Advancing LLM Reasoning Generalists with Preference Trees

Anonymous authors
Paper under double-blind review

Abstract

We introduce **Eurus**, a suite of large language models (LLMs) optimized for reasoning. Finetuned from Mistral-7B, Llama-3-8B, and Mixtral-8x22B, Eurus models achieve state-of-the-art results among open-source models on a diverse set of benchmarks covering mathematics, code generation, and logical reasoning problems. Notably, Eurux-8x22B outperforms GPT-3.5 Turbo in reasoning through a comprehensive benchmarking across 12 test sets covering five tasks. The strong performance of Eurus can be primarily attributed to **UltraInteract**, our newly-curated large-scale, high-quality training data dataset specifically designed for complex reasoning tasks. UltraInteract can be used in both supervised fine-tuning, preference learning, and reward modeling. It pairs each instruction with a preference tree consisting of (1) reasoning chains with diverse planning strategies in a unified format, (2) multi-turn interaction trajectories with the environment and the critique, and (3) pairwise positive and negative responses to facilitate preference learning. UltraInteract allows us to conduct an in-depth exploration of preference learning for reasoning tasks. Our investigation reveals that some well-established preference learning algorithms may be less suitable for reasoning tasks compared to their effectiveness in general conversations. The hypothesis is that in reasoning tasks, the space of correct answers is much smaller than that of incorrect ones, so it is necessary to explicitly increase the reward of chosen data. Therefore, in addition to increasing the reward margin as many preference learning algorithms do, the absolute values of positive responses' rewards should be positive and may serve as a proxy for performance. Inspired by this, we derive a novel reward modeling objective and empirically that it leads to a stable reward modeling curve and better performance. Together with UltraInteract, we obtain a strong reward model.

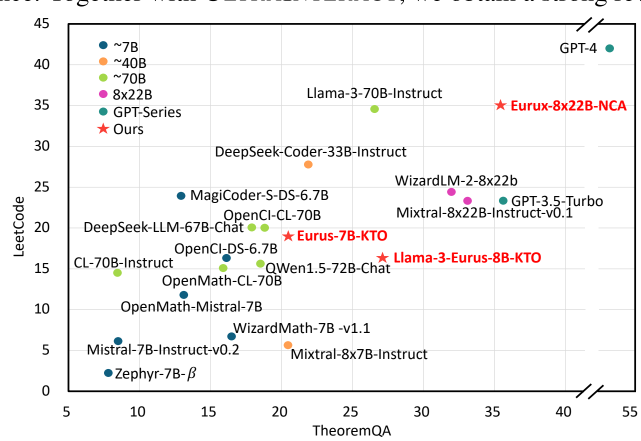

Figure 1: Evaluation results on LeetCode and TheoremQA, two challenging OOD coding and math benchmarks with only test sets. Our Eurus-7B is comparable with baselines that are 10x larger and Eurux-8x22B is the only one on par with GPT-3.5 Turbo.

1 Introduction

Current alignment techniques have significantly advanced the development of open-source large language models (LLMs) that effectively meet user expectations and align with human values (Touvron et al., 2023; Tunstall et al., 2023). On complex reasoning, success has been achieved by specializing models for specific capabilities, such as coding (Wei et al., 2023; Guo et al., 2024a; Zheng et al., 2024) and solving math problems (Fu et al., 2023; Yue et al., 2023; Luo et al., 2023a; Toshniwal et al., 2024). However, these models still fall short, by large margins, of the most advanced proprietary models in their all-around capabilities to tackle a diverse range of challenging problems. We conjecture that this performance gap can be primarily attributed to (1) the lack of high-quality alignment data and (2) the underexploration of preference learning techniques for improving models' complex reasoning capabilities. In this paper, we take strides towards bridging this gap by addressing both factors and developing EURUS.

EURUS consists of a suite of LLMs and reward model finetuned from Mistral-7B (Jiang et al., 2023a), Llama-3 (Meta, 2024), and Mixtral-8x22B (Jiang et al., 2024). Across a diverse set of complex reasoning benchmarks that are mostly out-of-distribution (OOD), EURUS achieves state-of-the-art overall performance among all open-source models. In particular, EURUS excels in solving challenging problems that often require sophisticated planning, reasoning, tool integration, and the ability to interact with and learn from the environment and users. As shown in Figure 1, on university-level STEM questions TheoremQA (Chen et al., 2023) and competition-level coding problems LeetCode Contest (Guo et al., 2024a), EURUS significantly outperforms all open-source models, achieving comparable performance to GPT-3.5 Turbo. Besides, our reward model, EURUS-RM-7B, outperforms baselines with a 5x larger size on various reward model benchmarks and demonstrates superiority on best-of-N and MCTS-guided decoding tasks.

EURUS models are trained on ULTRAINTERACT, our newly-curated, large-scale, and high-quality alignment data specifically designed to improve LLMs' reasoning capabilities. ULTRAINTERACT consists of a diverse set of instructions spanning math, coding, and logical reasoning problems from 12 established datasets. For each instruction, ULTRAINTERACT collects a preference tree that includes: (1) **Diverse planning strategies in a unified pattern,** such as sequential processing (Wei et al., 2022) and tool creation (Qian et al., 2023), followed by executing step-by-step actions formatted in either text or code, to provide divserse reasoning trajectories. (2) **Multi-turn interaction trajectories with the environment and the critique,** to improve models' capabilities to learn from feedback and correct previous errors (Wang et al., 2023b). (3) **Paired correct and incorrect actions organized in tree structures,** to facilitate preference learning. In total, ULTRAINTERACT contains 86K instructions and 220K action pairs, where each pair consists of an instruction, a correct response, and an incorrect one. Conceptually, ULTRAINTERACT's data resemble imbalanced binary trees as shown in Figure 2.

ULTRAINTERACT can be used in both supervised fine-tuning and preference learning. Our experiments show that, using ULTRAINTERACT along with established datasets in instruction fine-tuning already achieves strong performance. ULTRAINTERACT further facilitates preference learning for reasoning tasks, improving the performance even further with KTO (Ethayarajh et al., 2024) and NCA (Chen et al., 2024a). Surprisingly, applied to an instruction finetuned EURUS model, DPO (Rafailov et al., 2023) sometimes hurts the performance.

Through careful analysis, we provide evidence that the performance in reasoning correlates with the value of rewards of chosen data—a higher final reward often indicates a better reasoning capability. Besides, our investigation suggests that DPO may be less suitable for reasoning tasks than KTO and NCA. Inspired by this fresh finding, we devise a new objective for reward modeling to augment the Bradley-Terry objective (Bradley & Terry, 1952), explicitly encouraging training to increase the absolute rewards of chosen solution and decrease those of rejected data. Furthermore, ULTRAINTERACT leads to our reward model EURUS-RM-7B, which achieves a better correlation with human annotators than all existing models on AutoJ (Li et al., 2023a) and MT-Bench (Zheng et al., 2023), including GPT-4 (OpenAI, 2023). EURUS-RM-7B demonstrates especially strong preference modeling performance on reasoning tasks.

In short, we compiled this work by first synthesizing both SFT and preference datasets to improve the reasoning ability of open-source models (Section §2). We examined the effectiveness of our datasets by training both policy and reward models (Section §3). We evaluated the performance of policy models in Section §4, during which we observed a correlation between reward patterns and benchmark performances. Next, we then evaluated our reward models and validated that our insights on the reward-performance correlation can be converted into gains in model training (Section §5). Finally, we ablate some factors in our dataset construction in Section §6.

2 ULTRAINTERACT: TREE-STRUCTURED ALIGNMENT DATA FOR REASONING

Solving complex problems often requires the model's capability in planning and reasoning, integrating with tools, and interacting with and learning from both the environment and the users. This is reflected in ULTRAINTERACT's design choices: (1) Its instructions are diverse, challenging, and of a large scale (§2.1); (2) It provides multi-turn trajectories that solve the input instruction through multiple turns of interaction with and learning from the environment and critique. At each turn, it breaks down the problem into smaller ones (§2.2). (3) ULTRAINTERACT includes pairwise data to facilitate preference learning (§2.3).

Conceptually, ULTRAINTERACT collects a preference tree for each instruction, with the instruction being the root and each action a node (Figure 2). A trajectory is a root-to-leaf path consisting of a sequence of actions. In each preference tree, all nodes of correct actions and all trajectories ending with correct actions can be used for SFT. Paired correct and incorrect nodes or trajectories can be used for preference learning.

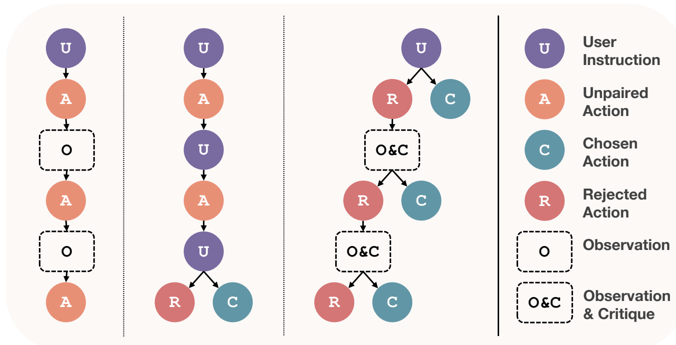

Figure 2: Left: CodeActInstruct (Wang et al., 2024) and Code-Feedback (Zheng et al., 2024); Middle: HH-RLHF (Bai et al., 2022); Right: ULTRAINTERACT. Each instruction in ULTRAINTERACT is constructed as a preference tree.

2.1 INSTRUCTION SELECTION EMPHASIZING COMPLEXITY, QUALITY, AND DIVERSITY

We target three representative reasoning tasks: math problem-solving, code generation, and logical reasoning. The complexity, quality, and diversity of the alignment data are crucial to the model's performance (Liu et al., 2023). Following Wang et al. (2023b), we select challenging problems that GPT-3.5-Turbo fails to solve. We intentionally restrict the selection of the datasets to those with ground-truth solutions, aiming to ensure high-quality oversight signals rather than relying on LLM-as-a-judge annotation (Weyssow et al., 2024). Besides, the gold solutions also serve as references for the critique model to generate feedback. To promote ULTRAINTERACT's diversity, we pick datasets of different categories. For each dataset, we include distinct reasoning patterns based on question categories or formulations necessary to solve the problems. Table 8 summarizes the datasets selected by ULTRAINTERACT. Except for MATH, none of the training datasets is used in our evaluation.

2.2 DECOMPOSITION AND INTERACTION AT EACH TURN

Figure 3 provides an illustrative example. In what follows, we connect the actor model with a Python interpreter as the "environment". We use GPT-3.5 Turbo as the actor model.

Following Wang et al. (2024), the actor model first decomposes the input problem into several sub-problems and then solves each by generating Python code pieces as actions and using the environment to execute them. To promote solution diversity, the actor model randomly samples one reasoning schema in the form of either CoT (Wei et al., 2022) or modularization programming (Qian et al., 2023; Yuan et al., 2023). The actor then generates actions in text or code to solve each sub-problem, with each step being marked by explicit notations.

Multi-turn interactions with the environment are often necessary to solve challenging problems (Wang et al., 2023b). To improve such capabilities of the models, ULTRAINTERACT collects trajectories in which the actor model interacts with the environment and a critique model (a proxy for user) and refines its action based on their feedback.

The environment receives an action from the actor model along with the interaction history, and then the code interpreter returns two kinds of "*Observation*": (1) Python execution results, either program outputs or error traceback messages; (2) binary feedback, indicating whether the solution is

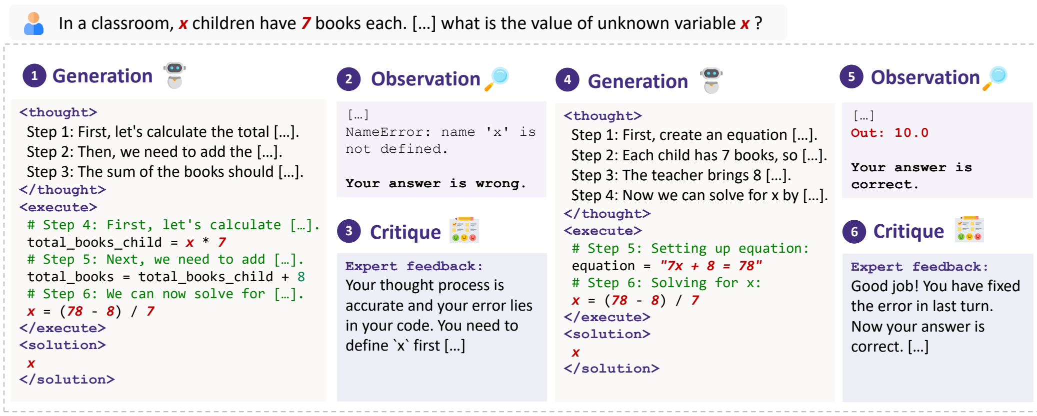

Figure 3: An illustrative example of an ULTRAINTERACT trajectory over two turns. In each turn, the actor model generates step-by-step reasoning chains, and the environment and the critique model provide observations and textual critique respectively.

correct or not. Then, the observations along with the history will be passed to a *critique* model, which locates the errors and provides suggestions for improvements. To avoid potential bias introduced by self-correction (Wang et al., 2023b; Xu et al., 2024), we adopt a stronger model, GPT-4[1]., as the critique and ensure critique quality by providing GPT-4 with ground truth answers as references.

This procedure resembles Wang et al. (2024). However, we adopt more diverse reasoning patterns to teach LLMs to learn rationales rather than simply memorizing answers (Mitra et al., 2023), and learn to create and use tools (Qian et al., 2023; Yuan et al., 2023; Qin et al., 2023). Besides, we believe that it is important for LLMs to learn from the feedback provided by the critique rather than solely from observations of the environment.

2.3 PREFERENCE TREES FACILITATES PREFERENCE LEARNING ACROSS MULTIPLE TURNS

Unlike open-ended conversations, where human preference is ambiguous and challenging to specify, many reasoning tasks have clear and objective preferences for *correct* actions. The preference annotation is therefore an evaluation of the correctness of the solutions *conditioning ground truth ones*, which come with the datasets in ULTRAINTERACT. This eliminates the need for human or LLM-based preference annotation and ensures high data quality. To facilitate preference learning, ULTRAINTERACT pairs correct and incorrect actions in each turn.

Sampling Paired Correct and Incorrect Actions at Each Turn. For each instruction in ULTRAINTERACT, we sample, from the actor model, a pair of correct and incorrect actions following §2.2. We follow Cui et al. (2023) to sample the pair from different actor models to ensure response diversity. To prevent models from exploiting shortcuts based on surface features, we exclude instances that fail to pass the Python syntax check.

Certain challenging problems in ULTRAINTERACT pose difficulties in obtaining correct actions, even using strong actors such as GPT-4, with nearly zero *pass@100* accuracies. To improve the pass rates of the actor models while keeping the expense under control, we sequentially take the following steps. (1) Directly sampling 20 actions and randomly keeping a correct one, if any. (2) If no correct action is obtained, we repeat the above process up to three times, progressively switching from more cost-effective models to the strong yet expensive GPT-4 Turbo. (3) For the remaining difficult problems where no correct action is acquired after the previous two steps, we provide the actor with ground-truth rationales and answers, and then apply various techniques to elicit correct actions. The specific information provided and the techniques applied vary depending on the tasks (Appendix A.2).

Tree-structured Action Pairs Across Multiple Turns. After each turn, the correct action concludes its trajectory. We expand the *incorrect* action into the next turn, and have the actor interact with the environment and the critique to refine its solution (§2.2). We then repeat the procedures introduced earlier in this section to collect an additional action pair. By expanding the *incorrect*

[1]The majority of data collection and experiments are prior to the release of GPT-4o.

action, ULTRAINTERACT can provide data to help models learn from feedback, and collect multiple action pairs for preference learning across multiple turns.

Conceptually, for every instruction, ULTRAINTERACT constructs a binary preference tree with each action being a node (Figure 2). We cap the tree at a maximum of five turns.

Additional Instruction-action Pairs for Challenging Problems. We believe the challenging instructions that make it to step (3) above can provide valuable training signals. Therefore, for a subset of these problems with multiple ground truth solutions, we further sample additional correct actions to cover all ground truths. Accordingly, we further sample incorrect actions to pair with these additional correct actions, so that they can be used in both supervised fine-tuning and preference learning.

With preference trees, ULTRAINTERACT enables comparisons at every turn, in contrast to comparing only at the last turn (Bai et al., 2022), and thus can improve the models' interaction ability. Closing this section, Table 1 summarizes statistics of ULTRAINTERACT. Check more details in Appendix A.4. We show examples of the data in Appendix G.

Table 1: Some statistics of ULTRAINTERACT.

Task	Type		# Instructions	# Turns per Traj.					# Tokens per Traj.	Avg. # Traj per Ins.	Total # Pairs	# Correct Answers
	w/ Interaction?	w/ Tool?		T1	T2	T3	T4	T5				
Math	✓	✓	22,928	10,440	4,122	1,898	904	5,564	1,750.0	1.0	42,780	68,033
	✗	✓	2,757	16,154	-	-	-	-	439.1	5.9	13,217	16,154
	✓	✗	22,639	10,708	3,521	1,459	723	6,228	1,521.9	1.0	44,750	62,182
	✗	✗	2,083	16,348	-	-	-	-	538.1	7.8	12,624	16,348
Coding	✓	-	20,463	13,265	2,584	987	379	3,248	1,728.5	1.0	18,106	22,215
	✗	-	8,495	92,618	-	-	-	-	1,070.4	5.5	78,634	92,618
Logic	✓	✓	2,086	1,685	298	72	8	23	1,299.8	1.0	1,750	2,198
	✓	✗	4,467	2,453	1,674	340	0	0	1,266.7	1.0	7,958	7,231
Total	-	-	85,918	163,671	12,199	4,756	2,014	15,063	1,201.8	2.3	219,819	286,979

3 EURUS: STATE-OF-THE-ART OPEN LLMS IN REASONING

ULTRAINTERACT helps us develop EURUS, a suite of LLMs and a reward model (RM).

Supervised Fine-Tuning. EURUS-7B-SFT and LLAMA-3-EURUS-8B-SFT are fine-tuned from Mistral-7B (Jiang et al., 2023a) and Llama-3-8B (Meta, 2024) respectively, and EURUX-8X22B-SFT from Mixtral-8x22B (Jiang et al., 2024). First, we perform SFT using all correct actions (287K) in ULTRAINTERACT. We find it yields better performance to discard interaction history and train only on correct leaf nodes in each tree. To improve general instruction-following ability, we include some open-source SFT datasets in our data mixture. Please find mixture details in Appendix B.

Preference Learning. Based on EURUS-SFT models, we explore three preference learning algorithms, DPO (Rafailov et al., 2023), KTO (Ethayarajh et al., 2024), and NCA (Chen et al., 2024a). Differently from SFT, here we include all multi-turn trajectory pairs in our ULTRAINTERACT (220K) and include all UltraFeedback (Cui et al., 2023) pairs (340K).

Reward Modeling. Similarly to the preference learning, we use all 220K multi-turn trajectory pairs from ULTRAINTERACT; it is further augmented with the 240K single-turn action pairs from ULTRAINTERACT. More details are in the Appendix B. We include all 340K pairs from UltraFeedback and one pair for each instruction from UltraSafety (Guo et al., 2024b), totaling 3K. EURUS-RM-7B is initialized from EURUS-7B-SFT with a new linear layer.

Our findings in §4.2 indicate that the absolute values of rewards make a big difference in the models' reasoning performance, with decreasing rewards of chosen actions possibly resulting in a perfroamce degradation. We therefore augment the established Bradley-Terry (BT) objective $\mathcal{L}_{\text{BT}}$ with an additional term $\mathcal{L}_{\text{DR}}$ to **dir**ectly increase the reward of the chosen actions for instances from ULTRAINTERACT, and decrease those of the rejected ones:

$$\mathcal{L}_{\text{ULTRAINTERACT}} = \underbrace{-\log\Big(\sigma\big(r_\theta(x, y_c) - r_\theta(x, y_r)\big)\Big)}_{\mathcal{L}_{\text{BT}}\text{: optimize relative rewards}} \underbrace{-\log\Big(\sigma\big(r_\theta(x, y_c)\big)\Big) - \log\Big(\sigma\big(-r_\theta(x, y_r)\big)\Big)}_{\mathcal{L}_{\text{DR}}\text{: increase } r_\theta(x, y_c) \text{ and decrease } r_\theta(x, y_r)}$$

We train ULTRAINTERACT examples with $\mathcal{L}_{\text{ULTRAINTERACT}}$, while for instances from other datasets, we train with $\mathcal{L}_{\text{BT}}$ only. θ denotes the reward model's parameters, $r_\theta\left(x, y_c\right)$ and $r_\theta\left(x, y_r\right)$ the rewards on the chosen and rejected actions respectively. We ablate the importance of $\mathcal{L}_{\text{BT}}$ and $\mathcal{L}_{\text{DR}}$ in §6.2.

Table 2: Open-source LLM baselines that we compare to.

Type	Models
General Purpose	Mistral-7B-Instruct-v0.2 (Jiang et al., 2023a), Zephyr-7B-β (Tunstall et al., 2023), OpenChat-3.5-1210 (Wang et al., 2023a), Starling-LM-7B-α (Zhu et al., 2023), Mixtral-8x7B-Instruct (Jiang et al., 2023a), DeepSeek-LLM-67B-Chat (DeepSeek-AI, 2024), QWen1.5-72B-Chat (Bai et al., 2023)
Coding	Magicoder-S-DS-6.7B (Wei et al., 2023), OpenCodeInterpreter (OpenCI for short, DS-6.7B/CL-70B) (Zheng et al., 2024), DeepSeek-Coder-33B-Instruct (Guo et al., 2024a), and CodeLLaMA-70B-Instruct(Roziere et al., 2023).
Math	MAmmoTH-7B-Mistral (Yue et al., 2023), WizardMath-7B-v1.1 (Luo et al., 2023a), OpenMath (Mistral-7B/CodeLLaMA-70B) (Toshniwal et al., 2024).

4 EVALUATION OF EURUS MODELS

Evaluation Setup. We consider both single-turn and multi-turn reasoning. For single-turn evaluation, we consider HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and LeetCode (Guo et al., 2024a) for coding, GSM-Plus (Li et al., 2024), MATH, TheoremQA (Chen et al., 2023), SVAMP (Patel et al., 2021), and ASDiv (Miao et al., 2020) for math, and BBH-Hard (Suzgun et al., 2022) for reasoning. We evaluate with pass@1 accuracy. We also use IFEval (Zhou et al., 2023) to assess the instruction-following ability and report the prompt-level loose score. For multi-turn evaluation, we adopt MINT (Wang et al., 2023b) and only consider the coding and math problems. We report the success rate at Turn 5. Please find further details on evaluation setups and evaluations beyond reasoning in Appendix C. As shown in Table 2, we compare our EURUS with general-purpose models, and those specialized in coding and math of various sizes. We also summarize the results of GPT-3.5 Turbo and GPT-4 reported in previous works.

Table 3: Overall performance. All test sets except MATH are out-of-distribution to our models and most baselines. MAmmoTH, OpenChat, and Starling-LM have been trained on TheoremQA test sets. We ~~strikethrough~~ the contaminated numbers.

Model	Coding			Math					Reasoning	Ins-Following	Multi-Turn		Avg.
	HumanE.	MBPP	LeetC.	GSM-Plus	MATH	Theo.QA	SVAMP	ASDiv	BBH	IFEval	Code	Math	
~7B													
Mistral-7B-Instruct-v0.2	39.0	30.8	6.1	15.7	9.5	8.5	42.9	49.5	62.4	44.4	7.4	26.2	28.5
Zephyr-7B-β	29.3	35.8	2.2	23.3	5.0	7.8	19.1	28.0	61.8	39.7	5.2	16.9	22.8
OpenChat-3.5-1210	64.0	61.7	11.7	46.7	28.1	~~19.1~~	75.4	77.0	67.0	50.3	21.3	32.4	~~46.2~~
Starling-LM-7B-α	46.3	51.1	8.9	23.7	21.5	~~12.0~~	26.3	39.8	67.1	26.1	18.4	28.9	~~30.8~~
Magicoder-S-DS-6.7B	75.6	**70.4**	**23.9**	16.4	19.9	13.1	61.6	62.8	57.0	21.1	**27.9**	8.0	38.1
OpenCI-DS-6.7B	**76.8**	66.2	16.1	41.5	31.6	16.1	74.5	79.8	53.9	22.6	5.9	1.3	40.5
MAmmoTH-7B-Mistral	24.4	42.4	7.2	40.1	36.0	~~26.3~~	60.7	72.3	57.7	34.9	3.7	6.7	~~34.4~~
WizardMath-7B-v1.1	50.0	53.9	6.7	54.6	30.0	16.5	57.8	73.5	64.4	22.6	16.2	8.9	37.9
OpenMath-Mistral-7B	33.5	46.6	11.7	**59.4**	**39.1**	13.1	83.4	79.8	58.6	15.0	2.9	5.3	37.4
EURUS-7B-SFT	55.5	59.1	20.0	52.1	32.6	20.0	82.2	84.1	64.6	44.0	15.4	28.4	46.5
+ DPO	50.6	52.1	8.3	51.0	28.3	20.9	78.7	83.8	65.0	42.5	20.6	32.4	44.5
+ KTO	56.1	58.6	18.9	55.0	33.2	20.6	84.4	85.0	67.6	43.1	19.1	**43.6**	48.8
+ NCA	55.5	60.2	14.4	54.9	34.2	20.9	**84.6**	85.4	64.3	42.7	21.3	38.7	48.1
LLAMA-3-EURUS-8B-SFT	51.2	57.9	17.2	50.7	32.0	21.3	82.2	83.7	**72.4**	47.1	18.4	24.5	46.6
+ DPO	43.9	50.1	11.7	45.3	26.8	21.4	54.1	67.5	71.3	**56.7**	21.3	39.2	42.4
+ KTO	51.8	58.1	15.6	54.8	34.2	24.9	80.1	86.7	71.7	50.6	26.5	37.4	49.4
+ NCA	50.6	60.4	15.6	55.2	34.8	**25.4**	79.9	**87.5**	71.7	56.2	21.3	36.3	**49.6**
~40B													
Mixtral-8x7B-Instruct	50.6	50.1	5.6	**49.6**	**25.9**	20.4	66.4	68.8	**73.5**	**48.8**	12.5	**37.3**	42.5
DeepSeek-Coder-33B-Ins	**82.3**	**73.9**	**27.8**	29.5	20.2	**21.9**	**75.2**	**85.0**	61.5	26.1	**35.3**	21.8	**46.7**
~70B													
CodeLLaMA-70B-Instruct	56.7	58.6	14.4	34.9	12.0	8.4	63.5	70.1	74.5	24.0	3.7	14.2	36.3
DeepSeek-LM-67B-Chat	70.7	65.7	20.0	65.0	41.0	17.9	74.0	84.0	78.9	52.7	30.9	41.8	53.5
QWen1.5-72B-Chat	71.3	56.9	15.6	65.4	43.4	18.5	79.5	79.1	78.0	53.4	27.2	38.2	52.2
OpenCI-CL-70B	**77.4**	**71.7**	20.0	46.1	29.2	18.8	76.1	79.4	66.7	26.8	30.9	12.0	46.3
OpenMath-CL-70B	39.0	52.6	15.0	62.2	45.9	15.9	86.6	82.8	59.9	15.7	14.0	0.4	40.8
WizardLM-2-8x22B	72.0	64.2	24.4	57.0	**50.9**	32.0	81.2	82.2	85.3	**68.9**	13.2	43.2	56.2
Mixtral-8x22B-Instruct-v0.1	76.2	68.7	23.3	51.2	49.6	33.1	88.3	**92.4**	**86.0**	67.1	**39.0**	59.7	61.2
EURUX-8X22B-KTO	71.3	68.9	29.4	**68.3**	48.1	35.3	91.5	90.6	83.6	67.1	38.2	57.5	62.5
EURUX-8X22B-NCA	75.0	69.7	**35.0**	68.1	49.0	**35.5**	**91.6**	92.1	83.5	67.1	33.1	**63.0**	**63.6**
Proprietary Models													
GPT-3.5 Turbo	76.8	82.5	23.3	61.2	37.8	35.6	83.0	90.6	70.1	56.6	29.4	36.9	57.0
GPT-4	85.4	**83.5**	**41.8**	**85.6**	**69.7**	**52.4**	**94.8**	**92.6**	**86.7**	**79.7**	**59.6**	**65.8**	**74.8**

4.1 RESULTS

According to Table 3, all **EURUS variants achieve the best overall performance among open-source models of similar sizes. EURUS even outperforms specialized models in corresponding domains in many cases.** Notably, EURUS-7B and LLAMA-3-EURUS-8B outperform baselines that are 5× larger, and **EURUX-8X22B achieves better performance than GPT-3.5 Turbo**. EURUS's instruction-following performance is among the best general-purpose models, substantially better than specialized ones.

Preference learning with ULTRAINTERACT can further improve the performance, especially in math and the multi-turn ability. KTO and NCA consistently improve the models' performance in all five math benchmarks and mult-turn evaluations, while their effects vary in others. Since SFT models only use the single-turn data from ULTRAINTERACT while preference learning uses the multi-turn ones, the improvements in interaction ability should also be attributed to ULTRAINTERACT rather than the algorithms alone (See Appendix H for data mixture ablation results). Surprisingly, we observe that **DPO is not as effective as its variants**. We analyze this phenomenon in §4.2.

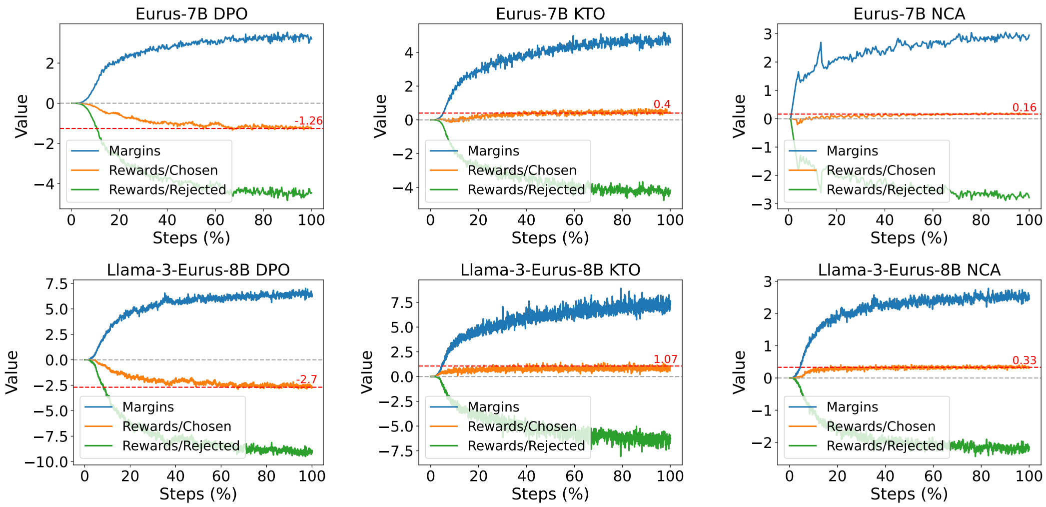

Figure 4: Reward patterns of EURUS-7B and LLAMA-3-EURUS-8B preference learning with DPO, KTO, and NCA. For all algorithms, the rewards of rejected data keep decreasing and the margins between chosen and rejected data keep increasing. However, the rewards of chosen data decrease below zero in DPO while keeping increasing and staying positive in KTO and NCA. The absolute values of the reward in the last step (in red) of the three algorithms positively correlate with their performance in Table 3.

4.2 EXPLICIT REWARD AS A PROXY? HYPOTHESIS FOR PREFERENCE LEARNING IN REASONING

We investigate the reason why DPO behaves differently than KTO and NCA. We start by empirically inspecting the rewards throughout the preference learning process, as shown in Figure 4. Rewards for chosen and rejected data both keep decreasing through DPO, though the rewards for chosen data is still higher hence the loss decreases. In KTO and NCA, the rewards of chosen data keep increasing with those of rejected data decreasing.

Therefore, we hypothesize it is the distinction in the trend of rewards that leads to the performance gap between DPO and the other two algorithms. This distinction can be attributed to that DPO, derived from the Bradley-Terry model, only optimizes the relative differences between chosen and rejected data overlooking the absolute values of the rewards. This is a non-issue in alignment with general human values where preference is "relative" and there can be many valid answers to the same input. However, in reasoning tasks, the space of correct answers is much smaller than that of incorrect ones. Further, we notice that the rewards of chosen data in the last training step follow the ranking order of KTO $>$ NCA $>$ DPO, positively correlate with their performance trends. Therefore, we believe that increasing the rewards of the chosen data is especially beneficial in preference learning for reasoning tasks.

5 EVALUATION OF EURUS-RM-7B

5.1 REWARD MODELING PERFORMANCE

We evaluate EURUS-RM-7B on three RM benchmarks, RewardBench (Lambert et al., 2024), AutoJ (Li et al., 2023a), and MT-Bench (Zheng et al., 2023). Aiming for a more realistic OOD evalation, we exclude the "prior sets" split from RewardBench, since many baselines train on the datasets that this split contains. We compare with PairRM (Jiang et al., 2023b), Starling-RM-7B/34B (Zhu et al., 2023), UltraRM-13B (Cui et al., 2023), GPT-3.5 Turbo, and GPT-4.

Table 4: Results on reward modeling benchmarks. UF: UltraFeedback; US: UltraSafety. The best performance in each benchmark is in bold and the second best one is underlined. Most baseline results are from Jiang et al. (2023b) and Lambert et al. (2024).

Model	Reward Bench					AutoJ				MT-Bench
	Chat	Chat-Hard	Safety	Reasoning	Avg.	Code	Math	Others	Overall	
PairRM	90.2	53.0	31.5	60.0	58.7	58.3	52.8	58.9	59.1	59.0
Starling-RM-7B	98.0	43.4	88.6	74.6	76.2	59.2	47.2	61.4	60.8	56.8
Starling-RM-34B	96.9	59.0	89.9	90.3	**84.0**	65.8	54.2	62.3	62.6	60.4
UltraRM-13B	96.1	55.3	45.8	82.0	69.8	55.0	43.1	59.6	59.9	56.0
GPT-3.5 Turbo	-	-	-	-	-	36.6	40.3	41.2	42.7	57.1
GPT -4	-	-	-	-	-	69.2	51.4	61.4	61.9	63.9
EURUS-RM-7B	96.5	65.3	80.7	87.0	<u>82.4</u>	87.5	82.5	78.0	80.7	<u>79.4</u>
w/o $\mathcal{L}_{\text{DR}}$	96.4	59.9	79.5	77.5	78.3	83.8	82.5	78.9	80.7	79.3
w/o $\mathcal{L}_{\text{BT}}$	96.8	58.5	83.8	84.2	80.8	88.8	92.5	79.4	**81.9**	**79.6**
w/o US	96.5	66.2	67.7	81.7	73.3	87.5	90.0	79.2	<u>81.8</u>	79.2
w/o UF + US	95.1	61.1	63.7	73.4	78.0	73.8	80.0	71.7	72.8	73.0

Results. Table 4 summarizes reward modeling performance. Takeaways are as follows:

EURUS-RM-7B stands out as the best 7B RM overall, and achieves similar or better performance than much larger baselines. Particularly, it outperforms GPT-4 in certain tasks. EURUS-RM-7B achieves a better correlation with human experts than all existing models on AutoJ and MT-Bench, and it achieves comparable performance to the 5× larger Starling-RM-34B on RewardBench. On RewardBench, EURUS-RM-7B outperforms all baselines on the "Chat-Hard" split while achieving very competitive performance on the "Reasoning" split. Across the AutoJ splits, EURUS-RM-7B outperforms all baselines, with the only exception being GPT-4's results on Coding.

Our training objective is beneficial in improving RM performance on hard problems and reasoning. Table 4 shows that optimizing $\mathcal{L}_{\text{DR}}$ improves RM's reasoning ability, but BT modeling is still beneficial in equipping RM with abilities in general chatting as suggested in the "Chat-Hard" column, though its effect on reasoning may vary.

ULTRAINTERACT is compatible with other datasets like UltraFeedback and UltraSafety, and mixing these datasets can balance different RM abilities. Improving RM's capabilities in reasoning with ULTRAINTERACT does not sacrifice others, which indicates that ULTRAINTERACT can be a great ingredient for the training data mixture of reward models.

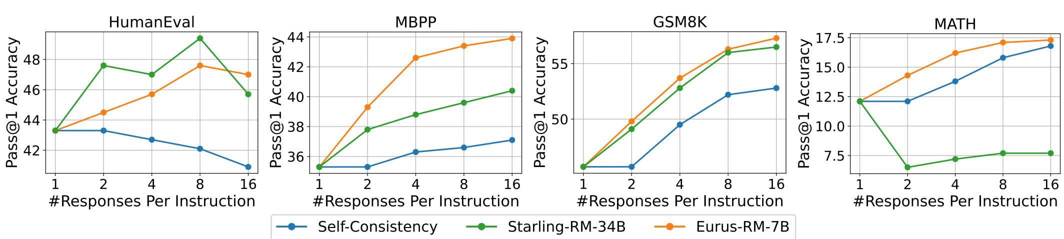

Figure 5: Results on reranking Mistral-7B-Instruct-v0.2's responses. Full results in Table 11.

5.2 BEST-OF-N AND MCTS DECODING

To further explore EURUS-RM-7B's potential in improving models' performance through reranking, we use it to rerank Mistral-7B-Instruct-v0.2's responses on HumanEval, MBPP, GSM8K, and MATH. We report the results of random sampling, self-consistency, and Starling-RM-34B as baselines. Finally, we examine the ability of EURUS-RM-7B to guide Mistral-7B-Instruct-v0.2 for MCTS decoding on math, compared to Starling-RM-7B. Due to the inference overhead, we only sample 500 samples from GSM8K and MATH for evaluation.

Results. Figure 5 and Table 5 presents some results with others in Appendix D.1.

EURUS-RM-7B improves LLMs' reasoning performance by a large margin through reranking. It consistently improves pass@1 accuracy across all tasks and performs better than 5× larger baseline Starling-RM-34B. Also, EURUS-RM-7B's reranking performance scales well with #responses per instruction, except for a slight decrease in HumanEval when increasing the response number from 8 to 16. In contrast, Starling-RM-34B suffers from a severe performance drop on HumanEval and it consistently hurts model accuracy on MATH.

EURUS-RM-7B can improve the performance of Mistral-7B-Instruct-v0.2 by 81.4% on GSM8K and 43.8% on MATH. According to Table 5, the policy model only achieves 25.8% and 6.4% on GSM8K and MATH respectively, but the performances are largely improved with RM-guided MCTS decoding. Compared to Starling-RM-7B, EURUS-RM-7B shows an advantage of 46.8% vs. 44.8% on GSM8K, and 9.2% vs. 6.6% on MATH.

Table 5: Mistral-7B-Instruct-v0.2 results with RM-guided MCTS decoding.

Model	GSM8K	MATH
Mistral-7B-Instruct-v0.2	25.8	6.4
+ Starling-RM-7B	44.8	6.6
+ EURUS-RM-7B	46.6	9.2

5.3 HOW DOES $\mathcal{L}_{\text{DR}}$ WORK IN REWARD MODELING?

Results in Table 4 have demonstrated the effectiveness of our proposed reward modeling objective. To figure out the working mechanism, we plot the reward patterns in reward modeling in Figure 6. As we can see, $\mathcal{L}_{\text{BT}}$ only optimize the reward margin, and the reward trend of chosen data is unstable while rewards of rejected data are positive for the most time. In contrast, $\mathcal{L}_{\text{DR}}$ forces the reward of chosen data to be positive and that of rejected data to be negative as expected. There is a consistent trend of increasing the reward of chosen data and decreasing that of rejected data, which also leads to a widening reward margin. When combining $\mathcal{L}_{\text{BT}}$ with $\mathcal{L}_{\text{DR}}$, the resulting pattern is more similar to using $\mathcal{L}_{\text{DR}}$ only. These results indicate the importance of the absolute value of rewards in reward model training.

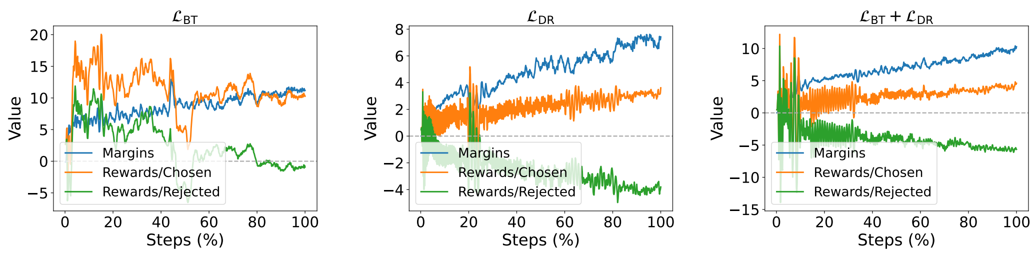

Figure 6: Reward pattern in reward modeling. The rewards of chosen data and margins increase regardless of $\mathcal{L}_{\text{DR}}$, but the rewards of rejected data decrease to be negative with regularization.

6 ANALYSIS

6.1 ARE PROPRIETARY MODELS NECESSARY FOR HIGH-QUALITY DATA CONSTRUCTION?

The current construction of ULTRAINTERACT largely adopts GPT models. However, using proprietary models raises concerns on the non-permissive license and the heavy burden in financial budgets, both of the which can be addressed by open-source models. Therefore, to ablate the use of proprietary models and examine how far we can go with open-source models for data synthesis, we supplement another version of ULTRAINTERACT by substituting all GPT responses with those generated by open-source models, **including EURUS itself**. We list the statistics, incluidng models used to sample responses and corresponding proportions, in Appendix F. We named the original GPT-involved version as ULTRAINTERACT-v1 and the open-source model generated as ULTRAINTERACT-v2. We retrain LLAMA-3-EURUS-8B with ULTRAINTERACT-v2 and union of v1 and v2. Results are presented in Table 6.

Surprisingly, LLAMA-3-EURUS-8B performances are greatly improved. Compared to v1, training on v2 improves model performance on both SFT and preference learning stage, and particularly, LLAMA-3-EURUS-8B-KTO (v2) successfully surpasses the official Llama-3-8b-Instruct model (Meta, 2024), which it previously failed to, indicating we can train strong models without distilling proprietary models. Further, trained on the mix of v1 and v2, LLAMA-3-EURUS-8B consistently outperforms Llama-3-8B-Instruct after preference learning. Notably, to the best of our knowledge, this is the first open recipe on Llama-3-8B that can outperform the official Llama-3-8B-Instruct model on reasoning.

6.2 ABLATION STUDY

Table 6: Performances of LLAMA-3-EURUS-8B trained with different versions of ULTRAINTERACT.

Model	Coding			Math					Reasoning	Ins-Following	Multi-Turn		Avg.
	HumanEval	MBPP	LeetC.	GSM-Plus	MATH	Theo.QA	SVAMP	ASDiv	BBH	IFEval	Code	Math	
Llama-3-8b-Instruct	59.8	60.4	**21.1**	49.1	28.3	15.0	73.7	75.9	**73.6**	**81.9**	25.0	**44.7**	50.7
LLAMA-3-EURUS-8B-SFT (v1)	51.2	57.9	17.2	50.7	32.0	21.3	82.2	83.7	72.4	47.1	18.4	24.5	46.6
+ DPO	43.9	50.1	11.7	45.3	26.8	21.4	54.1	67.5	71.3	56.7	21.3	39.2	42.4
+ KTO	51.8	58.1	15.6	54.8	34.2	24.9	80.1	86.7	71.7	50.6	26.5	37.4	49.4
+ NCA	50.6	60.4	15.6	55.2	34.8	25.4	79.9	87.5	71.7	56.2	21.3	36.3	49.6
LLAMA-3-EURUS-8B-SFT (v2)	55.5	60.2	17.8	54.4	37.7	24.6	**88.1**	85.0	73.2	49.9	18.4	29.7	49.5
+ DPO	57.9	53.9	12.2	49.1	37.5	**27.6**	78.6	76.9	73.1	55.6	22.8	**44.0**	49.1
+ KTO	59.1	55.9	**21.1**	60.1	**39.7**	25.9	86.9	86.6	73.4	47.5	22.1	38.8	51.4
+ NCA	56.7	52.6	15.6	56.8	36.2	24.1	88.0	84.1	**73.6**	51.0	21.3	36.3	49.7
LLAMA-3-EURUS-8B-SFT (v1 + v2)	54.9	**62.2**	16.7	54.4	38.0	25.0	86.4	85.9	72.5	**59.1**	21.3	30.4	50.6
+ DPO	53.0	54.4	18.3	58.8	37.0	27.5	83.1	**89.0**	72.2	51.0	25.0	42.1	51.0
+ KTO	**61.6**	59.1	15.6	61.2	39.0	26.8	86.8	88.2	72.1	47.7	**29.4**	39.2	**52.2**
+ NCA	53.7	60.9	12.8	**62.2**	38.1	25.4	87.3	87.7	72.5	49.0	25.0	35.5	50.8

Table 7: Ablation study of SFT data.

Model	Coding	Math	BBH	IFEval	Avg.
EURUS-7B-SFT	44.9	58.5	64.6	44.0	53.6
Ground-truth	33.9	46.1	64.4	42.9	44.0
Existing Data Only	31.2	33.5	65.3	43.6	37.0
ULTRAINTERACT Only	37.3	56.2	67.0	17.4	47.7

We study the impact of ULTRAINTERACT and other open-source alignment data on EURUS-7B-SFT's performance. We consider three settings: (1) **With original ground-truth answers,** which replaces the generated actions with ground-truth rationales and answers from the original datasets. If no rationales are available, we use those from ULTRAINTERACT. (2) **Existing data only.** (3)**ULTRAINTERACT only.** We evaluate with the same setting as §4 and report the averaged scores. See full results in Appendix E.

In Table 7, EURUS outperforms the "Grouth-truth" model on all tasks, confirming the advantage of ULTRAINTERACT's designs of divide-and-conquer and code-as-action patterns, in line with conclusions of concurrent work (Chen et al., 2024b; Wang et al., 2024). Training only on existing data without ULTRAINTERACT greatly hurts the reasoning performance, confirming the effectiveness of ULTRAINTERACT. Meanwhile, training only on ULTRAINTERACT suffers a performance drop except for BBH, especially in instruction following. We attribute the performance drop to a worse instruction-following ability. This suggests the necessity of mixing ULTRAINTERACT with other alignment data for better all-around supervised fine-tuning.

7 RELATED WORK

Open LLMs in Reasoning. Open-source LLMs have shown remarkable progress in building *specialists* that excel in mathematics reasoning (Luo et al., 2023a; Yue et al., 2023; Toshniwal et al., 2024) or coding abilities (Roziere et al., 2023; Wei et al., 2023; Guo et al., 2024a; Zheng et al., 2024). On the contrary, mastering general reasoning capabilities still challenges open models, while the most advanced ones (DeepSeek-AI, 2024; Bai et al., 2023; Touvron et al., 2023; Jiang et al., 2024) are well behind proprietary models. More, these cutting-edge open general-purpose models maintain their alignment recipes confidential, which further hinders the replication and development of open-source reasoning models.

Preference Learning for Reasoning. Aligning language models from human or AI preferences has emerged as a prevalent approach in the open-source community (Tunstall et al., 2023; Bai et al., 2023) with the proposal of DPO (Rafailov et al., 2023) and high-quality preference datasets (Cui et al., 2023; Zhu et al., 2023). Different from open-domain chatbots, preference learning is largely underexplored in complex reasoning. Recent research showed performance degradation when applying DPO on reasoning tasks, but some newly proposed algorithms demonstrated a positive effect (Ethayarajh et al., 2024; Chen et al., 2024a; Mitra et al., 2024; Shao et al., 2024b). However, a deep understanding of preference learning, specifically its efficacy on complex reasoning, is not yet established.

8 CONCLUSION

We strive to narrow the huge gap between open-source models and proprietary models from the perspective of alignment. Our work pushes the boundaries of open-source reasoning generalists by (1) releasing a high-quality multi-turn reasoning dataset ULTRAINTERACT with preference trees, (2) introducing EURUS-series LLMs which achieve new SOTA on challenging reasoning benchmarks and (3) providing insights on preference learning for reasoning through analysis, leading to new reward modeling objectives as well as a powerful reward model for reasoning.

REFERENCES

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In Proc. of NAACL-HLT, 2019.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. ArXiv preprint, abs/2108.07732, 2021.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. ArXiv preprint, abs/2309.16609, 2023.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, T. J. Henighan, Nicholas Joseph, Saurav Kadavath, John Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, Benjamin Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. ArXiv preprint, abs/2204.05862, 2022.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika, 39, 1952.

Huayu Chen, Guande He, Hang Su, and Jun Zhu. Noise contrastive alignment of language models with explicit rewards. ArXiv preprint, abs/2402.05369, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde 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, Josh 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, 2021.

Wenhu Chen, Ming Yin, Max W.F. Ku, Yixin Wan, Xueguang Ma, Jianyu Xu, Tony Xia, Xinyi Wang, and Pan Lu. Theoremqa: A theorem-driven question answering dataset. ArXiv preprint, abs/2305.12524, 2023.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. volume abs/2403.12881, 2024b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. volume abs/2110.14168, 2021.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. ArXiv preprint, abs/2310.01377, 2023.

DeepSeek-AI. Deepseek llm: Scaling open-source language models with longtermism. ArXiv preprint, abs/2401.02954, 2024.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In Conference on Empirical Methods in Natural Language Processing, 2023.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. ArXiv preprint, abs/2402.01306, 2024.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. In Proceedings of the International Conference on Machine Learning, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. Transactions of the Association for Computational Linguistics, 9, 2021.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu 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. ArXiv preprint, abs/2401.14196, 2024a.

Yiju Guo, Ganqu Cui, Lifan Yuan, Ning Ding, Jiexin Wang, Huimin Chen, Bowen Sun, Ruobing Xie, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Controllable preference optimization: Toward controllable multi-objective alignment. ArXiv preprint, abs/2402.19085, 2024b.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021b.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. ArXiv preprint, abs/2310.06825, 2023a.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. 2024.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In Annual Meeting of the Association for Computational Linguistics, 2023b.

Nathan Lambert, Valentina Pyatkin, Jacob Daniel Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Raghavi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hanna Hajishirzi. Rewardbench: Evaluating reward models for language modeling. 2024.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. Generative judge for evaluating alignment. ArXiv preprint, abs/2310.05470, 2023a.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. ArXiv preprint, abs/2402.19255, 2024.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. volume abs/2312.14852, 2023b.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. volume abs/2203.07814, 2022.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. https://https://huggingface.co/Open-Orca/OpenOrca, 2023.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. 2023.

Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa: Surpassing gpt-4 on conversational qa and rag. 2024. URL https://api.semanticscholar.org/CorpusID:267035133.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In Proceedings of ICLR, 2023.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. ArXiv preprint, abs/2308.09583, 2023a.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct, 2023b.

Meta. The llama 3 herd of models. ArXiv, 2024.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In Proc. of ACL, 2020.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In Proc. of ACL, 2022.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agrawal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. Orca 2: Teaching small language models how to reason. ArXiv preprint, abs/2311.11045, 2023.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. ArXiv preprint, abs/2402.14830, 2024.

OpenAI. Gpt-4 technical report, 2023.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In Proc. of ACL, 2015.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021.

Cheng Qian, Chi Han, Yi Ren Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In Conference on Empirical Methods in Natural Language Processing, 2023.

Yujia Qin, Shi Liang, Yining Ye, Kunlun Zhu, Lan Yan, Ya-Ting Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Marc H. Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. ArXiv preprint, abs/2307.16789, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. ArXiv preprint, abs/2305.18290, 2023.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. ArXiv preprint, abs/2308.12950, 2023.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. ArXiv preprint, abs/2402.03300, 2024b.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. ArXiv preprint, abs/2210.09261, 2022.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. arXiv preprint arXiv: Arxiv-2402.10176, 2024.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. ArXiv preprint, abs/2307.09288, 2023.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. Zephyr: Direct distillation of lm alignment. ArXiv preprint, abs/2310.16944, 2023.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. Openchat: Advancing open-source language models with mixed-quality data. ArXiv preprint, abs/2309.11235, 2023a.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. ArXiv preprint, abs/2309.10691, 2023b.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. ArXiv preprint, abs/2402.01030, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. ArXiv preprint, abs/2201.11903, 2022.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Source code is all you need, 2023.

Martin Weyssow, Aton Kamanda, and Houari Sahraoui. Codeultrafeedback: An llm-as-a-judge dataset for aligning large language models to coding preferences. ArXiv preprint, abs/2403.09032, 2024.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Yang Wang. Perils of self-feedback: Self-bias amplifies in large language models. ArXiv preprint, abs/2402.11436, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Proc. of EMNLP, 2018.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset requiring logical reasoning. In Proc. of ICLR, 2020.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Ren Fung, Hao Peng, and Heng Ji. Craft: Customizing llms by creating and retrieving from specialized toolsets. ArXiv preprint, abs/2309.17428, 2023.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. ArXiv preprint, abs/2309.05653, 2023.

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

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. ArXiv preprint, abs/2402.14658, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. ArXiv preprint, abs/2311.07911, 2023.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, 2023.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. arXiv preprint arXiv:2406.11931, 2024.

Table 8: ULTRAINTERACT covers a diverse set of datasets spanning three tasks.

Task	Datasets
Math	GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), MathQA (Amini et al., 2019), NumGlue (Mishra et al., 2022), TabMWP (Lu et al., 2023)
Coding	CodeContest (Li et al., 2022), TACO (Li et al., 2023b), WikiTableQuestions (Pasupat & Liang, 2015), Magicoder-Evol-Instruct (Luo et al., 2023b; Wei et al., 2023)
Logic	ReClor (Yu et al., 2020), HotpotQA (Yang et al., 2018), StrategyQA (Geva et al., 2021)

A Additional Details in UltraInteract Construction

A.1 Dataset Details

Math. We adopt **GSM8K** (Cobbe et al., 2021), **MATH** (Hendrycks et al., 2021b), **MathQA** (Amini et al., 2019), and **NumGLUE** (Mishra et al., 2022)for mathematic reasoning, and include **TabMWP** (Lu et al., 2023) for tabular processing. We retain all the instructions for all datasets except MathQA, NumGLUE, and TabMWP. MathQA divides problems into different categories according to the topics and annotates the formula that indicates the pattern needed to solve each problem. We apply stratified sampling to sample at most five problems for each pattern and prioritize the problems that come from the long-tail category. Numglue contains eight different reasoning tasks and we discard Task 5 (Reading Comprehension + Explicit Numerical Reasoning), Task 6 (Reading Comprehension + Implicit Numerical Reasoning), and Task 7 (Quantitative NLI) due to the simplicity Mishra et al. (2022). For TabMWP, we only keep the questions with difficulty levels 4 and 5 since the rest are too easy for current state-of-the-art models.

Code. We focus on programming with Python for the simplicity of integration of the interpreter. We use **CodeContest** (Li et al., 2022) and **TACO** (Li et al., 2023b), two competition-level coding datasets collected from various online platforms. We filter out the overlapped questions. Note that part of the questions in TACO only contain ground-truth solutions and do not contain test cases for evaluation, hence we apply GPT-4 to generate 12 test case inputs (4 basic inputs, 4 edge cases, and 4 large numbers) for each question and then execute the ground-truth solution snippets to produce outputs. Given that the two datasets mainly focus on competition problems that may deviate from real-world daily uses, we exclusively adopt **Magicoder-Evol-Instruct** (Luo et al., 2023b; Wei et al., 2023), the only dataset in our selection that does not contain test cases or ground-truth solutions. We employ GPT-4 Turbo to judge the correctness of generated code during interaction, and therefore we do not use this dataset for preference learning since we cannot rigorously construct pairs of correct and incorrect actions limited by the evaluation reliability. We also include **WikiTableQuestions** (Pasupat & Liang, 2015) for table processing with code.

Logical Reasoning. we use the multi-hop reasoning datasets **HotpotQA** (Yang et al., 2018) and **StrategyQA** (Geva et al., 2021), and the logical reasoning dataset **ReClor** (Yu et al., 2020). We follow the setting of Wang et al. (2023b) and convert HotpotQA to a generation task, removing the contexts and requiring LLMs to search relevant information using Wikipedia API.

A.2 Details on Preference Tree Construction

Models Adopted for Incorrect Action Sampling. We randomly sample one model from Mistral-7B-Instruct-v0.2, DeepSeek-Coder-33B-Instruct, Mixtral-8x7B-Instruct, and DeepSeek-LLM-67B-Chat to generate one incorrect action to pair with each correct one.

Correct Action Generation Based on Ground Truth Annotations.

We adopt GPT-3.5 Turbo as the generator to generate correct actions based on ground truth considering the instruction-following ability. We provide different access to the ground truth information for different tasks, specifically: (1) For coding, where test cases are black boxes to reference solutions, we provide full access to the solution codes. The actor model will add step marks and corresponding explanations to the ground-truth code to make it easier to understand, or further refine the code for optimization. (2) For tool-free math problems, to avoid the actor model directly copying the answers to pass the correctness checking, we mask the answer numbers in the rationale before providing it to LLMs. This approach can better ensure response quality since it encourages LLMs to generate

Table 9: Stats breakdown

Task	Dataset	w/ Tool?	# Prompts	# Pairs	# Correct Answers.	Avg. Length	Human Annotation	
							Has Answer?	Has Rationale?
Math	GSM8K	✓	4,522	10,277	17,392	1,746.7	✓	✓
		✗	7,257	10,879	15,752	823.3	✓	✓
	MATH	✓	7,474	22,905	34,667	1,189.0	✓	✓
		✗	7,471	25,765	36,005	1,735.0	✓	✓
	MathQA	✓	7,552	15,079	20,328	2,338.5	✓	✓
		✗	7,159	17,743	22,500	1,916.3	✓	✓
	NumGLUE	✓	3,020	3,601	5,717	1,474.6	✓	✗
		✗	2,835	2,987	4,273	1,056.1	✓	✗
	TabMWP	✓	3,117	4,135	6,083	842.6	✓	✗
Coding	CodeContest	-	8,167	44,319	44,666	2,061.7	✓	✓
	TACO	-	9,016	50,877	58,191	2,143.5	✓	✓
	WikiTableQuestions	-	1,401	1,544	1,738	1,794.8	✓	✗
	Magicoder-Evol-Instruct	-	10,374	0	10,238	687.1	✗	✗
Logic	Reclor	✗	4,467	7,958	7,231	1,266.7	✓	✗
	HotpotQA	✓	1,182	1,009	1,230	1,333.2	✓	✗
	StrategyQA	✓	904	741	968	1,256.2	✓	✗

responses with complete reasoning chains with each step clearly marked. (3) For program-enhanced math reasoning, we first translate the textual rationale into code. Then, we either directly provide it to the actor model to generate plans, or ask the actor model to convert the code into modularization programming and then make plans to create tools to solve problems.

A.3 Data Decomtamination

We conduct careful decontamination. Firstly, for LeetCode, we apply the Exact Substring Matching Algorithm[2] to compare with each instruction in the UltraInteract and find no overlaps. For others, we perform 8-gram exact matching to compare UltraInteract instructions with test sets of the same task. We remove those instructions that overlap 8 grams with any test sample.

A.4 Detailed Statistics

In total, UltraInteract has 86K instructions and 220K action pairs. The Total # Pairs does not equal Total # Turns in UltraInteract, since we fail to generate sufficient correct actions for every incorrect action in multi-turn trajectories mainly due to a lack of sufficient ground truth annotations. The total # pairs may not equal # correct answers, either, because it is also difficult and unnecessary to sample incorrect actions for the correct ones for some simple instructions. We present the specific information for each dataset. In particular, we list information on human annotation in each dataset, which plays an important role in correct action generation (§2.3 and Appendix A.2). All three steps of correct action sampling methods mentioned in §2.3 can be applied to datasets that have rationales, while for datasets only containing answers, only the first two steps are applicable. We do not apply any of the three-step methods to generate correct answers for Magicoder, the only dataset without any human annotation, to construct preference pairs.

B Additional Details on Training Eurus Models

Supervised Fine-Tuning. We finetune base models for 1 epoch with a 2e-5 learning rate and 0.1 warmup ratio using a cosine scheduler. For Eurus-7B, we mix 32K UltraChat (Ding et al., 2023), 30K ShareGPT[3], and 50K OpenOrca(Lian et al., 2023). For Eurus-70B, we mix 63K UltraChat, 30K ShareGPT, and 70K OpenOrca. For Llama-3-Eurus-8B, we mix 50k UltraChat, 18k OpenChat (Wang et al., 2023a), TheoremQA decomtaminated OpenHermes[4], and the entire ChatQA (Liu et al., 2024) and CodeAct (Wang et al., 2024).

[2] https://github.com/bigcode-project/bigcode-dataset/tree/main/decontamination

[3] https://huggingface.co/datasets/openchat/openchat_sharegpt4_dataset

[4] https://huggingface.co/datasets/teknium/OpenHermes-2.5

Preference Learning. For hyperparameters, all β is set to 0.1, and λ_+/λ_- in KTO is set to 1.33 as recommended. We finetune models for 1 epoch with a 5e-7 learning rate and 0.1 warmup ratio using a cosine scheduler.

Reward Modeling. We train RM for 1 epoch with lr=1e-5 learning rate. We also use a cosine scheduler with a warmup ratio of 0.1.

Regarding pair augmentation, we scale up the pairs by matching every correct action for each instruction with one incorrect action of other turns. This leads to NxN pairs of single-turn actions for a trajectory of depth N. We remove the action pairs consisting of nodes at the same turn, as they are already part of the multi-turn trajectory pairs we included. Next, to avoid overfitting on the training set, we only select instructions with NxN $\leq$ 10, and for these instructions, we randomly sample at most 9 pairs with each action occurring no more than 3 times. This leads to an augmentation of 240k single-turn action pairs.

C ADDITIONAL EVALUATION RESULTS OF EURUS

Detailed Setup in §4. For math, we test both textual reasoning and program-enhanced settings and report the best performance of the two. All evaluations are conducted in 0-shot CoT with two exceptions: BBH uses 3 shots and IFEval does not use CoT. For MINT, we select MATH, TheoremQA, and MMLU-math from "reasoning" as a new "math" split. We also evaluate 5-shot MMLU (Hendrycks et al., 2021a) for STEM knowledge and MT-Bench (Zheng et al., 2023) for conversation abilities to study whether EURUS needs to trade off other capabilities for reasoning.

Results. Results are shown in Table 10.

On MMLU, EURUS outperforms baselines dedicated to coding and math, and achieves comparable or higher results than Mistral-Instruct-v0.2 and Mixtral-8x22B-Instruct-v0.1, the official aligned versions of our base model built by their authors. Compared to general-purpose baseline models, EURUS-7B and LLAMA-3-EURUS-8B achieve comparable performance with the top-performance OpenChat and Starling-LM, and EURUX-8X22B also matches the top level of performance among other general-purpose models.

On MT-Bench, we report baseline numbers from the official leaderboard[5] if available. EURUS matches the performance of mainstream open-source general-purpose models, and EURUX-8X22B further surpasses the score of GPT-3.5 Turbo.

Table 10: MMLU and MT-Bench.

Model	MMLU	MT-Bench
~7B		
Mistral-7B-Instruct-v0.2	58.9	7.60
Zephyr-7B-β	59.7	7.34
OpenChat-3.5-1210	63.4	7.81
Starling-LM-7B-α	64.0	**8.09**
Magicoder-S-DS-6.7B	37.1	4.21
OpenCI-DS-6.7B	37.2	4.06
MAmmoTH-7B-Mistral	56.2	4.25
WizardMath-7B-v1.1	60.3	5.62
OpenMath-Mistral-7B	58.3	2.69
EURUS-7B-SFT	61.8	7.15
+ DPO	62.4	7.38
+ KTO	62.2	7.38
+ NCA	62.2	7.38
LLAMA-3-EURUS-8B-SFT	64.6	6.82
+ DPO	64.8	7.44
+ KTO	**64.9**	7.27
+ NCA	64.7	7.14
~40B		
Mixtral-8x7B-Instruct	**70.3**	**8.30**
DeepSeek-Coder-33B-Ins	40.2	3.83
~70B		
CodeLLaMA-70B-Instruct	55.1	5.12
DeepSeek-LM-67B-Chat	72.3	8.08
QWen1.5-72B-Chat	72.9	8.61
OpenCI-CL-70B	52.4	5.67
OpenMath-CL-70B	60.2	2.29
WizardLM-2-8x22B	77.0	**9.08**
Mixtral-8x22B-Instruct-v0.1	**77.6**	8.66
EURUX-8X22B-SFT	75.9	8.20
+ KTO	75.9	8.58
+ NCA	75.6	8.46
Proprietary Models		
GPT-3.5 Turbo	70.0	7.94
GPT-4	**86.4**	**8.96**

D DETAILED RESULTS ON REWARD MODELING

D.1 ADDITIONAL RESULTS ON RERANKING

We present the full results on reranking in Table 11, where the conclusions are consistent with those drawn from §D: (1) Our reward models always achieve the highest accuracy on all test sets across different N, except when N=2 on HumanEval. (2) Both $\mathcal{L}_{\text{BT}}$ and $\mathcal{L}_{\text{DR}}$ consistently help improve reranking performance on three test sets except for HumanEval, where removing either of the objectives can prevent the accuracy from dropping when increasing N from 8 to 16. (3) Modeling

[5] `https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard`

Table 11: Detailed results of reranking Mistral-Instruct-v0.2's responses on coding and math.

Datasets	HumanEval				MBPP				GSM8K				MATH			
N	2	4	8	16	2	4	8	16	2	4	8	16	2	4	8	16
Random	41.5	39.0	40.2	39.6	33.1	33.6	34.3	30.1	45.0	43.1	44.5	40.2	11.5	11.3	10.0	8.5
Top Logits	43.3	43.3	43.3	43.3	35.3	35.3	35.3	35.3	45.7	45.7	45.7	45.7	12.1	12.1	12.1	12.1
Self-Consistency	43.3	42.7	42.1	40.9	35.3	36.3	36.6	37.1	45.7	49.5	52.2	52.8	12.1	13.8	15.8	16.8
Starling-RM-34B	**47.6**	**47.0**	**49.4**	45.7	37.8	38.8	39.6	40.4	49.1	52.8	56.0	56.5	6.5	7.2	7.7	7.7
EURUS-RM-7B	44.5	45.7	47.6	47.0	39.3	**42.6**	**43.4**	**43.9**	**49.8**	53.7	56.3	57.3	14.3	16.2	17.1	17.3
w/o $\mathcal{L}_{\mathcal{DR}}$	45.7	44.5	46.3	50.0	39.3	42.4	42.4	42.1	49.4	53.2	55.4	56.3	14.2	16.1	17.0	16.9
w/o $\mathcal{L}_{\mathcal{BT}}$	45.1	44.5	47.0	48.2	38.6	40.6	39.6	40.1	49.1	52.5	55.2	57.8	14.3	16.3	17.2	17.1
w/o US	45.7	**47.0**	**49.4**	**50.6**	**39.3**	41.1	41.4	42.9	49.4	**53.8**	**57.4**	**58.7**	**14.5**	**16.6**	17.2	**17.5**
w/o UF + US	43.9	43.3	47.0	46.3	36.3	38.1	36.6	35.3	49.4	52.3	54.6	57.2	14.3	16.5	**17.4**	17.4
Pass@N	62.8	73.8	88.4	92.7	42.4	48.1	52.6	58.6	54.9	64.1	73.2	80.4	16.9	22.7	28.9	35.5

Table 12: Ablation Study.

Model	Coding			Math					Reasoning	Ins-Following	Avg.
	HumanEval	MBPP	LeetCode	GSM8K	MATH	TheoremQA	SVAMP	ASDiv	BBH	IFEval	
EURUS-7B-SFT	**55.5**	**59.1**	**20.0**	**73.7**	**32.6**	20.0	**82.2**	**84.1**	64.6	**44.0**	**53.6**
Ground-Truth	46.3	46.4	8.9	62.2	15.0	9.6	75.1	68.8	64.4	42.9	44.0
Exisiting Data Only	38.4	44.1	11.1	45.3	10.8	9.3	52.7	49.4	65.3	43.6	37.0
ULTRAINTERACT Only	46.3	50.1	15.6	67.6	30.9	**20.1**	80.4	82.0	**67.0**	17.4	47.7

safety hurts reranking performance in reasoning. When removing UltraSafety from the training data, the RM achieves higher accuracies than EURUS-RM-7B except on MBPP.

E DETAILED ABLATION RESULTS

We present the full results of §7 in Table 12, with detailed metrics on all coding and math datasets.

F STATISTICS OF ULTRAINTERACT-V2

We use EURUS-7B-KTO, EURUX-8X22B-NCA, Llama-3-8/70B-Instruct, Llama-3.1-8/70B-Instruct, DeepSeek-Chat-V2 Zhu et al. (2024), DeepSeek-Coder-V2 (Guo et al., 2024a), DeepSeek-Math-RL Shao et al. (2024a), etc, to construct the ULTRAINTERACT-v2. Particularly, the proportion of EURUS responses is 26.4% in the SFT split and 28.2% in the preference split, while the Llama responses take the share of 59.0% and 56.7% in the SFT split and preference split respectively. We present the detailed statistics of ULTRAINTERACT-v2 in Table 13 and Table 14.

G DATA EXAMPLE

We present an example for SFT data in Table 15 and an example for preference learning in Table 16.

H ADDITIONAL ABLATION EXPERIMENT RESULTS

We present the ablation study on prefernce data mixture in this section. We perform KTO on Llama-3-Eurus-8B-SFT with different combinations of UltraFeedback and UltraInteract and the results are presented in Table 17. From the results, we see that training only on UltraInteract leads to higher overall reasoning performances, which are mainly credited to the multi-turn interaction ability, demonstrating the superiority of the tree structure of our data. However, we also observe a lower MT-Bench score compared to training solely on UltraFeedback. Nevertheless, this can be mitigated without hurting reasoning performances by mixing these two datasets together, which indicates that our data is compatible with other datasets, consistent with our conclusions on reward modeling.

Table 13: Some statistics of ULTRAINTERACT-v2.

Task	Type: w/ Interaction?	Type: w/ Tool?	# Instructions	# Turns per Traj. T1	T2	T3	T4	T5	# Tokens per Traj.	Avg. # Traj per Ins.	Total # Pairs	# Correct Answers
Math	✓	✓	20,935	8,018	6,149	2,413	1,091	3,264	1,374.2	1.0	61,738	80,982
	✗	✓	1,279	6,723	-	-	-	-	423.0	5.3	2,055	6,723
	✓	✗	21,474	10,117	4,707	1,263	589	4,798	1,155.7	1.0	72,944	143,587
	✗	✗	634	4,876	-	-	-	-	396.0	7.7	2,172	4,876
Coding	✓	-	9,498	4,323	2,057	942	459	1,717	1,429.4	1.0	71,900	155,429
	✗	-	2,877	9,824	-	-	-	-	591.0	3.4	5,990	9,824
Total	-	-	56,697	43,881	12,913	4,618	2,139	9,779	894.9	3.2	216,799	410,421

Table 14: Stats breakdown for ULTRAINTERACT-v2

Task	Dataset	w/ Tool?	# Prompts	# Pairs	# Correct Answers.	Avg. Length	Human Annotation: Has Answer?	Human Annotation: Has Rationale?
Math	GSM8K	✓	4,438	14,616	17,797	642.4	✓	✓
		✗	6,997	10,708	11,893	367.6	✓	✓
	MATH	✓	6,197	20,056	27,217	681.8	✓	✓
		✗	6,799	46,720	115,924	531.7	✓	✓
	MathQA	✓	6,046	19,315	22,062	653.8	✓	✓
		✗	5,898	14,525	16,237	570.3	✓	✓
	NumGLUE	✓	2,436	4,241	9,113	570.4	✓	✗
		✗	2,414	3,163	4,409	409.0	✓	✗
	TabMWP	✓	3,097	5,565	11,516	507.1	✓	✗
Coding	CodeContest	-	5,788	39,713	76,272	1,291.7	✓	✓
	TACO	-	6,459	37,317	86,844	1,062.9	✓	✓
	Codeforces	-	128	860	2,137	1,480.9	✓	✓

Table 15: An example from the SFT data.

Task: Coding

Instruction: Write Python code to solve the task.
Mike has a string s consisting of only lowercase English letters. He wants to change exactly one character from the string so that the resulting one is a palindrome.
A palindrome is a string that reads the same backward as forward, for example strings z, aaa, aba, abccba are palindromes, but strings codeforces, reality, ab are not.
Input
The first and single line contains string s.
Output
Print YES (without quotes) if Mike can change exactly one character so that the resulting string is palindrome or NO (without quotes) otherwise.
Response:
Step 1: This problem asks us to determine if it is possible to change exactly one character in the string to make it a palindrome.
Step 2: We can start by iterating through the characters in the string and counting the number of differences between corresponding characters from the beginning and end of the string.
Step 3: If the length of the string is odd and there is only one difference, or if the length is even and there is exactly one difference, then it is possible to change exactly one character to make the string a palindrome.
Step 4: We need to consider edge cases such as a single character string or an empty string. “‘python

```
# Step 5: Take the input string
s = input()
# Step 6: Calculate the length of the string
n = len(s)
# Step 7: Initialize a counter to keep track of the number of differences
cnt = 0
# Step 8: Iterate through the first half of the string and count the number of differences between corresponding characters from the beginning and end of the string.
for i in range(n//2):
    if s[i] != s[n-i-1]:
        cnt += 1
# Step 9: If the length of the string is odd and there are no differences, or if the length is even and there is exactly one difference, print ”YES” (without quotes)
if (n % 2 == 1 and cnt == 0) or (cnt == 1):
    print('YES')
# Step 10: Otherwise, print ”NO” (without quotes)
else:
    print('NO')
```

Table 16: An example from the Preference data.

Task:Math

Trajectory:

[{
from: user,
value: Tool available:[1] Python interpreter. When you send a message containing Python code to python, it will be executed in a stateful Jupyter notebook environment. Solve the following math problem step-by-step. Simplify your answer as much as possible. Simplify $(-3-2i)-(1+4i)$.
},
{
from: assistant,
value: Step 1: To simplify the expression $(-3-2i)-(1+4i)$, we need to distribute the negative sign to both terms inside the parentheses. Step 2: We can then combine like terms by adding the real parts and the imaginary parts separately. # Step 3: Distribute the negative sign to both terms inside the parentheses. (-3) - (-1) + (-2i) - (-4i) # Step 4: Combine like terms by adding the real parts and the imaginary parts separately. -3 + 1 + (-2i + 4i)
},
{
from: user,
value: Out: Cell In[1], line 2 (-3) - (-1) + (-2i) - (-4i) SyntaxError: invalid syntax. Your answer is wrong. Your thought process for simplifying the given complex numbers is correct. However, you made a syntax error in the code. In Python, the multiplication operation must be explicitly written out with an asterisk (*), even for complex numbers. Remember, the imaginary unit in Python is denoted as j and not i. Try correcting the syntax and calculate again.
},
{
from: assistant,
value: Step 1: To simplify the expression $(-3-2i)-(1+4i)$, we need to distribute the negative sign to both terms inside the parentheses. Step 2: We can then combine like terms by adding the real parts and the imaginary parts separately. # Step 3: Distribute the negative sign to both terms inside the parentheses. (-3) - (-1) + (-2j) - (-4j) # Step 4: Combine like terms by adding the real parts and the imaginary parts separately. -3 + 1 + (-2j + 4j)
},
{
from: user,
value: Out: (-2+2j). Your answer is wrong. Your thought process for simplifying the given complex numbers is on the right track. However, you made an arithmetic error when adding the real parts and the imaginary parts. You should have combined the like terms correctly to get the right answer -4 and -6j respectively. Please try again with the correct arithmetic calculations.
}]

Chosen:

Step 1: Split the expression into real and imaginary terms.
Step 2: Subtract the real terms from each other.
Step 3: Subtract the imaginary terms from each other.
Step 4: Combine the results of the subtraction to get the simplified form of the expression.
# Step 5: Split the expression into real and imaginary parts and then subtract them.
real_part = -3 - 1 parts
imaginary_part = -2 - 4
# Step 6: Combine the real and imaginary parts to get the result.
result = f{real_part}{imaginary_part:+d}i
print(result)

Rejected:

Step 1: To simplify the expression $(-3-2i)-(1+4i)$, we need to distribute the negative sign to both terms inside the parentheses.
Step 2: We can then combine like terms by adding the real parts and the imaginary parts separately.
Step 3: Distribute the negative sign to both terms inside the parentheses.
(-3) - 1 - (-2j) - 4j
Step 4: Combine like terms by adding the real parts and the imaginary parts separately.
-3 - 1 - 2j - 4j

Table 17: Additional experiment results on preference learning with different compositions of UltraFeedback and UltraInteract.

Model	Coding	Math	BBH	IFEval	Multi-Turn	Avg.	MT-Bench
Llama-3-Eurus-8B-SFT	42.11	53.97	72.40	47.10	21.45	46.60	6.8
+KTO (UF + UI)	41.84	56.14	71.70	50.60	31.92	49.40	7.3
+KTO (Only UF)	44.60	56.26	72.00	50.10	24.03	48.80	7.5
+KTO (Only UI)	40.70	55.86	71.70	50.60	34.49	49.40	7.2