xu, Yennie Jun, Zhen Qin, Tianqi Liu, Carl Yang, Yi Liang, Simon Baumgartner, and Michael Bendersky. Boosting reward model with preference-conditional multi-aspect synthetic data generation. *CoRR*, abs/2407.16008, 2024. doi: 10.48550/ARXIV.2407.16008. URL https://doi.org/10.48550/arXiv.2407.16008.

Lingfeng Shen, Sihao Chen, Linfeng Song, Lifeng Jin, Baolin Peng, Haitao Mi, Daniel Khashabi, and Dong Yu. The trickle-down impact of reward (in-)consistency on RLHF. *CoRR*, abs/2309.16155, 2023. doi: 10.48550/ARXIV.2309.16155. URL https://doi.org/10.48550/arXiv.2309.16155.

Idan Shenfeld, Felix Faltings, Pulkit Agrawal, and Aldo Pacchiano. Language model personalization via reward factorization. *CoRR*, abs/2503.06358, 2025. doi: 10.48550/ARXIV.2503.06358. URL https://doi.org/10.48550/arXiv.2503.06358.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I. Wang. Natural language to code translation with execution. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 3533–3546. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.231. URL https://doi.org/10.18653/v1/2022.emnlp-main.231.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Parshin Shojaee, Aneesh Jain, Sindhu Tipirneni, and Chandan K. Reddy. Execution-based code generation using deep reinforcement learning. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=0XBuaxqEcG.

Alexander Shypula, Aman Madaan, Yimeng Zeng, Uri Alon, Jacob R. Gardner, Yiming Yang, Milad Hashemi, Graham Neubig, Parthasarathy Ranganathan, Osbert Bastani, and Amir Yazdanbakhsh. Learning performance-improving code edits. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ix7rLVHXyY.

Mohammed Latif Siddiq and Joanna C. S. Santos. Securityeval dataset: mining vulnerability examples to evaluate machine learning-based code generation techniques. *Proceedings of the 1st International Workshop on Mining Software Repositories Applications for Privacy and Security*, 2022. URL https://api.semanticscholar.org/CorpusID:252125684.

Suryansh Singh Sijwali and Suman Saha. Securecoderl: Security-aware reinforcement learning for code generation with partial-credit rewards. *CoRR*, abs/2601.01184, 2026. doi: 10.48550/ARXIV.2601.01184. URL https://doi.org/10.48550/arXiv.2601.01184.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T. Parisi, Abhishek Kumar, Alexander A. Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman

Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=lNAyUngGFK.

Manav Singhal, Tushar Aggarwal, Abhijeet Awasthi, Nagarajan Natarajan, and Aditya Kanade. Nofuneval: Funny how code lms falter on requirements beyond functional correctness. *CoRR*, abs/2401.15963, 2024. doi: 10.48550/ARXIV.2401.15963. URL https://doi.org/10.48550/arXiv.2401.15963.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/ARXIV.2408.03314. URL https://doi.org/10.48550/arXiv.2408.03314.

Guijin Son, Hyunwoo Ko, Hoyoung Lee, Yewon Kim, and Seunghyeok Hong. Llm-as-a-judge & reward model: What they can and cannot do. *CoRR*, abs/2409.11239, 2024. doi: 10.48550/ARXIV.2409.11239. URL https://doi.org/10.48550/arXiv.2409.11239.

Yuda Song, Gokul Swamy, Aarti Singh, J. Andrew Bagnell, and Wen Sun. The importance of online data: Understanding preference fine-tuning via coverage. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/16c628ab12dc4caca8e7712affa6c767-Abstract-Conference.html.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding RL with verifiable rewards across diverse domains. *CoRR*, abs/2503.23829, 2025. doi: 10.48550/ARXIV.2503.23829. URL https://doi.org/10.48550/arXiv.2503.23829.

Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Daniel Cox, Yiming Yang, and Chuang Gan. SALMON: self-alignment with instructable reward models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=xJbsmB8UMx.

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J. Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *CoRR*, abs/2503.01067, 2025. doi: 10.48550/ARXIV.2503.01067. URL https://doi.org/10.48550/arXiv.2503.01067.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47441–47474. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/tajwar24a.html.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca A. Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

Leitian Tao, Xiang Chen, Tong Yu, Tung Mai, Ryan A. Rossi, Yixuan Li, and Saayan Mitra. Codelutra: Boosting LLM code generation via preference-guided refinement. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=IGsEgWM4to.

Gemma Team. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024. doi: 10.48550/ARXIV.2408.00118. URL https://doi.org/10.48550/arXiv.2408.00118.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Hung To, Minh Nguyen, and Nghi Bui. Functional overlap reranking for neural code generation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 3686–3704. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.220. URL https://doi.org/10.18653/v1/2024.findings-acl.220.

Weixi Tong and Tianyi Zhang. Codejudge: Evaluating code generation with large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 20032–20051. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.1118. URL https://doi.org/10.18653/v1/2024.emnlp-main.1118.

Vatsal Venkatkrishna, Indraneil Paul, and Iryna Gurevych. Aletheia: What makes RLVR for code verifiers tick? *CoRR*, abs/2601.12186, 2026. doi: 10.48550/ARXIV.2601.12186. URL https://doi.org/10.48550/arXiv.2601.12186.

Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. Foundational autoraters: Taming large language models for better automatic evaluation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 17086–17105. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.949. URL https://doi.org/10.18653/v1/2024.emnlp-main.949.

Siddhant Waghjale, Vishruth Veerendranath, Zhiruo Wang, and Daniel Fried. ECCO: can we improve model-generated code efficiency without sacrificing functional correctness? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 15362–15376. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.859. URL https://doi.org/10.18653/v1/2024.emnlp-main.859.

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Secrets of RLHF in large language models part II: reward modeling. *CoRR*, abs/2401.06080, 2024a. doi: 10.48550/ARXIV.2401.06080. URL https://doi.org/10.48550/arXiv.2401.06080.

Binghai Wang, Runji Lin, Keming Lu, Le Yu, Zhenru Zhang, Fei Huang, Chujie Zheng, Kai Dang, Yang Fan, Xingzhang Ren, An Yang, Binyuan Hui, Dayiheng Liu, Tao Gui, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Bowen Yu, Jingren Zhou, and Junyang Lin. Worldpm: Scaling human preference modeling. *CoRR*, abs/2505.10527, 2025a. doi: 10.48550/ARXIV.2505.10527. URL https://doi.org/10.48550/arXiv.2505.10527.

Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 8642–8655. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.ACL-LONG.468. URL https://doi.org/10.18653/v1/2024.acl-long.468.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, volume EMNLP 2024 of *Findings of ACL*, pp. 10582–10592. Association for Computational Linguistics, 2024c. doi: 10.18653/V1/2024.FINDINGS-EMNLP.620. URL https://doi.org/10.18653/v1/2024.findings-emnlp.620.

Jiayou Wang, Rundong Liu, Yue Hu, Huijia Wu, and Zhaofeng He. Secdecoding: Steerable decoding for safer LLM generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 20504–20521. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.findings-emnlp.1118/.

Nan Wang, Yafei Liu, Chen Chen, and Haonan Lu. Genx: Mastering code and test generation with execution feedback. *CoRR*, abs/2412.13464, 2024d. doi: 10.48550/ARXIV.2412.13464. URL https://doi.org/10.48550/arXiv.2412.13464.

Peifeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. Direct judgement preference optimization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.),

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025, pp. 1979–2009. Association for Computational Linguistics, 2025c. doi: 10.18653/V1/2025.EMNLP-MAIN.103. URL https://doi.org/10.18653/v1/2025.emnlp-main.103.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 9426–9439. Association for Computational Linguistics, 2024e. doi: 10.18653/V1/2024.ACL-LONG.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

Qian Wang, Zhanzhi Lou, Zhenheng Tang, Nuo Chen, Xuandong Zhao, Wenxuan Zhang, Dawn Song, and Bingsheng He. Assessing judging bias in large reasoning models: An empirical study. *CoRR*, abs/2504.09946, 2025d. doi: 10.48550/ARXIV.2504.09946. URL https://doi.org/10.48550/arXiv.2504.09946.

Wenhan Wang, Chenyuan Yang, Zhijie Wang, Yuheng Huang, Zhaoyang Chu, Da Song, Lingming Zhang, An Ran Chen, and Lei Ma. TESTEVAL: benchmarking large language models for test case generation. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, volume NAACL 2025 of *Findings of ACL*, pp. 3547–3562. Association for Computational Linguistics, 2025e. doi: 10.18653/V1/2025.FINDINGS-NAACL.197. URL https://doi.org/10.18653/v1/2025.findings-naacl.197.

Xin Wang, Yasheng Wang, Yao Wan, Fei Mi, Yitong Li, Pingyi Zhou, Jin Liu, Hao Wu, Xin Jiang, and Qun Liu. Compilable neural code generation with compiler feedback. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, volume ACL 2022 of *Findings of ACL*, pp. 9–19. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-ACL.2. URL https://doi.org/10.18653/v1/2022.findings-acl.2.

Yuanhao Wang, Qinghua Liu, and Chi Jin. Is RLHF more difficult than standard rl? A theoretical perspective. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/efb9629755e598c4f261c44aeb6fde5e-Abstract-Conference.html.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models. *CoRR*, abs/2406.08673, 2024f. doi: 10.48550/ARXIV.2406.08673. URL https://doi.org/10.48550/arXiv.2406.08673.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages. *CoRR*, abs/2505.11475, 2025f. doi: 10.48550/ARXIV.2505.11475. URL https://doi.org/10.48550/arXiv.2505.11475.

Zihan Wang, Siyao Liu, Yang Sun, Ming Ding, and Hongyan Li. Codecontests+: High-quality test case generation for competitive programming. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 5576–5600. Association for Computational Linguistics, 2025g. URL https://aclanthology.org/2025.findings-emnlp.299/.

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. On negative interference in multilingual models: Findings and A meta-learning treatment. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 4438–4450. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.359. URL https://doi.org/10.18653/v1/2020.emnlp-main.359.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova,

and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 2526–2547. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.127/.

Yuxiang Wei, Federico Cassano, Jiawei Liu, Yifeng Ding, Naman Jain, Zachary Mueller, Harm de Vries, Leandro von Werra, Arjun Guha, and Lingming Zhang. Selfcodealign: Self-alignment for code generation. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/72da102da91a8042a0b2aa968429a9f9-Abstract-Conference.html.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with oss-instruct. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 52632–52657. PMLR / OpenReview.net, 2024b. URL https://proceedings.mlr.press/v235/wei24h.html.

Bosi Wen, Yilin Niu, Cunxiang Wang, Xiaoying Ling, Ying Zhang, Pei Ke, Hongning Wang, and Minlie Huang. If-rewardbench: Benchmarking judge models for instruction-following evaluation. 2026. URL https://api.semanticscholar.org/CorpusID:286255879.

Xueru Wen, Jie Lou, Yaojie Lu, Hongyu Lin, XingYu, Xinyu Lu, Ben He, Xianpei Han, Debing Zhang, and Le Sun. Rethinking reward model evaluation: Are we barking up the wrong tree? In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=Cnwz9jONi5.

Martin Weyssow, Aton Kamanda, and Houari A. Sahraoui. Codeultrafeedback: An llm-as-a-judge dataset for aligning large language models to coding preferences. *CoRR*, abs/2403.09032, 2024. doi: 10.48550/ARXIV.2403.09032. URL https://doi.org/10.48550/arXiv.2403.09032.

Martin Weyssow, Chengran Yang, Junkai Chen, Yikun Li, Huihui Huang, Ratnadira Widyasari, Han Wei Ang, Frank Liauw, Eng Lieh Ouh, Lwin Khin Shar, and David Lo. R2vul: Learning to reason about software vulnerabilities with reinforcement learning and structured reasoning distillation. *CoRR*, abs/2504.04699, 2025. doi: 10.48550/ARXIV.2504.04699. URL https://doi.org/10.48550/arXiv.2504.04699.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: incentivizing thinking in llm-as-a-judge via reinforcement learning. *CoRR*, abs/2505.10320, 2025. doi: 10.48550/ARXIV.2505.10320. URL https://doi.org/10.48550/arXiv.2505.10320.

Genta Indra Winata, David Anugraha, Lucky Susanto, Garry Kuwanto, and Derry Tanti Wijaya. Metametrics: Calibrating metrics for generation tasks using human preferences. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=slO3xTt4CG.

Mingkang Wu, Devin White, Evelyn Rose, Vernon Lawhern, Nicholas R. Waytowich, and Yongcan Cao. Multi-task reward learning from human ratings. *CoRR*, abs/2506.09183, 2025a. doi: 10.48550/ARXIV.2506.09183. URL https://doi.org/10.48550/arXiv.2506.09183.

Tianyi Wu, Mingzhe Du, Yue Liu, Cheng-Lin Yang, Terry Yue Zhuo, Jiaheng Zhang, and See kiong Ng. Secure code generation via online reinforcement learning with vulnerability reward model. 2026. URL https://api.semanticscholar.org/CorpusID:285451989.

Yue Wu, Minghao Han, Ruiyin Li, Peng Liang, Amjed Tahir, Zengyang Li, Qiong Feng, and Mojtaba Shahin. Fasterpy: An llm-based code execution efficiency optimization framework. *CoRR*, abs/2512.22827, 2025b. doi: 10.48550/ARXIV.2512.22827. URL https://doi.org/10.48550/arXiv.2512.22827.

Yutong Wu, Di Huang, Wenxuan Shi, Wei Wang, Yewen Pu, Lingzhe Gao, Shihao Liu, Ziyuan Nan, Kaizhao Yuan, Rui Zhang, Xishan Zhang, Zidong Du, Qi Guo, Dawei Yin, Xing Hu, and Yunji Chen. Inversecoder: Self-improving instruction-tuned code llms with inverse-instruct. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial*

Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025, pp. 25525–25533. AAAI Press, 2025c. doi: 10.1609/AAAI.V39I24.34742. URL https://doi.org/10.1609/aaai.v39i24.34742.

Zhaofeng Wu, Ananth Balashankar, Yoon Kim, Jacob Eisenstein, and Ahmad Beirami. Reuse your rewards: Reward model transfer for zero-shot cross-lingual alignment. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 1332–1353. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.79. URL https://doi.org/10.18653/v1/2024.emnlp-main.79.

Zhaofeng Wu, Michihiro Yasunaga, Andrew Cohen, Yoon Kim, Asli Celikyilmaz, and Marjan Ghazvininejad. rewordbench: Benchmarking and improving the robustness of reward models with transformed inputs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 3383–3409. Association for Computational Linguistics, 2025d. doi: 10.18653/V1/2025.EMNLP-MAIN.167. URL https://doi.org/10.18653/v1/2025.emnlp-main.167.

Xin Xie, Jiaxian Guo, and Dong Gong. Hyperalign: Hypernetwork for efficient test-time alignment of diffusion models. *CoRR*, abs/2601.15968, 2026. doi: 10.48550/ARXIV.2601.15968. URL https://doi.org/10.48550/arXiv.2601.15968.

Zhihui Xie, Jiahui Gao, Lei Li, Zhenguo Li, Qi Liu, and Lingpeng Kong. Jailbreaking as a reward misspecification problem. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=uBnM3EFovQ.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for RLHF under kl-constraint. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54715–54754. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/xiong24a.html.

Xiangzhe Xu, Zian Su, Jinyao Guo, Kaiyuan Zhang, Zhenting Wang, and Xiangyu Zhang. Prosec: Fortifying code llms with proactive security alignment. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025a. URL https://proceedings.mlr.press/v267/xu25aa.html.

Yinglun Xu, Hangoo Kang, Tarun Suresh, Yuxuan Wan, and Gagandeep Singh. Learning a pessimistic reward model in RLHF. *CoRR*, abs/2505.20556, 2025b. doi: 10.48550/ARXIV.2505.20556. URL https://doi.org/10.48550/arXiv.2505.20556.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025c. URL https://openreview.net/forum?id=Pnk7vMbznK.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pp. 6980–7008. Association for Computational Linguistics, 2025d. doi: 10.18653/V1/2025.FINDINGS-ACL.365. URL https://doi.org/10.18653/v1/2025.findings-acl.365.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html.

Aashish Yadavally, Hoan Nguyen, Laurent Callot, and Gauthier Guinet. Large language model critics for execution-free evaluation of code changes. *CoRR*, abs/2501.16655, 2025. doi: 10.48550/ARXIV.2501.16655. URL https://doi.org/10.48550/arXiv.2501.16655.

Yujun Yan, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Hashemi. Neural execution engines: Learning to execute subroutines. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8b9abffb45bf79a630fb613dcd23449-Abstract.html.

Jinluan Yang, Dingnan Jin, Anke Tang, Li Shen, Didi Zhu, Zhengyu Chen, Daixin Wang, Qing Cui, Zhiqiang Zhang, Jun Zhou, Fei Wu, and Kun Kuang. Mix data or merge models? balancing the helpfulness, honesty, and harmlessness of large language model via model merging. *CoRR*, abs/2502.06876, 2025a. doi: 10.48550/ARXIV.2502.06876. URL https://doi.org/10.48550/arXiv.2502.06876.

Jiuding Yang, Shengyao Lu, Hongxuan Liu, Shayan Shirahmad Gale Bagi, Zahra Fazel, Tomasz Czajkowski, and Di Niu. Perfcoder: Large language models for interpretable code performance optimization. *CoRR*, abs/2512.14018, 2025b. doi: 10.48550/ARXIV.2512.14018. URL https://doi.org/10.48550/arXiv.2512.14018.

Kailai Yang, Zhiwei Liu, Qianqian Xie, Jimin Huang, Tianlin Zhang, and Sophia Ananiadou. Metaaligner: Towards generalizable multi-objective alignment of language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/3d03800841fa1bb2f43ef1750aafcce4-Abstract-Conference.html.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. RLCD: reinforcement learning from contrastive distillation for LM alignment. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=v3XXtxWKi6.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024c. URL http://papers.nips.cc/paper_files/paper/2024/hash/71f7154547c748c8041505521ca433ab-Abstract-Conference.html.

Rui Yang, Xiaoman Pan, Feng Luo, Shuang Qiu, Han Zhong, Dong Yu, and Jianshu Chen. Rewards-in-context: Multi-objective alignment of foundation models with dynamic preference adjustment. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56276–56297. PMLR / OpenReview.net, 2024d. URL https://proceedings.mlr.press/v235/yang24q.html.

Sen Yang, Leyang Cui, Deng Cai, Xinting Huang, Shuming Shi, and Wai Lam. Not all preference pairs are created equal: A recipe for annotation-efficient iterative preference learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, volume EMNLP 2024 of *Findings of ACL*, pp. 6549–6561. Association for Computational Linguistics, 2024e. doi: 10.18653/V1/2024.FINDINGS-EMNLP.382. URL https://doi.org/10.18653/v1/2024.findings-emnlp.382.

Weiqing Yang, Hanbin Wang, Zhenghao Liu, Xinze Li, Yukun Yan, Shuo Wang, Yu Gu, Minghe Yu, Zhiyuan Liu, and Ge Yu. COAST: enhancing the code debugging ability of llms through communicative agent based data synthesis. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, volume NAACL 2025 of *Findings of ACL*, pp. 2570–2585. Association for Computational Linguistics, 2025c. doi: 10.18653/V1/2025.FINDINGS-NAACL.139. URL https://doi.org/10.18653/v1/2025.findings-naacl.139.

Tong Ye, Weigang Huang, Xuhong Zhang, Tengfei Ma, Peiyu Liu, Jianwei Yin, and Wenhai Wang. LLM4EFFI: leveraging large language models to enhance code efficiency and correctness. *CoRR*, abs/2502.18489, 2025. doi: 10.48550/ARXIV.2502.18489. URL https://doi.org/10.48550/arXiv.2502.18489.

Hao Yu, Bo Shen, Dezhi Ran, Jiaxin Zhang, Qi Zhang, Yuchi Ma, Guangtai Liang, Ying Li, Qianxiang Wang, and Tao Xie. Codereval: A benchmark of pragmatic code generation with generative pre-trained models. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*, pp. 37:1–37:12. ACM, 2024a. doi: 10.1145/3597503.3623316. URL https://doi.org/10.1145/3597503.3623316.

Huimu Yu, Xing Wu, Weidong Yin, Debing Zhang, and Songlin Hu. Codepmp: Scalable preference model pretraining for large language model reasoning. *CoRR*, abs/2410.02229, 2024b. doi: 10.48550/ARXIV.2410.02229. URL https://doi.org/10.48550/arXiv.2410.02229.

Zhuohao Yu, Jiali Zeng, Weizheng Gu, Yidong Wang, Jindong Wang, Fandong Meng, Jie Zhou, Yue Zhang, Shikun Zhang, and Wei Ye. Rewardanything: Generalizable principle-following reward models. *CoRR*, abs/2506.03637, 2025. doi: 10.48550/ARXIV.2506.03637. URL https://doi.org/10.48550/arXiv.2506.03637.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. ACECODER: acing coder RL via automated test-case synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 12023–12040. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.587/.

Alexander Zhang, Marcus Dong, Jiaheng Liu, Wei Zhang, Yejie Wang, Jian Yang, Ge Zhang, Tianyu Liu, Zhongyuan Peng, Yingshui Tan, Yuanxing Zhang, Zhexu Wang, Weixun Wang, Yancheng He, Ken Deng, Wangchunshu Zhou, Wenhao Huang, and Zhaoxiang Zhang. Codecriticbench: A holistic code critique benchmark for large language models. *CoRR*, abs/2502.16614, 2025a. doi: 10.48550/ARXIV.2502.16614. URL https://doi.org/10.48550/arXiv.2502.16614.

Beiqi Zhang, Peng Liang, Qiong Feng, Yujia Fu, and Zengyang Li. Copilot-in-the-loop: Fixing code smells in copilot-generated python code using copilot. In Vladimir Filkov, Baishakhi Ray, and Minghui Zhou (eds.), *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE 2024, Sacramento, CA, USA, October 27 - November 1, 2024*, pp. 2230–2234. ACM, 2024a. doi: 10.1145/3691620.3695290. URL https://doi.org/10.1145/3691620.3695290.

Boyu Zhang, Tianyu Du, Junkai Tong, Xuhong Zhang, Kingsum Chow, Sheng Cheng, Xun Wang, and Jianwei Yin. Seccoder: Towards generalizable and robust secure code generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 14557–14571. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.EMNLP-MAIN.806. URL https://doi.org/10.18653/v1/2024.emnlp-main.806.

Dutao Zhang, Sergey V. Kovalchuk, and YuLong He. Style2code: A style-controllable code generation framework with dual-modal contrastive representation learning. *CoRR*, abs/2505.19442, 2025b. doi: 10.48550/ARXIV.2505.19442. URL https://doi.org/10.48550/arXiv.2505.19442.

Junkai Zhang, Zihao Wang, Lin Gui, Swarnashree Mysore Sathyendra, Jaehwan Jeong, Victor Veitch, Wei Wang, Yunzhong He, Bing Liu, and Lifeng Jin. Chasing the tail: Effective rubric-based reward modeling for large language model post-training. *CoRR*, abs/2509.21500, 2025c. doi: 10.48550/ARXIV.2509.21500. URL https://doi.org/10.48550/arXiv.2509.21500.

Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. Self-edit: Fault-aware code editor for code generation. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 769–787. Association for Computational Linguistics, 2023a. doi: 10.18653/V1/2023.ACL-LONG.45. URL https://doi.org/10.18653/v1/2023.acl-long.45.

Kechi Zhang, Ge Li, Yihong Dong, Jingjing Xu, Jun Zhang, Jing Su, Yongfei Liu, and Zhi Jin. Codedpo: Aligning code models with self generated and verified source code. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pp. 15854–15871. Association for Computational Linguistics, 2025d. URL https://aclanthology.org/2025.acl-long.771/.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025e. URL https://openreview.net/forum?id=Ccwp4tFEtE.

Tianyi Zhang, Tao Yu, Tatsunori Hashimoto, Mike Lewis, Wen-Tau Yih, Daniel Fried, and Sida Wang. Coder reviewer reranking for code generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 41832–41846. PMLR, 2023b. URL https://proceedings.mlr.press/v202/zhang23av.html.

Zhiwei Zhang, Hui Liu, Xiaomin Li, Zhenwei Dai, Jingying Zeng, Fali Wang, Minhua Lin, Ramraj Chandradevan, Zheng Li, Chen Luo, Xianfeng Tang, Qi He, and Suhang Wang. Bradley-terry and multi-objective reward modeling are complementary. *CoRR*, abs/2507.07375, 2025f. doi: 10.48550/ARXIV.2507.07375. URL https://doi.org/10.48550/arXiv.2507.07375.

Yuwei Zhao, Ziyang Luo, Yuchen Tian, Hongzhan Lin, Weixiang Yan, Annan Li, and Jing Ma. Codejudge-eval: Can large language models be good judges in code understanding? In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 73–95. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.7/.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/65a39213d7d0e1eb5d192aa77e77eeb7-Abstract-Conference.html.

Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. Towards a unified multi-dimensional evaluator for text generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 2023–2038. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.131. URL https://doi.org/10.18653/v1/2022.emnlp-main.131.

Changzhi Zhou, Xinyu Zhang, Dandan Song, Xiancai Chen, Wanli Gu, Huipeng Ma, Yuhang Tian, Mengdi Zhang, and Linmei Hu. Refinecoder: Iterative improving of large language models via adaptive critique refinement for code generation. *CoRR*, abs/2502.09183, 2025a. doi: 10.48550/ARXIV.2502.09183. URL https://doi.org/10.48550/arXiv.2502.09183.

Enyu Zhou, Guodong Zheng, Binghai Wang, Zhiheng Xi, Shihan Dou, Rong Bao, Wei Shen, Limao Xiong, Jessica Fan, Yurong Mou, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. RMB: comprehensively benchmarking reward models in LLM alignment. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=kmgrlG9TR0.

Karen Zhou and Chenhao Tan. Autochecklist: Composable pipelines for checklist generation and scoring with llm-as-a-judge. 2026. URL https://api.semanticscholar.org/CorpusID:286371821.

Shang Zhou, Zihan Zheng, Kaiyuan Liu, Zeyu Shen, Zerui Cheng, Zexing Chen, Hansen He, Jianzhu Yao, Huanzhi Mao, Qiuyang Mang, Tianfu Fu, Beichen Li, Dongruixuan Li, Wenhao Chai, Zhuang Liu, Aleksandra Korolova, Peter Henderson, Natasha Jaques, Pramod Viswanath, Saining Xie, and Jingbo Shang. Autocode: Llms as problem setters for competitive programming. *CoRR*, abs/2510.12803, 2025c. doi: 10.48550/ARXIV.2510.12803. URL https://doi.org/10.48550/arXiv.2510.12803.

Shuyan Zhou, Uri Alon, Sumit Agarwal, and Graham Neubig. Codebertscore: Evaluating code generation with pretrained models of code. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of*

the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pp. 13921–13937. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.859. URL https://doi.org/10.18653/v1/2023.emnlp-main.859.

Yefan Zhou, Austin Xu, Yilun Zhou, Janvijay Singh, Jiang Gui, and Shafiq Joty. Variation in verification: Understanding verification dynamics in large language models. *CoRR*, abs/2509.17995, 2025d. doi: 10.48550/ARXIV.2509.17995. URL https://doi.org/10.48550/arXiv.2509.17995.

Zhanhui Zhou, Jie Liu, Jing Shao, Xiangyu Yue, Chao Yang, Wanli Ouyang, and Yu Qiao. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 10586–10613. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.630. URL https://doi.org/10.18653/v1/2024.findings-acl.630.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, Karthik Ganesan, Wei-Lin Chiang, Jian Zhang, and Jiantao Jiao. Starling-7b: Improving helpfulness and harmlessness with RLAIF. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=GqDntYTTbk.

Chenyang Zhu, Spencer Hong, Jingyu Wu, Kushal Chawla, Charlotte Tang, Youbing Yin, Nathan Wolfe, Erin Babinsky, and Daben Liu. RAFFLES: reasoning-based attribution of faults for LLM systems. *CoRR*, abs/2509.06822, 2025a. doi: 10.48550/ARXIV.2509.06822. URL https://doi.org/10.48550/arXiv.2509.06822.

Xiao Zhu, Xinyu Zhou, Boyu Zhu, Hanxu Hu, Mingzhe Du, Haotian Zhang, Huiming Wang, and Zhijiang Guo. Codescaler: Scaling code llm training and test-time inference via execution-free reward models. 2026. URL https://api.semanticscholar.org/CorpusID:285872163.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in LLM reasoning. *CoRR*, abs/2506.01347, 2025b. doi: 10.48550/ARXIV.2506.01347. URL https://doi.org/10.48550/arXiv.2506.01347.

Terry Yue Zhuo. Ice-score: Instructing large language models to evaluate code. In Yvette Graham and Matthew Purver (eds.), *Findings of the Association for Computational Linguistics: EACL 2024, St. Julian's, Malta, March 17-22, 2024*, volume EACL 2024 of *Findings of ACL*, pp. 2232–2242. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-eacl.148.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. Cyber-zero: Training cybersecurity agents without runtime. *CoRR*, abs/2508.00910, 2025. doi: 10.48550/ARXIV.2508.00910. URL https://doi.org/10.48550/arXiv.2508.00910.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. URL http://arxiv.org/abs/1909.08593.

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. *CoRR*, abs/2504.16084, 2025. doi: 10.48550/ARXIV.2504.16084. URL https://doi.org/10.48550/arXiv.2504.16084.

A Architecture And Training Details

Attribute	Themis-RM 0.6B	Themis-RM 1.7B	Themis-RM 4B	Themis-RM 8B	Themis-RM 14B	Themis-RM 32B
Architecture Attributes						
Backbone Architecture	🤗 Qwen/Qwen3-0.6B	🤗 Qwen/Qwen3-1.7B	🤗 Qwen/Qwen3-4B	🤗 Qwen/Qwen3-8B	🤗 Qwen/Qwen3-14B	🤗 Qwen/Qwen3-32B
Training Attributes: Preference Model Pre-Training (PT)						
Training Dataset	Themis-GeneralPreference					
Criteria Following	True					
Reward Centering Coefficient (μ)	0.01					
Behavior Cloning Coefficient (λ)	0.4					
Scheduler Type	Cosine					
Scheduler Warmup Proportion	0.05					
Optimizer Type	AdamW-Fused					
Peak Learning Rate	2e-5					
Terminal Learning Rate	1e-5					
Beta	{0.9, 0.95}					
Epsilon	1e-8					
Gradient Clipping	2.0					
Gradient Checkpointing	True					
Weight Decay	0.1					
FSDP Variant	Version 1					
FSDP Param Offload	True					
Flash Attention Variant	Version 2					
Flash Attention RMS Norm	True					
Flash Attention Fuse QKV	True					
Flash Attention Fuse MLP	True					
Liger Kernels	True					
Liger RoPE	True					
Liger RMS Norm	True					
Liger GLU Activation	True					
Model Datatype	bfloat16					
Softmax Datatype	float32					
AllReduce Datatype	float32					
Training Sequence Length	2560					
Global Batch Size	1024					
Training Epochs	2.0					
Training Attributes: Preference Modeling (PM)						
Training Dataset	Themis-CodePreference					
Criteria Following	True					
Reward Centering Coefficient (μ)	0.001					
Behavior Cloning Coefficient (λ)	0.25					
Scheduler Type	Cosine					
Scheduler Warmup Proportion	0.05					
Optimizer Type	AdamW-Fused					
Peak Learning Rate	1e-5					
Terminal Learning Rate	5e-7					
Beta	{0.9, 0.95}					
Epsilon	1e-8					
Gradient Clipping	1.5					
Gradient Checkpointing	True					
Weight Decay	0.1					
FSDP Variant	Version 1					
FSDP Param Offload	True					
Flash Attention Variant	Version 2					
Flash Attention RMS Norm	True					
Flash Attention Fuse QKV	True					
Flash Attention Fuse MLP	True					
Liger Kernels	True					
Liger RoPE	True					
Liger RMS Norm	True					
Liger GLU Activation	True					
Model Datatype	bfloat16					
Softmax Datatype	float32					
AllReduce Datatype	float32					
Training Sequence Length	4096					
Global Batch Size	512					
Training Epochs	1.0					

Table 5: Architectural heritage and training attributes of the Themis-RM suite of models across the preference model pre-training and preference modeling stages. The training objective is described in Section 4 and the training data composition for both stages is detailed in Appendix B.

B Data Collection

B.1 BigQuery SQL Query For Commit Mining

github/github-repos BigQuery Single-File Commit Mining

```
SELECT
  c.commit,
  c.subject,
  c.message,
  STRING_AGG(DISTINCT unnested_repo_name) AS repos,
  l.license,
  d.old_path AS old_file,
  d.new_path AS new_file,
  c.committer.time_sec AS unix_time
FROM
  `bigquery-public-data.github_repos.languages` AS lang_table,
  UNNEST(language) AS lang
JOIN `bigquery-public-data.github_repos.licenses` AS l
  ON l.repo_name = lang_table.repo_name
JOIN (
  SELECT *, unnested_repo_name
  FROM `bigquery-public-data.github_repos.commits`,
  UNNEST(repo_name) AS unnested_repo_name
) c
  ON c.unnested_repo_name = lang_table.repo_name,
  UNNEST(c.difference) AS d
WHERE
  l.license IN (
    'mit', 'artistic-2.0', 'isc', 'cc0-1.0', 'epl-1.0', 'mpl-2.0',
    'unlicense', 'apache-2.0', 'bsd-3-clause', 'agpl-3.0', 'lgpl-2.1', 'bsd-2-clause'
  )
  AND lang.name IN (
    'Python', 'Java', 'JavaScript', 'C', 'C#', 'C++', 'TypeScript', 'Go', 'Ruby',
  )
  AND LENGTH(c.message) > 10 AND LENGTH(c.message) < 15000
  AND LOWER(c.message) NOT IN (
    'update readme.md', 'initial commit', 'update', 'mirroring from micro.blog.',
    'update data.json', 'update data.js', 'add files via upload', 'update readme',
    "can't you see i'm updating the time?", 'dummy', 'update index.html', 'first commit',
    'create readme.md', 'heartbeat update', 'updated readme', 'update log', 'test',
    'no message', 'readme', 'wip', 'updates', 'commit', 'update _config.yaml', 'testing',
    'tweak', 'tweaks', 'modified', 'edited', 'yolo commit', 'yolo', 'made it work',
    'work in progress', 'fixing', 'for review', 'my changes', 'revised', 'addressed comments',
    'placeholder', 'test commit', 'trying something', 'experimental changes', 'hack',
    'do not merge', 'various updates', 'stuff'
  )
  AND LOWER(c.message) NOT LIKE '%pi push%' AND LOWER(c.message) NOT LIKE '%push pi%'
  AND LOWER(c.message) NOT LIKE 'merge%' AND d.old_path = d.new_path
  AND d.old_path IS NOT NULL AND d.new_path IS NOT NULL
GROUP BY
  c.commit, c.subject, c.message, l.license, d.old_path, d.new_path, c.committer.time_sec
HAVING COUNT(DISTINCT d.old_path) = 1
```

Listing 1: Streamlined GoogleSQL query for mining single-file-changing commits in openly licensed repositories, modified from the BigQuery GitHub dataset query pipeline presented in Muennighoff et al. (2024).

B.2 Search Terms For Aspect-Specific Commit Classifier Training

Criteria	Terms
Functional Correctness	adjust logic, align, boundary case, bug fix, class misus, corner case, correct algorithm, correct infinite, correct layout, correct logic, edge case, edgecase, enhancement, error handling, error recovery, faulty index, faulty logic, faulty loop, feature, fix broken, fix bug, fix cache, fix case, fix concurre, fix defect, fix do, fix duplicate, fix else, fix error, fix exception, fix fault, fix for, fix if, fix infinite, fix issue, fix iteration, fix logic, fix loop, fix null, fix problem, fix recursion, fix regres, fix switch, fix ui, fix ux, fix while, function misus, handle error, handle exception, handle failure, handle fault, handle null, handling error, handling exception, handling failure, illegal command, illegal comment, illegal condition, illegal declaration, illegal definition, illegal directive, illegal expression, illegal field, illegal indentation, illegal index, illegal key, illegal offset, illegal variable, input handling, integer overflow, library misus, logic error, missing command, missing comment, missing condition, missing declaration, missing definition, repair broken, repair bug, repair error, repair issue, resolve logic, self assign, self-assign, unclosed brace, unclosed bracket, unclosed comment, unclosed parenthesis, unclosed string, undefined class, undefined constant, undefined field, undefined function, undefined index, undefined key, undefined member, undefined method, undefined object, undefined offset, undefined property, undefined value, undefined variable, uninitialized class, uninitialized constant, uninitialized field, uninitialized function, uninitialized index, uninitialized key, uninitialized member, uninitialized method, uninitialized object, uninitialized offset, uninitialized property, uninitialized value, uninitialized variable, unmatched brace, unmatched bracket, wrong logic, wrong loop, wrong variable, wrongly assign
Execution Efficiency	add async, asynchronous, avoid unnecessary computation, batch api, batch operation, batch proc, boost efficiency, boost performance, cache frequently, cache function, cache results, cache the, cache values, cache variables, cache with memoization, caching, code optimization, cold path, concurrent, constant time, constant-time, continuous batch, decrease runtime, dynamic batch, early termination, efficiency improvement, efficient access, efficient algorithm, efficient code, efficient execution, efficient implementation, efficient iteration, efficient processing, efficient structures, enhance bit manipulation, enhance bit operations, enhance bit twiddling, enhance bitwise, enhance efficiency, enhance latency, enhance processing speed, enhance sort, faster, fix latency, fix stalling, fix startup time, fix time complexity, hash, hot path, implement async operations, implement connection pooling, implement efficient data serialization, implement lazy evaluation, implement lazy loading, implement multithreading, implement parallel processing, implement producer-consumer pattern, branch prediction, improve cache coherency, cache hit, cache locality, cache performance, improve cache utilization, improve speed, inplace, latency sensitive, load data in chunks, lru, make code more efficient, make code more performant, make code more responsive, make code run quicker, memoize, memory map, o(1) lookup, o(nlogn) sorting algorithm, optimize data, optimize file, optimize i/o, optimize import, optimize initialization, optimize latency, optimize list, optimize lookahead, optimize lookup, optimize loop, optimize memoization, optimize merge step, optimize network requests, optimize numpy operations, optimize path cost calculation, optimize pathfinding, optimize performance, overhead reduc, oversynchroni, parallelize, perf improvement, performance improvement, precompute expensive operations, precompute frequently used results, precompute frequently used values, precompute lookup tables, precompute results, precompute values, prune unnecessary, quickness improvement, reduce algorithm complexity, reduce computational overhead, reduce function call, reduce i/o operations, reduce lookup time, reduce overhead, reduce recursion, reduce recursive call, reduce runtime, time complexity, remove overhead, remove slow, replace slow, reduce latency, speed fix, speed up, speedup, tail call optimization, unroll loops, use bisect for binary search, use built-in, use builtin, use in-place, use list comprehension for speed, use map, use numpy vectorization, use optimized, use set, utilize multiprocessing, vectorize, work steal
Memory Efficiency	fix memory management issue, fix memory, garbage collection, high memory, improve memory allocation, improve memory bound, improve memory usage, improve memory management, lazy eval, lazy load, memory bloat, memory consumption, memory efficien, memory footprint, memory leak fix, memory leak, memory map, memory optimization, memory pool, memory wastage, memory-efficien, resource leak, unnecessary malloc, unnecessary memory alloc, use less memory, zero copy
Readability And Maintainability	abstract, anti-pattern, antipattern, appropriate nam, big class, big method, break down, break up, builder pattern, chain of respons, circular dependency, clarify, clarity, clean up, cleanup, code formatting, code layout, code path, code quality, code smell, code style, coding standards, coding style, cohesion, cohesive, command pattern, complex condition, consistent nam, control coupling, control dependency, control flow, convention, cyclomatic, data clump, data coupling, data dependency, dead assignment, dead class, dead code, dead function, dead method, dead parameter, dead variable, decouple, deep nest, deeply nest, dependency inversion, descriptive nam, dry, duplication, halstead, inconsistent doc, inconsistent format, interface segregation, intimacy violation, large class, large method, large parameter list, liskov substitution, long class, long method, long parameter list, maintainability, maintainable, message chain, modernize, modular, normalize, omitted default, omitted else, omitted error, omitted exception, omitted if, omitted switch, open/closed, parallel inheritance, pep 8, pep8, portability, primitive obsession, readability, readable, reduce complexity, reduce coupling, reduce dependency, reduce depth, reduce nest, refactor, refused bequest, remove goto, remove hardcod, separate, separation, shotgun surgery, simplify, single responsibility, solid, split function, style guide, tech debt, technical debt, temporary field, test coverage, testability, testable, unconditional branch, unconditional control, unconditional jump, unconditional statement, unused assignment, unused class, unused code, unused function, unused method, unused parameter, unused variable, visitor pattern
Security Hardness	access control, android injection, auth bypass, auth forgery, authorization bypass, avoid directory traversal, avoid dos attack, avoid file inclusion, avoid format string vulnerability, avoid overflow, avoid vuln, avoid weak encryption, avoid weak hashing, clear text pass, clear text secret, clear text sess, clear-text cookie, clear-text cred, clear-text pass, clear-text secret, clear-text sess, cmd injection, code injection, collision attack, collision resist, command injection, critical patch, critical vuln, cross path injection, cross site scripting, cross-path injection, cross-site scripting, csrf, cve, cwe, data forgery, data validation, denial of service, deprecated hash, deprecate cipher, deprecate encryption, deprecate vuln, deprecated crypto, dns prefetch, double dealloc, double free, double-free, dynamic rege, dynamic require, enhance safety, eval injection, exploit, expression injection, file injection, fix deseriali, fix entropy, fix hash, fix key entropy, fix key length, fix key randomness, fix key size, fix key strength, fix overflow, fix randomness, fix safety, fix sensitive, fix secret, fix seriali, fix snyk, fix vuln, fix weak cipher, fix weak encryption, fix weak hash, fixed entropy, fixed nonce, fixed prng, fixed random, fixed seed, fixed-random, fortify, fragment injection, hard-coded secret, hard-coded key, hard-coded password, hard-coded token, hardcoded key, hardcoded password, hardcoded secret, hardcoded token, harden, header injection, heap corrupt, heap overflow, http response split, imap injection, improper authentication, improper authorization, improper validation, improve safety, input validation, insufficient entropy, ldap injection, log injection, meet-in-the-middle attack, mem align, memory abuse, memory alignment, memory corrup, memory misuse, module injection, non-constant rege, non-constant require, nvd, open redirect, origin validat, periodic random, permission bypass, permission escalation, permission exploit, permission injection, permission override, permission poisoning, plaintext cookie, plaintext cred, plaintext pass, plaintext secret, plaintext sess, pointer abuse, pointer misma, pointer misuse, pre-image attack, preimage attack, preimage resist, privilege elevation, privilege escalation, privilege exploit, random byte, random-byterce, remove deprecated encryption, remove deprecated hash, remove overflow, remove secret, security, sess fix, sess hijack, sess poison, sess repla, session fixation, session hijacking, session poisoning, session replay, shell injection, signal error, signal exploit, signal injection, smtp injection, sql injection, sqli, timing attack, type forgery, type injection, type validation, uaf, unauthori, unsafe, unsalted hash, untrusted content, untrusted data, untrusted deseriali, untrusted host, untrusted input, untrusted origin, untrusted source, untrusted user, unverified data, unverified host, unverified input, unverified origin, unverified source, unverified user, update hash, upgrade hash, use after dealloc, use after free, use-after-free, validate auth, variable require, vuln fix, vuln patch, vulnerability fix, vulnerability patch, xml external entit, xml injection, xpath injection, xquery injection, xss, xxe, zero day, zero-day

Table 6: Per-criteria search terms used to recall positives for training the commit classifier. Refer to Section 3 for a description of the complete commit preference mining pipeline.

B.3 Themis-RM Training Data Mixture And Filtering

The Themis-GeneralPreference collection: We curate Themis-GeneralPreference from a mixture of pre-existing preference datasets that pertain to LM helpfulness and harmlessness in natural language. We also repurpose a small number of relevance preferences from code retrieval datasets. Designed to teach RMs the nuances of general human preferences, this 110k+ sample dataset is composed as follows:

1. **CodeR-Pile** ■ nebula2025/CodeR-Pile ■ 41924 samples

   We create preference pairs from the CodeR-Pile (Li et al., 2025a) code retrieval dataset. Specifically, we leverage the code augmentation, exemplar, integration, refinement, simplification, pseudo-code, tutorial, and web query subsets of the dataset. We curate the chosen response from code parsed in the positive document. Similarly, we parse the rejected response from a document chosen via Zipfian sampling over the top-10 ranked list of mined hard negatives.

   Pref. Criteria: Helpfulness
   Response Lang: C C# C++ Go Java JS Python Ruby

2. **Skywork-Preference** ■ Skywork/Skywork-Reward-Preference-80K-v0.2 ■ 30146 samples

   We source preference pairs from the Skywork Preference (Liu et al., 2024a) dataset. Specifically, we select and further filter the WildGuard (Han et al., 2024), OffsetBias (Park et al., 2024), Magpie (Xu et al., 2025c), and HelpSteer2 (Wang et al., 2024f) subsets.

   Pref. Criteria: Helpfulness Harmlessness
   Response Lang: Natural Language

3. **Tulu-IF** ■ allenai/tulu-3-pref-personas-instruction-following ■ 13705 samples

   We source response pairs that follow and violate a pre-specified set of instructions present in the Tulu v3 (Lambert et al., 2024) instruction-tuning dataset.

   Pref. Criteria: Helpfulness
   Response Lang: Natural Language

4. **H4-Stackexchange** ■ HuggingFaceH4/stack-exchange-preferences ■ 12139 samples

   We follow Askell et al. (2021) and filter StackExchange technical forums, selecting highly-voted accepted answers as the chosen response and well-formed but low-scoring answers as the rejected ones.

   Pref. Criteria: Helpfulness
   Response Lang: Natural Language

5. **Arena-HumanPreference** ■ lmarena-ai/arena-human-preference-140k ■ 7068 samples

   Human voting validated preferences from filtered data dumps of model head-to-head contests on LMArena (Chiang et al., 2024).

   Pref. Criteria: Helpfulness
   Response Lang: Natural Language

6. **Prometheus-Preference** ■ prometheus-eval/Preference-Collection ■ 2864 samples

   We mine response pairs from instruction clusters corresponding to helpfulness and harmlessness preferences from the Prometheus Preference (Kim et al., 2024a) data.

   Pref. Criteria: Helpfulness Harmlessness
   Response Lang: Natural Language

7. **HelpSteer3** ■ nvidia/HelpSteer3 ■ 1452 samples

   High-margin preference pairs from the HelpSteer3 Preference (Wang et al., 2025f) data. Specifically, we incorporate the general and stem subsets.

   Pref. Criteria: Helpfulness
   Response Lang: Natural Language

8. **Argilla-DPO** ■ argilla/distilabel-math-preference-dpo ■ 1042 samples

   High-margin math answer preference pairs from the Argilla Preference data. We synthetically filter for pairs in which both the chosen and rejected responses converge on the same answer, thereby selecting for stylistic preferences that capture aspects beyond correctness.

   Pref. Criteria: Helpfulness
   Response Lang: Natural Language

9. **Truthy-DPO** ■ jondurbin/truthy-dpo-v0.1 ■ 377 samples

   Synthetically labeled truthfulness preferences mined from existing instruction-tuning data.

   Pref. Criteria: Helpfulness
   Response Lang: Natural Language

The Themis-CodePreference collection: We primarily curate Themis-CodePreference from code preference datasets we construct. Our collection centers on sourcing diverse preference scenarios from GitHub commits and from synthetically bugged instruction-tuning data. We further augment our data mixture with training sets from pre-existing code-preference and retrieval datasets. Primarily designed to teach RMs the nuances of scoring code along functional and non-functional axes, this 350k+ sample dataset is composed as follows:

1. **Commit-Preference** ■ 126586 samples

   We source code preferences from non-reverted single-file GitHub commits verified to be part of subsequent successfully-merged pull requests. Preference strength is validated via consensus between multiple frontier LMs, and pairs are selected for the presence of a single intent (e.g., improving runtime efficiency). A detailed description of the commit preference collection is outlined in Section 3 and Appendices B and C.

   Pref. Criteria: Correctness, Runtime, Memory, Readability, Security
   Response Lang: C, C#, C++, Go, Java, JS, Python, Ruby

2. **CodeR-Pile** ■ nebula2025/CodeR-Pile ■ 77153 samples

   We create preference pairs from the CodeR-Pile (Li et al., 2025a) code retrieval dataset. Specifically, we leverage the code augmentation, exemplar, integration, refinement, simplification, pseudo-code, tutorial, and web query subsets of the dataset. We curate the chosen response from code parsed in the positive document. Similarly, we parse the rejected response from a document chosen via Zipfian sampling over the top-10 ranked list of mined hard negatives.

   Pref. Criteria: Correctness
   Response Lang: C, C#, C++, Go, Java, JS, Python, Ruby

3. **Bugged-Instruct** ■ 54969 samples

   We repurpose instruction-tuning datasets by creating rejected responses via model-based introduction of algorithmic or syntactic bugs. We incorporate data from a mix of diverse extant datasets, including OSS-Instruct (Wei et al., 2024b), Inverse-Instruct (Wu et al., 2025c), McEval-Instruct (Chai et al., 2025), and Package-Instruct (Huang et al., 2025b). The prompt used is outlined in Section C.4.

   Pref. Criteria: Correctness
   Response Lang: C, C#, C++, Go, Java, JS, Python, Ruby

4. **ProSec** ■ prosecalign/prosec-mixed-clm7b-inst ■ 37690 samples

   We procure synthetically generated pairs of vulnerability-fixes from ProSec (Xu et al., 2025a).

   Pref. Criteria: Security
   Response Lang: C, C++, Java, JS, Python

5. **Venus** ■ Elfsong/Venus ■ 21784 samples

   We source runtime and memory usage code preferences from Venus (Du et al., 2025), selecting samples where the chosen completion is at least 5x as efficient as the rejected one.

   Pref. Criteria: Runtime, Memory
   Response Lang: Python

6. **CodeNet** ■ iNeil77/CodeNet ■ 16819 samples

   We source runtime and memory usage code preferences per submitter from CodeNet (Puri et al., 2021), selecting samples where the chosen completion is at least 5x as efficient as the rejected one.

   Pref. Criteria: ~~Correctness~~ Runtime Memory ~~Readability~~ ~~Security~~ ~~Helpfulness~~ ~~Harmlessness~~
   Response Lang: C C# C++ Go Java JS Python Ruby ~~Natural Language~~

7. **RunBugRun** ■ ASSERT-KTH/RunBugRun-Final ■ 6931 samples

   We procure code contest program-repair pairs from the train set of RunBugRun (Prenner & Robbes, 2023).

   Pref. Criteria: Correctness ~~Runtime~~ ~~Memory~~ ~~Readability~~ ~~Security~~ ~~Helpfulness~~ ~~Harmlessness~~
   Response Lang: ~~C~~ ~~C#~~ ~~C++~~ ~~Go~~ ~~Java~~ ~~JS~~ Python ~~Ruby~~ ~~Natural Language~~

8. **ECCO** ■ CodeEff/ECCO ■ 6188 samples

   We source code contest runtime preferences from the train set of ECCO (Waghjale et al., 2024).

   Pref. Criteria: ~~Correctness~~ Runtime ~~Memory~~ ~~Readability~~ ~~Security~~ ~~Helpfulness~~ ~~Harmlessness~~
   Response Lang: ~~C~~ ~~C#~~ ~~C++~~ ~~Go~~ ~~Java~~ ~~JS~~ Python ~~Ruby~~ ~~Natural Language~~

9. **CodeScaleR** ■ LARK-Lab/CodeScalerPair-51K ■ 2896 samples

   Model-generated correctness preference pairs procured from CodeScaleR (Zhu et al., 2026). We obtain solutions to coding problems with a focus on difficult preference pairs where the rejected sample passes most but not all test-cases.

   Pref. Criteria: Correctness ~~Runtime~~ ~~Memory~~ ~~Readability~~ ~~Security~~ ~~Helpfulness~~ ~~Harmlessness~~
   Response Lang: ~~C~~ ~~C#~~ ~~C++~~ ~~Go~~ ~~Java~~ ~~JS~~ Python ~~Ruby~~ ~~Natural Language~~

10. **Pie4Perf** ■ iNeil77/pie4perf-Train ■ 1640 samples

    We source execution runtime preferences from the Pie4Perf (Shypula et al., 2024) dataset. Specifically, we use the HQ-Train subset, which contains preference pairs derived from successful submissions by non-spam authors with highly divergent runtimes.

    Pref. Criteria: ~~Correctness~~ Runtime ~~Memory~~ ~~Readability~~ ~~Security~~ ~~Helpfulness~~ ~~Harmlessness~~
    Response Lang: ~~C~~ ~~C#~~ ~~C++~~ ~~Go~~ ~~Java~~ ~~JS~~ ~~Python~~ ~~Ruby~~ ~~Natural Language~~

11. **Cybernative-DPO** ■ CyberNative/Code_Vulnerability_Security_DPO ■ 1354 samples

    We source security-fix preferences created via synthetic fixes to vulnerable code.

    Pref. Criteria: ~~Correctness~~ ~~Runtime~~ ~~Memory~~ ~~Readability~~ Security ~~Helpfulness~~ ~~Harmlessness~~
    Response Lang: ~~C~~ C# C++ Go Java JS Python Ruby ~~Natural Language~~

The Themis-GeneralPreference and Themis-CodePreference filtering procedure: We thoroughly clean and decontaminate our training data for the preference model pre-training (PT) and the preference modeling (PM) phases via the following steps:

1. We first ensure that all samples in Themis-GeneralPreference and Themis-CodePreference are no longer than 2560 and 4096 tokens, respectively.[7] Subsequently, we filter out samples with trivial code responses whose syntax tree is shallower than 3 levels deep. Additionally, we ensure that all GitHub commit preference data we train on is sourced no later than March 2019.

2. We next leverage the GlotLID (Kargaran et al., 2023) language classifier to discard samples with non-English prompts, followed by filtering out samples with prompt perplexities greater than 1200, as measured by a KenLM (Heafield, 2011) model trained on the OSCAR EN corpus (Abadji et al., 2022).

3. Next, we run a dataset-level (i.e., Themis-GeneralPreference and Themis-CodePreference separately) near-deduplication step using a MinHash (Broder, 1997) filter with a shingle size of 20 and a similarity threshold of 0.75. Finally, following prior work (Brown et al., 2020; Elazar et al., 2024), we decontaminate our training data by removing any sample whose prompt registers a 13-gram overlap with a prompt in Themis-CodeRewardBench, RewardBench V1 (Lambert et al., 2025), RewardBench V2 (Malik et al., 2025), JudgeBench (Tan et al., 2025) or RM-Bench (Liu et al., 2025f).

[7] As measured by the Themis-RM tokenizer.

C Data Acquisition And Filtering Prompts

C.1 Commit Saliency Rubric

System Prompt

You are an experienced Principal Software Engineer asked to critically review the code changes by a colleague tasked with improving the production readiness of a critical service's codebase.

User Prompt

Code Change Review: {{criteria}}

This specific change edits a single source file written in {{programming_language}}. The faithfulness of the change description to the actual code changes is not guaranteed and should be verified by you via careful examination of the code.

[OLD_CODE]{{old_file_contents}}[/OLD_CODE]

[NEW_CODE]{{new_file_contents}}[/NEW_CODE]

[CHANGES]{{commit_message}}[/CHANGES]

Provide a factual summary of the specific functional and technical changes made by your colleague. You are free to use the provided file contents to support your summary, but ensure that you do not copy-paste your colleague's change description verbatim. This summary must be enclosed within [SUMMARY] and [/SUMMARY]. Subsequently, score the specific functional and technical changes made by your colleague and score the quality of the code changes on a scale of 1 to 5 (inclusive) based on whether they improve the code along the axis under consideration and whether the changes are specific to {{criteria}} within [RATING] and [/RATING] tags. The scoring rubric follows:

1. The change doesn't improve the code's {{criteria}} or degrades it overall. The change may or may not also contain unrelated edits that are not specific to {{criteria}}. The change may also introduce other issues or bugs unrelated to {{criteria}}.

2. The code change is unnecessary and does not have any discernible effect on the code's {{criteria}}, but does not degrade its {{criteria}} either. The change may or may not also contain unrelated edits that are not specific to {{criteria}}.

3. The code change makes the code slightly better with respect to {{criteria}} but largely leaves it the same. The change might also contain unnecessary edits unrelated to {{criteria}}, but the majority of the changes are specific to {{criteria}}.

4. The code change makes the code significantly better with respect to {{criteria}}. Sporadic edits unrelated to {{criteria}} may exist, but the majority of the changes are specific to {{criteria}}. The change does not introduce any new issues or bugs unrelated to {{criteria}}.

5. The code change greatly improves the code's {{criteria}}, making it a must-have feature or addition. The change is also well implemented and specific, i.e., not a generic suggestion that could apply to any codebase. The incidence of unnecessary edits that are unrelated to {{criteria}} is minimal or non-existent in the change. The change does not introduce any new issues or bugs unrelated to {{criteria}}.

Listing 2: The scoring rubric used to evaluate the criteria-level preference strength present in single-file GitHub commits. Refer to Sections 3 and 4 for more details on commit preference mining.

C.2 Commit Inverse Instruction Generation

System Prompt

You are a veteran problem-setter for a popular programming contest and coding interview preparation platform. You are adept at crafting problems of varying situational backgrounds, difficulty levels, and styles.

User Prompt

The {{programming_language}} and {{content_style}} can be one of the following tuples

- Graduate Course Assignment: Problems that resemble assignments given in a graduate-level computer science course in their level of detail, complexity, and structure.
- Quora Question: Problems that are reflective of the kind of questions asked on the popular question-and-answer platform Quora in their style.
- Stackoverflow Question: Problems that are reflective of the kind of questions asked on the popular question-and-answer platform Stackoverflow in their level of detail, complexity, structure, and style.
- Google Search Query: Problems that are reflective of the kind of queries entered into the popular search engine Google in their level of detail, complexity, structure, and style.
- Programming Contest Problem: Problems that are reflective of the kind of problems set in a programming contest in their level of detail, structure, and style.

You are given two snippets in {{programming_language}} that answer a {{problem_style}} (). These are specified between the [EXAMPLE1] and [\EXAMPLE1] and the [EXAMPLE2] and [\EXAMPLE2] tags below:

[EXAMPLE1]{{old_file_contents}}[/EXAMPLE1]

[EXAMPLE2]{{new_file_contents}}[/EXAMPLE2]

While we have the resulting solution code snippets, your first order of business is to inspect the reference solutions and detail what they accomplish in full. The single description must faithfully outline both the provided code snippets and their intended functionality. This description must be enclosed within [DESCRIPTION] and [/DESCRIPTION]

Secondly, you must craft a clear and concise problem statement that fulfils the following criteria:

1. Is consistent in meaning with the provided reference solutions.

2. Would sufficiently identifiably lead a mid-tier to experienced developer to plausibly converge on either of the reference solutions (EXAMPLE1 or EXAMPLE2) with equal likelihood.

3. Resembles a {{content_style}} in its level of detail, complexity, structure, and style.

4. Is free of any direct or indirect references to the reference solutions or the specific code constructs used in them, and does not copy the description verbatim.

Provide the instruction you craft between [INSTRUCTION] and [\INSTRUCTION] tags.

Listing 3: The inverse instruction creation prompts for crafting realistic queries for code change pairs mined from GitHub. Refer to Sections 3 and 4 for more details on commit preference mining.

C.3 Themis-RM Training And Evaluation System Prompt

Themis-RM Evaluation Principles

You are an experienced AI judge and evaluator, specifically designed and trained to assess the overall quality of responses to human prompts. The prompts can be of various types, including but not limited to open-ended questions, coding tasks, creative writing, and requests for information. The responses may be generated by a language model or a human and may include text , code, or both. You will receive the prompt and the generated response for each judgement session. You must score the response based on a pre-specified set of criteria.

The following criteria must govern your scoring for the current judgment session:

General Principles
1. **Helpfulness**: How well does the response address the prompt? Does it provide a complete and thorough answer, or is it lacking in essential details? This tenet applies to evaluating all types of responses. Helpful responses must score higher than unhelpful ones.

2. **Harmlessness**: Does the response contain any harmful or inappropriate content? This includes hate speech, violence, self-harm, or other harmful content. This tenet applies to evaluating all types of responses. You must score harmless responses higher than harmful ones.

Criteria-Specific Code-Only Principles
3. **Memory Efficiency**: Does the response follow best practices for memory efficiency? Examples include using efficient data structures, minimizing memory usage, avoiding memory leaks, and effectively pooling/managing resources, among others. This tenet applies to evaluating code responses. You must score more memory-efficient responses higher than less memory-efficient ones.

4. **Functional Correctness**: Does the response follow best practices for functional correctness? Examples include algorithmic correctness, specifications adherence, and edge-case handling, among others. This tenet applies to evaluating code responses. You must score more functionally correct responses higher than less functionally correct ones.

5. **Readability and Maintainability**: Does the response follow best practices for readability and maintainability? Examples include using clear, descriptive names, following consistent formatting and style guidelines, modularizing code, and providing comments and documentation where necessary. This tenet applies to evaluating code responses. You must score more readable and maintainable responses higher than less readable and maintainable ones.

6. **Runtime Efficiency**: Does the response follow best practices for runtime efficiency? Examples include using efficient algorithms and data structures, minimizing time complexity, avoiding unnecessary computations, caching results, and leveraging parallel processing or asynchronous programming techniques where appropriate. This tenet applies to evaluating code responses. You must score more runtime-efficient responses higher than less runtime-efficient ones.

7. **Security Hardness**: Does the response follow best practices for security hardness? Examples include input validation, output encoding, proper error handling, and secure coding practices. This tenet applies to evaluating code responses. You must score more secure, less vulnerable responses higher than less secure, more vulnerable ones.

Listing 4: The list of evaluation principles used for training and evaluating `Themis-RM`. Refer to Section 5.2 for analysis on how specifying scoring criteria improves RMs.

C.4 Synthetic Bug-Laden Solution Generation

System Prompt

```
You are an experienced {{programming_language}} developer who also works part-time at a contest
- and interview-preparation platform. You incorporate your knowledge of common coding patterns
and best practices to suggest challenging coding problems that require a deep understanding of
algorithms and data structures.
```

User Prompt

```
You are tasked with creating a coding problem for a contest in {{programming_language}}. In the
 interest of raising the quality of the problems, you decide to make the problems involve an
open-ended question (PROBLEM) with a buggy code snippet (BUGGY_CODE). The contestants will have
 to fix the bug and match the reference solution to answer the question successfully. To scale
this problem-setting process, you decide to base your problems on pre-existing validated (
PROBLEM, REFERENCE_SOLUTION) pairs and generate the buggy code (BUGGY_CODE) yourself. The buggy
 code must follow the following constraints:

1. It must be a modification of the reference solution that introduces only functional, logical
, and algorithmic bugs.

2. The introduction of small syntax and grammatical errors is also allowed. However, you must
try to maintain the code's surface-level structure as much as possible.

3. The buggy code must not allude to the original problem or the reference solution in any way.
 The problem statement and the reference solution are provided to you as part of the input, but
 you must not use them in your output.

4. The buggy code must not allude to the introduced bugs in any way. Variables, functions,
classes, and other identifiers should not be named in a way that suggests the presence of bugs.
 Similarly, the comments and documentation should not hint at the bugs.

5. The addition of new features or the removal of existing ones is out of scope for this task.

6. The introduction of security vulnerabilities, memory leaks, or other non-functional bugs is
out of scope for this task.

Below is a validated (PROBLEM, REFERENCE_SOLUTION) pair that you can use to generate the buggy
code snippet. The problem is enclosed between the tags [PROBLEM] and [\PROBLEM]. The reference
solution is enclosed between the tags [REFERENCE_SOLUTION] and [\REFERENCE_SOLUTION].

[PROBLEM]{{problem_description}}[/PROBLEM]

[REFERENCE_SOLUTION]{{solution_code}}[/REFERENCE_SOLUTION]

Firstly, develop a buggy code snippet (BUGGY_CODE) that modifies the reference solution. The
buggy code must follow the constraints mentioned above and be enclosed between the [BUGGY_CODE]
 and [\BUGGY_CODE] tags. Secondly, outline and explain the bugs that you have introduced in the
 buggy code snippet. The explanation must be enclosed between the tags [BUG_EXPLANATION] and [\
BUG_EXPLANATION].
```

Listing 5: The prompt for generating buggy solutions conditioned on pre-existing valid solutions. Such samples enable the creation of diverse pairs of functional correctness preferences, which we use to enhance the robustness of `Themis-RM` in Section 4.

D Detailed Results

D.1 Functional Correctness (FC) Dataset-Level Accuracy

Auxiliary Training Objectives | Code RM | Criteria-Following RM | Math RM | Generative RM | Reasoning RM

	Model	Size	Functional Correctness (FC)					
			HEPack	MBPP+Fix (Hard)	MDEval	DebugEval	RunBugRun	CommitPref
XL	Qwen/Qwen2.5-Math-RM-72B	72B	98.41	81.08	83.58	83.70	79.30	63.52
	nvidia/AceMath-72B-RM	72B	99.20	70.27	94.78	96.55	86.26	58.06
	Qwen/WorldPM-72B-RLHFLow	72B	98.73	54.05	93.28	96.27	85.39	69.45
	ContextualAI/LMUnit-qwen2.5-72b	72B	84.24	40.54	63.43	76.38	24.28	11.64
	nvidia/Llama-3.3-Nemotron-70B-Reward	70B	98.57	56.76	94.03	97.93	88.11	75.76
	infly/INF-ORM-Llama3.1-70B	70B	99.20	51.35	97.01	95.30	80.58	65.45
	allenai/Llama-3.1-70B-Instruct-RM-RB2	70B	99.20	43.34	93.28	96.55	84.81	72.24
	allenai/Llama-3.1-Tulu-3-70B-SFT-RM-RB2	70B	99.04	45.95	95.52	96.96	84.12	73.58
	Nexusflow/Athene-RM-70B	70B	98.89	56.76	95.52	95.99	85.02	79.88
	Nexusflow/Starling-RM-34B	34B	96.02	21.62	91.04	93.78	73.21	63.52
	Themis-RM 32B	32B	99.84	89.19	95.52	98.34	91.98	93.21
	TIGER-Lab/AceCodeRM-32B	32B	97.93	64.86	90.30	95.03	70.62	43.15
	nvidia/Qwen3-Nemotron-32B-GenRM-Principle	32B	96.34	78.38	81.34	94.75	77.98	56.61
L	nicolinho/QRM-Gemma-2-27B	27B	64.01	56.76	60.25	61.19	50.53	48.48
	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1	27B	97.45	45.95	92.54	96.96	84.16	63.88
	Skywork/Skywork-Reward-Gemma-2-27B-v0.2	27B	68.31	78.38	65.67	65.47	52.59	50.06
	internlm/internlm2-20b-reward	20B	97.93	35.14	90.30	94.47	78.40	66.75
	Themis-RM 14B	14B	98.57	72.97	94.03	97.24	90.70	91.03
	rubricreward/R3-Qwen3-14B-14k	14B	87.10	56.76	66.42	73.48	53.37	27.84
	openbmb/UltraRM-13b	13B	88.85	16.22	91.79	83.29	67.94	73.09
M	**Themis-RM 8B**	8B	98.57	62.16	94.03	97.24	89.51	89.33
	LARK-Lab/CodeScaler-8B	8B	99.04	43.24	94.78	97.51	85.60	75.88
	rubricreward/R3-Qwen3-8B-14k	8B	82.80	43.24	56.72	72.79	48.15	26.55
	Skywork/Skywork-Reward-V2-Qwen3-8B	8B	98.89	37.84	94.78	97.51	84.86	77.45
	RLHFlow/ArmoRM-Llama3-8B-v0.1	8B	97.61	45.95	85.07	93.92	76.83	62.06
	nicolinho/QRM-Llama3.1-8B-v2	8B	95.86	45.95	94.78	94.34	76.50	65.70
	allenai/Llama-3.1-8B-Base-RM-RB2	8B	97.77	32.43	97.01	94.89	78.68	71.52
	Ray2333/GRM-llama3-8B-sftreg	8B	96.97	24.32	83.58	93.78	75.23	65.45
	LxzGordon/URM-LLaMa-3.1-8B	8B	95.86	48.65	94.78	95.30	75.84	65.21
	NCSOFT/Llama-3-OffsetBias-RM-8B	8B	96.82	40.54	91.79	92.68	76.54	64.97
	sfairXC/FsfairX-LLaMA3-RM-v0.1	8B	97.45	32.43	91.04	94.34	77.12	68.85
	Nexusflow/Athene-RM-8B	8B	94.90	37.84	95.52	91.71	78.52	74.18
	TIGER-Lab/AceCodeRM-7B	7B	96.18	59.46	91.79	95.86	80.70	65.09
	eth-dl-rewards/internlm2-7b-reward-code-100k	7B	96.66	21.62	85.82	93.92	75.06	65.09
	eth-dl-rewards/internlm2-7b-reward-math-100k	7B	95.22	29.73	85.07	89.50	74.77	63.88
	reciprocate/mistral-7b-gsm8k-code-rm	7B	88.22	29.73	85.82	88.26	66.91	58.18
	nvidia/AceMath-7B-RM	7B	95.70	64.86	82.09	93.37	75.14	55.88
	openbmb/Eurus-RM-7b	7B	89.97	24.32	74.63	91.44	69.14	63.52
	internlm/internlm2-7b-reward	7B	94.59	32.43	87.21	92.13	76.09	66.67
S	**Themis-RM 4B**	4B	98.29	67.57	91.04	94.20	86.54	89.45
	LARK-Lab/CodeScaler-4B	4B	98.09	48.65	95.52	97.38	83.05	73.21
	PKU-ONELab/CE-RM-4B	4B	91.08	51.35	75.37	86.74	61.69	44.24
	rubricreward/R3-Qwen3-4B-14k	4B	81.21	54.05	59.70	68.78	45.76	23.52
	Skywork/Skywork-Reward-V2-Qwen3-4B	4B	98.73	45.95	96.27	97.79	83.54	75.64
	Ray2333/GRM-llama3.2-3B-sftreg	3B	94.90	35.14	93.28	93.09	74.86	62.30
XS	internlm/internlm2-1_8b-reward	1.8B	81.69	21.62	79.10	72.38	62.80	63.88
	Themis-RM 1.7B	1.7B	93.47	43.24	90.30	88.81	78.85	81.70
	LARK-Lab/CodeScaler-1.7B	1.7B	92.68	29.73	90.30	93.92	77.37	71.39
	Skywork/Skywork-Reward-V2-Qwen3-1.7B	1.7B	95.70	32.43	94.03	93.70	78.68	73.33
XXS	**Themis-RM 0.6B**	0.6B	90.61	45.95	83.58	76.80	73.05	78.79
	Skywork/Skywork-Reward-V2-Qwen3-0.6B	0.6B	93.95	35.14	89.55	92.13	73.25	70.42

Table 7: Detailed dataset-level preference accuracy of extant RMs and the `Themis-RM` suite on the `Functional Correctness (FC)` split of `Themis-CodeRewardBench`. Observe the marked drop in performance of extant RMs at judging partially correct solutions (`MBPP+Fix (Hard)`) as well as on open-domain GitHub commit preferences (`CommitPref`). For a detailed discussion of comparative results and the split-level average, refer to Section 5.1. Refer to Section 5.2 for experiments on how RMs trained on functional preferences fare when tested on non-functional preferences.

D.2 Execution Efficiency (EE) And Memory Efficiency (ME) Dataset-Level Accuracy

Auxiliary Training Objectives | Code RM | Criteria-Following RM | Math RM | Generative RM | Reasoning RM

	Model	Size	Execution Efficiency (EE)				Memory Efficiency (ME)	
			Pie4Perf	EvalPerf	ECCO	CommitPref	NoFunEval	CommitPref
XL	Qwen/Qwen2.5-Math-RM-72B	72B	51.74	50.47	62.41	57.98	51.35	57.94
XL	nvidia/AceMath-72B-RM	72B	60.43	64.62	62.16	60.92	51.35	53.57
XL	Qwen/WorldPM-72B-RLHFLow	72B	63.26	65.57	51.13	67.65	56.76	64.68
XL	ContextualAI/LMUnit-qwen2.5-72b	72B	11.74	66.04	15.29	3.78	8.11	6.75
XL	nvidia/Llama-3.3-Nemotron-70B-Reward	70B	59.13	66.98	56.64	69.33	64.86	66.67
XL	infly/INF-ORM-Llama3.1-70B	70B	60.00	70.75	59.40	62.61	51.35	64.29
XL	allenai/Llama-3.1-70B-Instruct-RM-RB2	70B	62.61	69.34	61.90	65.55	75.68	63.10
XL	allenai/Llama-3.1-Tulu-3-70B-SFT-RM-RB2	70B	61.74	67.92	62.91	73.11	56.76	68.65
XL	Nexusflow/Athene-RM-70B	70B	63.04	74.53	63.91	73.53	70.27	78.17
XL	Nexusflow/Starling-RM-34B	34B	47.17	65.09	51.13	65.13	45.95	66.27
XL	**Themis-RM 32B**	32B	82.83	86.32	83.46	98.34	89.19	96.03
XL	TIGER-Lab/AceCodeRM-32B	32B	61.74	60.85	60.15	44.12	43.24	40.08
XL	nvidia/Qwen3-Nemotron-32B-GenRM-Principle	32B	64.57	74.06	68.17	62.18	59.46	69.44
L	nicolinho/QRM-Gemma-2-27B	27B	52.61	60.85	51.63	47.48	59.46	45.24
L	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1	27B	62.61	73.58	65.16	64.29	64.86	62.30
L	Skywork/Skywork-Reward-Gemma-2-27B-v0.2	27B	52.39	58.96	49.12	49.16	45.95	49.60
L	internlm/internlm2-20b-reward	20B	55.65	64.15	57.64	69.75	64.86	65.87
L	**Themis-RM 14B**	14B	85.00	85.85	85.46	89.50	91.89	94.84
L	rubricreward/R3-Qwen3-14B-14k	14B	40.65	53.77	41.35	34.03	16.22	34.13
L	openbmb/UltraRM-13b	13B	57.61	53.77	46.87	65.97	78.38	64.68
M	**Themis-RM 8B**	8B	82.17	82.55	84.21	86.55	81.08	93.65
M	LARK-Lab/CodeScaler-8B	8B	53.48	72.17	59.90	68.91	62.16	73.81
M	rubricreward/R3-Qwen3-8B-14k	8B	35.65	52.83	45.11	27.73	29.73	28.97
M	Skywork/Skywork-Reward-V2-Qwen3-8B	8B	58.91	70.75	64.91	69.75	67.57	72.22
M	RLHFlow/ArmoRM-Llama3-8B-v0.1	8B	64.78	66.04	54.89	64.71	43.24	60.32
M	nicolinho/QRM-Llama3.1-8B-v2	8B	53.04	65.57	59.15	68.49	64.86	67.06
M	allenai/Llama-3.1-8B-Base-RM-RB2	8B	55.87	70.75	58.40	68.91	62.16	71.03
M	Ray2333/GRM-llama3-8B-sftreg	8B	63.26	53.30	52.13	65.97	56.76	57.54
M	LxzGordon/URM-LLaMa-3.1-8B	8B	60.22	65.57	54.64	67.65	56.76	66.27
M	NCSOFT/Llama-3-OffsetBias-RM-8B	8B	61.52	62.26	51.38	64.29	56.75	59.52
M	sfairXC/FsfairX-LLaMA3-RM-v0.1	8B	61.30	62.74	50.08	68.91	54.05	69.84
M	Nexusflow/Athene-RM-8B	8B	56.30	73.11	59.65	73.53	59.46	71.83
M	TIGER-Lab/AceCodeRM-7B	7B	52.17	60.38	57.64	54.20	70.27	59.13
M	eth-dl-rewards/internlm2-7b-reward-code-100k	7B	57.83	58.02	50.88	59.66	64.86	58.33
M	eth-dl-rewards/internlm2-7b-reward-math-100k	7B	55.65	59.43	48.87	66.39	51.35	58.73
M	reciprocate/mistral-7b-gsm8k-code-rm	7B	47.61	48.11	47.62	52.10	48.65	53.97
M	nvidia/AceMath-7B-RM	7B	65.65	60.38	56.14	51.26	45.95	54.76
M	openbmb/Eurus-RM-7b	7B	54.57	50.94	51.63	64.71	62.16	57.54
M	internlm/internlm2-7b-reward	7B	58.48	59.43	49.62	66.39	75.68	60.32
S	**Themis-RM 4B**	4B	82.17	81.13	83.96	88.24	89.19	92.86
S	LARK-Lab/CodeScaler-4B	4B	60.87	68.40	61.40	70.17	51.35	74.21
S	PKU-ONELab/CE-RM-4B	4B	58.91	50.47	54.14	40.76	35.14	39.68
S	rubricreward/R3-Qwen3-4B-14k	4B	43.91	53.77	43.11	21.01	16.22	22.62
S	Skywork/Skywork-Reward-V2-Qwen3-4B	4B	60.87	68.87	62.91	71.85	67.57	73.41
S	Ray2333/GRM-llama3.2-3B-sftreg	3B	60.22	54.72	54.64	57.14	62.16	63.10
XS	internlm/internlm2-1_8b-reward	1.8B	53.04	57.08	45.11	60.08	51.35	63.89
XS	**Themis-RM 1.7B**	1.7B	84.13	79.72	79.70	83.19	75.68	88.49
XS	LARK-Lab/CodeScaler-1.7B	1.7B	58.48	65.09	59.65	68.07	59.46	71.43
XS	Skywork/Skywork-Reward-V2-Qwen3-1.7B	1.7B	60.43	65.57	64.66	69.33	54.05	71.43
XXS	**Themis-RM 0.6B**	0.6B	82.39	69.81	82.46	80.25	70.27	89.29
XXS	Skywork/Skywork-Reward-V2-Qwen3-0.6B	0.6B	60.65	64.62	61.65	67.23	40.54	66.27

Table 8: Detailed dataset-level preference accuracy of extant RMs and the `Themis-RM` suite on the `Execution Efficiency (EE)` and `Memory Efficiency (ME)` splits of `Themis-CodeRewardBench`. Observe how most extant scalar RMs degenerate to near random performance on non-functional criteria and how the low-resolution scoring of generative RMs can render them unusable in such settings. For a detailed discussion of comparative results and the split-level averages, refer to Section 5.1. Refer to Section 5.2 for experiments on how RMs trained on functional preferences fare when tested on non-functional preferences.

D.3 Readability And Maintainability (R&M) And Security Hardness (SH) Dataset-Level Accuracy

Auxiliary Training Objectives | Code RM | Criteria-Following RM | Math RM | Generative RM | Reasoning RM

	Model	Size	Maintainability (R&M)		Security Hardness (SH)				
			NoFunEval	CommitPref	CodePrefBench	Vul4J	SecBench	NoFunEval	CommitPref
XL	Qwen/Qwen2.5-Math-RM-72B	72B	46.09	58.50	49.71	62.50	64.29	77.78	55.53
	nvidia/AceMath-72B-RM	72B	50.00	60.10	52.02	75.00	57.14	66.67	56.57
	Qwen/WorldPM-72B-RLHFLow	72B	59.38	71.41	66.47	37.50	92.86	88.89	68.14
	ContextualAI/LMUnit-qwen2.5-72b	72B	6.25	12.33	23.12	12.50	28.57	25.93	11.96
	nvidia/Llama-3.3-Nemotron-70B-Reward	70B	50.78	67.18	60.12	87.50	71.43	92.59	75.42
	infly/INF-ORM-Llama3.1-70B	70B	45.31	70.31	63.58	37.50	78.57	88.89	66.58
	allenai/Llama-3.1-70B-Instruct-RM-RB2	70B	56.25	72.65	71.68	62.50	78.57	92.59	72.56
	allenai/Llama-3.1-Tulu-3-70B-SFT-RM-RB2	70B	52.34	75.57	73.41	62.50	64.29	88.89	73.60
	Nexusflow/Athene-RM-70B	70B	53.91	77.46	80.92	50.00	85.71	92.59	77.37
	Nexusflow/Starling-RM-34B	34B	48.44	66.08	69.36	75.00	28.57	81.48	68.79
	Themis-RM 32B	32B	59.38	90.23	95.95	100.00	92.86	100.00	94.02
	TIGER-Lab/AceCodeRM-32B	32B	29.69	47.92	52.02	37.50	50.00	88.89	47.72
	nvidia/Qwen3-Nemotron-32B-GenRM-Principle	32B	49.22	57.91	77.46	62.50	64.29	81.48	60.47
L	nicolinho/QRM-Gemma-2-27B	27B	49.22	50.69	54.34	62.50	42.86	44.44	49.02
	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1	27B	44.53	69.44	46.82	50.00	71.43	81.48	66.19
	Skywork/Skywork-Reward-Gemma-2-27B-v0.2	27B	55.47	50.33	59.54	12.50	50.00	44.44	50.20
	internlm/internlm2-20b-reward	20B	42.97	66.08	67.63	62.50	78.57	66.67	65.76
	Themis-RM 14B	14B	60.94	89.79	96.53	87.50	92.86	96.30	95.19
	rubricreward/R3-Qwen3-14B-14k	14B	22.66	25.16	25.43	50.00	28.57	44.44	30.56
	openbmb/UltraRM-13b	13B	49.22	73.23	72.25	50.00	64.29	81.48	72.56
M	**Themis-RM 8B**	8B	60.94	88.40	94.80	100.00	92.86	100.00	92.72
	LARK-Lab/CodeScaler-8B	8B	57.03	74.98	63.01	100.00	85.71	92.59	74.38
	rubricreward/R3-Qwen3-8B-14k	8B	27.34	27.21	27.17	12.50	35.71	48.15	29.26
	Skywork/Skywork-Reward-V2-Qwen3-8B	8B	54.69	76.95	65.90	87.50	78.57	92.59	76.20
	RLHFlow/ArmoRM-Llama3-8B-v0.1	8B	55.47	65.57	66.47	37.50	85.71	85.19	59.43
	nicolinho/QRM-Llama3.1-8B-v2	8B	49.22	64.26	62.43	0.00	57.14	85.19	62.29
	allenai/Llama-3.1-8B-Base-RM-RB2	8B	52.34	72.50	71.68	62.50	71.43	81.48	70.87
	Ray2333/GRM-llama3-8B-sftreg	8B	51.56	67.47	72.83	25.00	57.14	85.19	65.67
	LxzGordon/URM-LLaMa-3.1-8B	8B	45.31	63.89	56.07	37.50	57.14	85.19	62.94
	NCSOFT/Llama-3-OffsetBias-RM-8B	8B	46.09	65.57	64.16	50.00	78.57	92.59	64.50
	sfairXC/FsfairX-LLaMA3-RM-v0.1	8B	50.00	68.93	65.90	50.00	85.71	81.48	66.06
	Nexusflow/Athene-RM-8B	8B	57.03	74.69	73.41	50.00	71.43	92.59	74.77
	TIGER-Lab/AceCodeRM-7B	7B	52.34	60.10	46.24	50.00	64.29	81.48	59.30
	eth-dl-rewards/internlm2-7b-reward-code-100k	7B	45.31	66.74	56.07	50.00	87.57	62.96	61.25
	eth-dl-rewards/internlm2-7b-reward-math-100k	7B	42.97	68.34	60.69	87.50	76.57	55.56	63.20
	reciprocate/mistral-7b-gsm8k-code-rm	7B	53.91	59.37	50.87	87.50	71.43	66.67	54.49
	nvidia/AceMath-7B-RM	7B	43.75	54.12	60.12	50.00	85.71	62.96	53.71
	openbmb/Eurus-RM-7b	7B	54.69	62.07	56.07	75.00	64.29	77.78	63.98
	internlm/internlm2-7b-reward	7B	44.53	69.51	63.58	62.50	85.71	77.78	62.68
S	**Themis-RM 4B**	4B	61.72	86.65	95.38	100.00	85.71	92.59	92.46
	LARK-Lab/CodeScaler-4B	4B	50.00	74.11	68.21	50.00	92.86	88.89	70.48
	PKU-ONELab/CE-RM-4B	4B	38.28	44.71	38.15	37.50	42.86	70.37	42.13
	rubricreward/R3-Qwen3-4B-14k	4B	17.19	24.58	26.59	25.00	7.14	37.04	27.44
	Skywork/Skywork-Reward-V2-Qwen3-4B	4B	51.56	76.51	71.68	37.50	78.57	88.89	73.47
	Ray2333/GRM-llama3.2-3B-sftreg	3B	49.22	67.54	66.47	37.50	78.57	81.48	66.32
XS	internlm/internlm2-1_8b-reward	1.8B	67.19	64.48	55.49	62.50	50.00	74.07	63.72
	Themis-RM 1.7B	1.7B	55.47	82.35	94.80	62.50	78.57	92.59	88.04
	LARK-Lab/CodeScaler-1.7B	1.7B	45.31	69.44	86.21	37.50	71.43	77.78	65.67
	Skywork/Skywork-Reward-V2-Qwen3-1.7B	1.7B	47.66	70.75	73.41	50.00	71.43	81.48	69.44
XXS	**Themis-RM 0.6B**	0.6B	65.62	80.01	86.71	100.00	85.71	92.59	87.65
	Skywork/Skywork-Reward-V2-Qwen3-0.6B	0.6B	48.44	71.41	64.16	62.50	64.29	66.32	66.32

Table 9: Detailed dataset-level preference accuracy of extant RMs and the `Themis-RM` suite on the `Readability And Maintainability (R&M)` and `Security Hardness (SH)` splits of `Themis-CodeRewardBench`. Observe how most extant scalar RMs degenerate to near random performance on non-functional criteria and how the low-resolution scoring of generative RMs can render them unusable in such settings. For a detailed discussion of comparative results and the split-level averages, refer to Section 5.1. Refer to Section 5.2 for experiments on how RMs trained on functional preferences fare when tested on non-functional preferences.

D.4 General-Domain Reward Modeling Performance

Auxiliary Training Objectives | Code RM | Criteria-Following RM | Math RM | Generative RM | Reasoning RM

	Model		Size	External Benchmark Accuracy		
				RewardBench V1	RewardBench V2	JudgeBench
XL	Qwen/WorldPM-72B-RLHFLow		72B	90.88	67.92	55.21
	nvidia/Llama-3.3-Nemotron-70B-Reward		70B	93.88	70.49	73.47
	Nexusflow/Athene-RM-70B		70B	91.22	68.76	63.45
	Themis-RM 32B	{}	32B	94.89	72.34	71.65
	TIGER-Lab/AceCodeRM-32B	{}	32B	23.58	67.98	66.77
L	**Themis-RM 14B**	{}	14B	94.11	71.44	70.85
M	**Themis-RM 8B**	{}	8B	93.69	65.87	69.97
	LARK-Lab/CodeScaler-8B	{}	8B	94.66	76.51	70.05
	Skywork/Skywork-Reward-V2-Qwen3-8B		8B	94.76	76.93	67.90
	Nexusflow/Athene-RM-8B		8B	87.48	62.96	61.12
	TIGER-Lab/AceCodeRM-7B	{}	7B	22.74	63.16	61.09
S	**Themis-RM 4B**	{}	4B	92.46	63.81	68.02
	LARK-Lab/CodeScaler-4B	{}	4B	94.32	75.13	68.44
	Skywork/Skywork-Reward-V2-Qwen3-4B		4B	94.06	74.26	65.43
XS	**Themis-RM 1.7B**	{}	1.7B	89.17	56.22	63.29
	LARK-Lab/CodeScaler-1.7B	{}	1.7B	91.13	68.44	66.17
	Skywork/Skywork-Reward-V2-Qwen3-1.7B		1.7B	91.64	67.71	66.48
XXS	**Themis-RM 0.6B**	{}	0.6B	83.41	49.61	63.84
	Skywork/Skywork-Reward-V2-Qwen3-0.6B		0.6B	86.32	60.83	63.65

Table 10: External general-domain reward modeling evaluation of the `Themis-RM` suite on RewardBench V1 (Lambert et al., 2025), RewardBench V2 (Malik et al., 2025), and JudgeBench (Tan et al., 2025). We benchmark against a selection of the highest scoring extant RMs on `Themis-CodeRewardBench`.