Simultaneous Multi-objective Alignment Across Verifiable and Non-verifiable Rewards

Yiran Shen[1], **Yu Xia**[1], **Jonathan Chang**[2], **Prithviraj Ammanabrolu**[1,3]

[1] UC San Diego [2] Databricks [3] NVIDIA

{jes038, yux078, prithvi}@ucsd.edu, j.chang@databricks.com

Abstract

Aligning large language models to human preferences is inherently multidimensional, yet most large language model alignment pipelines reduce diverse human preferences to a single objective. We address simultaneous alignment across domains with verifiable rewards, non-verifiable subjective preferences, and complex interactions. Our framework unifies process reward model (PRM) training across these settings, applies **M**ulti-**A**ction-**H**ead **DPO** (MAH-DPO) with a vectorized reward to capture multiple objectives, and enables fine-grained user control at inference time. Our experiments show improved multi-objective performance with reduced trade-offs and enhanced controllability.

1 Introduction

The success and widespread deployment of large language models (LLMs) has enabled AI assistance in tasks from mathematical problem solving to educational tutoring [6, 18, 19, 36, 41, 42]. Such applications often demand that models meet multiple objectives simultaneously, showing that alignment to human preferences is inherently *multi-dimensional* [3, 4, 29]. We focus on three categories: *verifiable rewards* (e.g., mathematical accuracy), *non-verifiable preferences* (e.g., helpfulness, honesty, truthfulness), and *interactive scenarios* (e.g., multi-turn AI tutoring).

Current pipelines struggle to represent such structure. RLHF [10, 42] reduces comparisons to a scalar reward, and DPO [44] optimizes a single preference axis. Both collapse rich feedback into one dimension, discarding trade-off information. Multi-objective variants based on linear scalarization [17, 24, 31, 57, 71] or post-hoc merging of specialized models [27, 45] are costly and require retraining to change objectives or weights. Test-time steering with reward models offers adaptability but often suffers from granularity mismatch between reward definition and generation decisions [13, 28, 30, 62]. Process reward models (PRMs) mitigate this by providing step-level signals [23, 34, 38, 54, 61], but most work focuses on verifiable domains where intermediate steps are checkable [69, 70]; extending PRMs to non-verifiable settings remains challenging.

We present a framework that addresses these limitations. **First**, we standardize PRM training across verifiable and non-verifiable domains. For verifiable settings, we combine rollouts with hindsight credit assignment. For non-verifiable settings, we use majority voting evaluation, direct step judgment, and step reward approximation, chosen by task properties. **Second**, we introduce **M**ulti-**A**ction-**H**ead **DPO** (MAH-DPO), which retains the multi-dimensional structure by placing lightweight, objective-specific action heads on a shared backbone and optimizing head-wise DPO losses while updating shared parameters across dimensions. **Third**, we use PRM-guided decoding with continuing hidden state to provide fine-grained inference-time user control. Across math reasoning, human value alignment, and multi-turn AI tutoring, this combination improves multi-objective performance, reduces cross-objective trade-offs, and enables flexible inference-time control.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

2 Background

To understand the challenges and opportunities in multi-objective alignment, we examine three representative domains.

Mathematics. Mathematics represents a typical verifiable domain where ground truth can be automatically determined with datasets such as GSM8K [11], MATH [22], GaoKao [68], and OlympiadBench [21]. The verifiable nature of mathematical correctness enables automatic reward assignment at both outcome and process levels. Recent work has demonstrated the effectiveness of process reward models [34, 51, 54] that provide step-by-step supervision to validate intermediate reasoning steps. Mathematical problem-solving can also involve dimensions beyond accuracy, including explanation clarity for diverse user expertise levels and pedagogical engagement in practical applications.

Human Values. Unlike mathematical correctness, human values include a broad range of subjective preferences that cannot be automatically verified, including aspects such as helpfulness, harmlessness, and honesty [3–5, 42]. These qualities require human judgment and are subjective, context-dependent, and sometimes conflicting. Recent work such as HelpSteer [55, 56] and UltraFeedback [12] provides multi-dimensional annotations and reference comparisons across multiple criteria including helpfulness, coherence, and truthfulness. The challenge lies in the subjectivity and multi-dimensionality of human preferences, while the lack of automatic verification makes it difficult to provide more fine-grained supervision.

Interactive AI Tutoring. Interactive AI tutoring represents another challenging domain that combines objective and subjective evaluation within multi-turn dialogues, where success depends not only on correctness but also on pedagogical effectiveness, engagement, and scaffolding strategies. Datasets in this domain include educational dialogue corpora [7, 39, 50] and socratic questioning collections [2, 14, 48]. Unlike static domains, the quality of a tutor's response should be evaluated based on its impact on subsequent student responses and learning trajectories. We provide an example AI tutoring dialogue in Appendix K.

3 Process Reward Model Training

3.1 Verifiable Domains

For tasks with objective correctness criteria, e.g., math, we augment the step-level supervision with outcome signals with a value target estimator to train PRMs that both validate current intermediate step and predict future correctness. Given a trajectory $y_{1:N} = (y_1, y_2, \ldots, y_N)$, the step level reward signals textual validity and local coherence at step y_t [34, 38]. Labels commonly come from multi stage sampling and annotation [34, 54, 61], for example Math Shepherd [54] marks a step correct if any completion reaches a correct final answer. We add hindsight relabeling [1, 20]: from each step y_t, roll out to $y_{t+1:} = (y_{t+1}, \ldots, y_n)$ to get a terminal reward $z \in \{0, 1\}$, blend it with the annotated r_t, average over M rollouts, and train with mean squared error:

$$\tilde{r}_t = r_t + \gamma^{n-t} z, \qquad V_t^{\text{target}} = \frac{1}{M} \sum_{m=1}^{M} \tilde{r}_t^m, \qquad \mathcal{L}_{\text{PRM}} = \mathbb{E}_{t, y_{1:t}} \left[\left(p_t - V_t^{\text{target}} \right)^2 \right]. \qquad (1)$$

Here $\gamma \in (0, 1)$ discounts by temporal distance, which helps the PRM score local reasoning and forecast final correctness.

3.2 Non-verifiable Domains

For domains lacking objective correctness measures, we adapt our PRM training framework based on the availability of clear process structure and rollout difficulty.

Case A: Clear Process Structure with Efficient Rollout. When the task has clearly-defined intermediate steps that can be meaningfully evaluated, e.g. engagement in math reasoning process, we employ a rollout-based labeling strategy similarly to our verifiable domain approach. We first calibrate an LLM-as-Judge J using a few human annotated ratings $\hat{R}$ to approximate the expected human judgment, $J(y_{1:t}) \approx \mathbb{E}[\hat{R}]$. Then we sample M completions from each step y_t and evaluate the resulting full trajectories using our calibrated LLM-as-Judge J. We label the step y_t as positive when the majority of completions are judged as positive by J:

$$r_t \;=\; \mathbf{1}\Bigg[\Big(\frac{1}{M}\sum_{m=1}^{M}\mathbf{1}_{\text{positive}}\big[\big(J\big(y_{1:t}, y^{m}_{t+1:n}\big)\big)\big]\Big) \;>\; \tfrac{1}{2}\Bigg]. \tag{2}$$

This majority voting criterion reflects the inherent subjectivity in non-verifiable domains, where a reasoning step's quality is measured by its tendency to lead to generally acceptable outcomes rather than definite correctness.

Case B: Clear Process Structure with Costly Rollout. When generating rollouts is costly or difficult, for example multi-turn dialogue which requires real user interactions, we directly query the LLM-as-Judge J on observed trajectory prefixes to otain the training label: $r_t = J(y_{1:t})$. This approach trades the robustness of rollout-based evaluation for computational efficiency. One can mitigate the increased label noise inherent in this approach through careful judge calibration, ensemble methods, and multi annotator agreement when feasible.

Case C: Unclear Process Structure. For domains where step wise decomposition lacks clear structures, for example general question answering tasks, we approximate the process modeling through directly evaluating the partial response with a reward model trained with complete responses. For example, one may collect or reuse available pairwise preference data $\{(y^w, y^l)\}$ to train a Bradley-Terry model to score the process generation $R_\phi(y_{1:t}) \to \mathbb{R}$. The trained reward model provides holistic quality assessment that serves as guidance during decoding, approximating the intermediate process supervision even when the process structure is not well defined.

4 Alignment: Training and Decoding

4.1 Training-Time Optimization: Multi-Action-Head DPO

Multi-Action-Head LLM. To optimize H objectives efficiently, we extend a shared LLM backbone θ_b with H lightweight output heads, one per objective. Let $h_{\theta_b}(x, y_{1:t}) \in \mathbb{R}^d$ be the hidden state from the shared backbone for prefix $(x, y_{1:t})$. Each objective $i \in \{1, \ldots, H\}$ has projection matrix $W_i \in \mathbb{R}^{d\times|V|}$ producing logits and token probabilities:

$$\pi_{\theta_b, W_i}(y_t \mid x, y_{1:t}) = \text{softmax}(W_i^\top h_{\theta_b}(x, y_{1:t})). \tag{3}$$

The shared backbone captures general language ability, while heads encode objective-specific preferences. During inference, users can select a single head i or ensemble multiple heads with flexible weights $w_i \geq 0$:

$$\pi_{\text{MAH}}(y_t \,|\, x, y_{<t}) = \sum_{i=1}^{H} w_i \pi_{\theta_b, W_i}(y_t \,|\, x, y_{<t}). \tag{4}$$

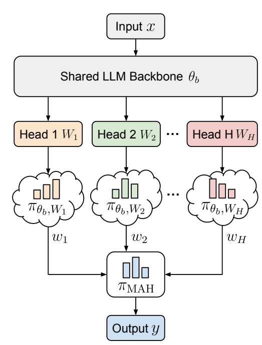

Figure 1: Multi-Action-Head LLM.

Multi-Action-Head DPO Objective. We curate H preference datasets $\{\mathcal{D}_i\}_{i=1}^H$, each tailored to objective i and labeled by PRMs or human annotations. All heads W_i are initialized from the SFT language modeling head of π_{θ_b} with small perturbations and the reference π_{ref} uses the unperturbed SFT head. For each $(x, y^w, y^l) \in \mathcal{D}_i$, routed to head i, the per-head DPO loss is

$$\mathcal{L}_i(\theta_b, W_i) = -\mathbb{E}_{(x,y^w,y^l)\sim\mathcal{D}_i}\left[\log\sigma\left(\beta\left(\log\frac{\pi_{\theta_b,W_i}(y^w \mid x)}{\pi_{\text{ref}}(y^w \mid x)} - \log\frac{\pi_{\theta_b,W_i}(y^l \mid x)}{\pi_{\text{ref}}(y^l \mid x)}\right)\right)\right]. \tag{5}$$

Let a mini-batch during training be partitioned as $\mathcal{B} = \bigsqcup_{i=1}^H \mathcal{B}_i$ where $\mathcal{B}_i$ gathers the examples assigned to head i. The combined loss we are minimizing is

$$\mathcal{L}_{\text{MAH-DPO}}(\theta_b, \{W_i\}) = \sum_{i=1}^{H}\alpha_i \cdot \frac{1}{|\mathcal{B}_i|}\sum_{(x,y^w,y^l)\in\mathcal{B}_i}\mathcal{L}_i\big(\theta_b, W_i; x, y^w, y^l\big), \tag{6}$$

where $\alpha_i \geq 0$ are objective weights with $\sum_i \alpha_i = 1$. More details including gradient analayasis can be found in Appendix C.

Table 1: Alignment performances of training-time methods across three datasets.

Method	Acc	Eng
Base	0.7107	0.5007
SFT	<u>0.7300</u>	0.5920
Single-Head DPO	0.7253	0.7160
MODPO	0.7280	0.7367
MAH-DPO Acc Head	**0.7353**	0.8667
MAH-DPO Eng Head	0.7267	**0.8840**
MAH-DPO Ensemble	0.7247	<u>0.8733</u>

(a) Math

Method	Help	Honest	Truth
Base	0.5800	0.3042	0.1888
SFT	0.5546	0.2998	0.1992
Single-Head DPO	0.6043	0.3055	0.2014
MODPO	0.6175	0.3477	<u>0.2325</u>
MAH-DPO Help Head	<u>0.6309</u>	0.3465	0.2239
MAH-DPO Honest Head	0.6257	<u>0.3516</u>	0.2303
MAH-DPO Truth Head	0.6257	0.3461	0.2286
MAH-DPO Ensemble	**0.6389**	**0.3687**	**0.2478**

(b) Human Values

Method	Acc	Eng
Base	0.6560	0.3220
SFT	0.6793	0.3473
Single-Head DPO	<u>0.7040</u>	0.4460
MODPO	**0.7047**	0.3600
MAH-DPO Acc Head	0.7007	0.4447
MAH-DPO Eng Head	0.6953	<u>0.4480</u>
MAH-DPO Ensemble	0.6893	**0.4513**

(c) Socratic Mind

Table 2: Alignment performances of test-time methods across three datasets.

Method	Acc	Eng
Base	0.6853	<u>0.5133</u>
Accuracy PRM-guided	<u>0.7633</u>	0.4720
Accuracy Value-guided	**0.7993**	0.4553
Engaging PRM-guided	0.7013	**0.7187**

(a) Math

Method	Help	Honest	Truth
Base	0.5750	0.3036	0.1904
Helpful PRM-guided	**0.6706**	0.4050	0.2791
Honesty PRM-guided	<u>0.6448</u>	**0.4693**	**0.3383**
Truthful PRM-guided	0.6350	<u>0.4394</u>	<u>0.3296</u>

(b) Human Values

Method	Acc	Eng
Base	0.6400	<u>0.3380</u>
Accuracy PRM-guided	**0.7127**	0.2660
Engaging PRM-guided	<u>0.6507</u>	**0.4663**

(c) Socratic Mind

4.2 Test-Time Optimization: PRM-Guided Decoding with Continuing Hidden State

We also explore PRM guided decoding with continuing hidden state for test time search, contrasting it with prior methods that rebuild the prompt at every step [28, 33, 43], which can alter hidden representations due to tokenization and special token placement, leading to performance drops observed in Appendix H. Our appraoch maintain a running key value cache initialized with a single forward pass on the chat formatted prompt. At each step we sample K candidates from the current cache, score them with a PRM P given the current prefix, select the best candidate, update the cache, and continue until an end of sequence token appears or the budget is reached. More details can be found in Appendix D and Algorithm 1 summarizes the procedure.

5 Experiments

In this section, we evaluate our multi-objective alignment framework across three domains.

Datasets, Evaluation, and PRM Training. We evaluate across three domains: **Math** using MATH [22], **Human Values** using UltraFeedback [12], and **AI Tutoring Dialogues** using Socratic Mind [25]. In mathematics we measure **Accuracy** by final answers and **Engagement** with calibrated LLM as Judge with human annotations; in human values we score **Helpfulness**, **Honesty**, and **Truthfulness** with trained reward models; in tutoring dialogues we estimate accuracy and engagement by simulating the student's next turn after the aligned assistant response and scoring it with a trained PRM. We train PRMs per domain using the standardized pipeline in Section 3, with full details in Appendix F.

5.1 Training-Time Alignment

We first evaluate our MAH-DPO approach across the above described three domains to validate its advantages. The baseline and implementation detail can be found in Appendix E.

Finding 1 - MAH-DPO yields the best multi-objective alignment performance. We present in Table 1 the main alignment results across Math, Human Values, and Socratic Mind for all compared training-time methods. Specialized heads lead on their target metrics without collapsing others, and the equal weight ensemble head in MAH-DPO aggregates these gains to deliver the strongest overall alignment across domains, including higher engagement in Socratic Mind with usable accuracy, while removing the need for objective specific selection at inference time.

Finding 2 - Head weighting provides smooth control with limited interference. We also show in Figure 3 and 4 further results on varying head weighting of MAH-DPO models during inference. Adjusting head weights at inference time traces a stable accuracy and engagement tradeoff in Math and improves combined outcomes in Human Values, delivering smooth gains in the emphasized dimension with only modest losses elsewhere and enabling practical weight choices that meet targets without retraining or manual selection.

Table 3: Alignment performances of synergizing training and test-time methods.

Method	Acc	Eng
Single-Head DPO	0.7253	0.7160
MODPO	0.7280	0.7367
MAH-DPO	0.7247	0.8733
MAH-DPO + Accuracy Value	**0.8000**	0.8553
MAH-DPO + Engaging PRM	0.7207	**0.9060**

(a) Math

Method	Help	Honest	Truth
Single-Head DPO	0.6043	0.3055	0.2014
MODPO	0.6175	0.3477	0.2325
MAH-DPO	0.6389	0.3687	0.2478
MAH-DPO + Help PRM	**0.7165**	0.4554	0.3890
MAH-DPO + Honest PRM	0.6968	**0.5196**	**0.4107**
MAH-DPO + Truth PRM	0.6834	0.4872	0.3630

(b) Human Values

Method	Acc	Eng
Single-Head DPO	0.7040	0.4460
MODPO	0.7047	0.3600
MAH-DPO	0.6893	0.4513
MAH-DPO + Accuracy PRM	**0.7160**	0.3800
MAH-DPO + Engaging PRM	0.7120	**0.5420**

(c) Socratic Mind

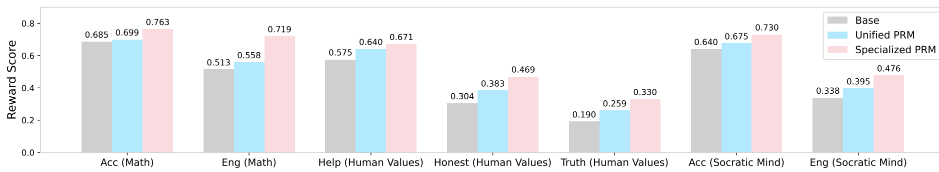

Figure 2: Alignment performances of a unified PRM trained across 7 dimensions in three domains compared with base model and the specialized PRM trained on each dimension per domain.

5.2 Test-Time Alignment

We evaluate PRM guided decoding with continuing hidden state across all three domains. The baseline and implementation detail can be found in Appendix E.

Finding 3 - PRM-guided decoding effectively improves the targeted objective. We report in Table 2 results across three datasets and observe that PRM guidance reliably raises the chosen metric while keeping non target metrics near base levels. In Math and Socratic Mind, accuracy focused or engagement focused guidance lifts its target with limited tradeoffs, and in Human Values, per dimension guidance reaches the best or second best scores on its axis, enabling smooth movement along objective fronts without retraining.

Finding 4 - Unified PRM trained on mixture of data shows cross-domain effectiveness. As shown in Figure 2, a single PRM trained on a seven dimension mixture improves all objectives over the base in Math, Human Values, and Socratic Mind, typically close to but below the best specialist on each axis. This unified model transfers across domains and provides a balanced improvement profile without domain specific retraining or serving multiple models, with details in Appendix F.

5.3 Synergizing Training and Test-Time Alignment

Finding 5 - Training and test-time methods complement each other in alignment. In Table 3 we pair MAH-DPO with an Ensemble head and PRM guidance at inference, which consistently improves the joint profile across Math, Human Values, and Socratic Mind over training only baselines. Accuracy focused and engagement focused selection shift outcomes along the expected tradeoff in Math and Socratic Mind, while in Human Values per dimension PRMs set the best axis scores and an ensemble PRM gives a balanced profile with positive transfer, all without retraining.

Finding 6 - Reward verifiability guides whether test-time or training-time method selection. From Tables 1, 2, and 3, verifiable rewards like Math accuracy see modest training gains, while PRM guided decoding at test time yields larger improvements using precise step level signals. For less verifiable goals such as helpfulness, honesty, truth, and engagement, multi head training provides the main lift and test time guidance then rebalances with minimal tradeoffs.

6 Conclusion

In this paper, we introduce a unified framework for multi-objective alignment at both training and inference, standardizing PRM training for verifiable and non-verifiable settings, using Multi-Action-Head DPO with vectorized rewards, and pairing it with PRM guided decoding with continuing hidden state. Experiments on math reasoning, value alignment, and multi-turn tutoring domains demonstrate the effectiveness of our framework for multi-objective alignment as well as fine-grained and flexible user control for alignment dimensions.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[2] Beng Heng Ang, Sujatha Das Gollapalli, and See Kiong Ng. Socratic question generation: A novel dataset, models, and evaluation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 147–165, 2023.

[3] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

[4] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[5] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Jiahao Chen, Zitao Liu, Mingliang Hou, Xiangyu Zhao, and Weiqi Luo. Multi-turn classroom dialogue dataset: Assessing student performance from one-on-one conversations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 5333–5337, 2024.

[8] Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. Pad: Personalized alignment of llms at decoding-time. *arXiv preprint arXiv:2410.04070*, 2024.

[9] Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Rui Zheng, Nijun Li, Tao Gui, Yun Li, et al. Better process supervision with bi-directional rewarding signals. *arXiv preprint arXiv:2503.04618*, 2025.

[10] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[11] 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. *arXiv preprint arXiv:2110.14168*, 2021.

[12] 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, 2023.

[13] Haikang Deng and Colin Raffel. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. *arXiv preprint arXiv:2310.09520*, 2023.

[14] Yuyang Ding, Hanglei Hu, Jie Zhou, Qin Chen, Bo Jiang, and Liang He. Boosting large language models with socratic method for conversational mathematics teaching. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3730–3735, 2024.

[15] 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. *Transactions on Machine Learning Research*.

[16] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

[17] Yiju Guo, Ganqu Cui, Lifan Yuan, Ning Ding, Zexu Sun, Bowen Sun, Huimin Chen, Ruobing Xie, Jie Zhou, Yankai Lin, et al. Controllable preference optimization: Toward controllable multi-objective alignment. *arXiv preprint arXiv:2402.19085*, 2024.

[18] Kunal Handa, Drew Bent, Alex Tamkin, Miles McCain, Esin Durmus, Michael Stern, Mike Schiraldi, Saffron Huang, Stuart Ritchie, Steven Syverud, Kamya Jagadish, Margaret Vo, Matt Bell, and Deep Ganguli. Anthropic education report: How university students use claude. `https://www.anthropic.com/news/anthropic-education-report-how-university-students-use-claude`, 2025. Accessed September 12, 2025.

[19] Kunal Handa, Drew Bent, Alex Tamkin, Miles McCain, Esin Durmus, Michael Stern, Mike Schiraldi, Saffron Huang, Stuart Ritchie, Steven Syverud, Kamya Jagadish, Margaret Vo, Matt Bell, and Deep Ganguli. Anthropic education report: How university students use claude, 2025. URL `https://www.anthropic.com/news/anthropic-education-report-how-university-students-use-claude`.

[20] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. *Advances in neural information processing systems*, 32, 2019.

[21] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

[22] 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. *arXiv preprint arXiv:2103.03874*, 2021.

[23] Yulan Hu, Sheng Ouyang, Jinman Zhao, and Yong Liu. Coarse-to-fine process reward modeling for mathematical reasoning. *arXiv preprint arXiv:2501.13622*, 2025.

[24] Yuzheng Hu, Ruicheng Xian, Qilong Wu, Qiuling Fan, Lang Yin, and Han Zhao. Revisiting scalarization in multi-task learning: A theoretical perspective. *Advances in Neural Information Processing Systems*, 36:48510–48533, 2023.

[25] Jui-Tse Hung, Christopher Cui, Diana M Popescu, Saurabh Chatterjee, and Thad Starner. Socratic mind: Scalable oral assessment powered by ai. In *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, pages 340–345, 2024.

[26] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[27] 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. *arXiv preprint arXiv:2310.11564*, 2023.

[28] Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Args: Alignment as reward-guided search. *arXiv preprint arXiv:2402.01694*, 2024.

[29] Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models. *arXiv preprint arXiv:2310.11589*, 2023.

[30] Bolian Li, Yifan Wang, Anamika Lochab, Ananth Grama, and Ruqi Zhang. Cascade reward sampling for efficient decoding-time alignment. *arXiv preprint arXiv:2406.16306*, 2024.

[31] Kaiwen Li, Tao Zhang, and Rui Wang. Deep reinforcement learning for multiobjective optimization. *IEEE transactions on cybernetics*, 51(6):3103–3114, 2020.

[32] Moxin Li, Yuantao Zhang, Wenjie Wang, Wentao Shi, Zhuo Liu, Fuli Feng, and Tat-Seng Chua. Self-improvement towards pareto optimality: Mitigating preference conflicts in multi-objective alignment. *arXiv preprint arXiv:2502.14354*, 2025.

[33] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

[34] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[35] Baijiong Lin, Weisen Jiang, Yuancheng Xu, Hao Chen, and Ying-Cong Chen. Parm: Multi-objective test-time alignment via preference-aware autoregressive reward model. *arXiv preprint arXiv:2505.06274*, 2025.

[36] Chien-Chang Lin, Anna YQ Huang, and Owen HT Lu. Artificial intelligence in intelligent tutoring systems toward sustainable education: a systematic review. *Smart learning environments*, 10(1):41, 2023.

[37] Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. *arXiv preprint arXiv:2309.15028*, 2023.

[38] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[39] Jakub Macina, Nico Daheim, Sankalan Pal Chowdhury, Tanmay Sinha, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. Mathdial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems. *arXiv preprint arXiv:2305.14536*, 2023.

[40] Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, et al. Controlled decoding from language models. *arXiv preprint arXiv:2310.17022*, 2023.

[41] OpenAI. Introducing study mode. `https://openai.com/index/chatgpt-study-mode/`, July 2025. Accessed September 12, 2025. Feature launched July 29, 2025.

[42] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[43] Sungjin Park, Xiao Liu, Yeyun Gong, and Edward Choi. Ensembling large language models with process reward-guided tree search for better complex reasoning. *arXiv preprint arXiv:2412.15797*, 2024.

[44] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[45] Alexandre Rame, 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. *Advances in Neural Information Processing Systems*, 36:71095–71134, 2023.

[46] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[47] Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hanna Hajishirzi, Noah A Smith, and Simon S Du. Decoding-time language model alignment with multiple objectives. *Advances in Neural Information Processing Systems*, 37:48875–48920, 2024.

[48] Kumar Shridhar, Jakub Macina, Mennatallah El-Assady, Tanmay Sinha, Manu Kapur, and Mrinmaya Sachan. Automatic generation of socratic subquestions for teaching math word problems. *arXiv preprint arXiv:2211.12835*, 2022.

[49] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] Katherine Stasaski, Kimberly Kao, and Marti A Hearst. Cima: A large open access dialogue dataset for tutoring. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–64, 2020.

[51] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

[52] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*, 2024.

[53] Kaiwen Wang, Jin Peng Zhou, Jonathan Chang, Zhaolin Gao, Nathan Kallus, Kianté Brantley, and Wen Sun. Value-guided search for efficient chain-of-thought reasoning. *arXiv preprint arXiv:2505.17373*, 2025.

[54] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

[55] Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, et al. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *arXiv preprint arXiv:2311.09528*, 2023.

[56] Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer 2: Open-source dataset for training top-performing reward models. *Advances in Neural Information Processing Systems*, 37:1474–1501, 2024.

[57] Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36: 59008–59033, 2023.

[58] Yu Xia, Tong Yu, Zhankui He, Handong Zhao, Julian McAuley, and Shuai Li. Aligning as debiasing: Causality-aware alignment via reinforcement learning with interventional feedback. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4684–4695, 2024.

[59] Yu Xia, Rui Wang, Xu Liu, Mingyan Li, Tong Yu, Xiang Chen, Julian McAuley, and Shuai Li. Beyond chain-of-thought: A survey of chain-of-x paradigms for llms. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10795–10809, 2025.

[60] Zhouhang Xie, Junda Wu, Yiran Shen, Yu Xia, Xintong Li, Aaron Chang, Ryan Rossi, Sachin Kumar, Bodhisattwa Prasad Majumder, Jingbo Shang, et al. A survey on personalized and pluralistic preference alignment in large language models. *arXiv preprint arXiv:2504.07070*, 2025.

[61] Wei Xiong, Wenting Zhao, Weizhe Yuan, Olga Golovneva, Tong Zhang, Jason Weston, and Sainbayar Sukhbaatar. Stepwiser: Stepwise generative judges for wiser reasoning. *arXiv preprint arXiv:2508.19229*, 2025.

[62] Yuancheng Xu, Udari Madhushani Sehwag, Alec Koppel, Sicheng Zhu, Bang An, Furong Huang, and Sumitra Ganesh. Genarm: Reward guided generation with autoregressive reward model for test-time alignment. *arXiv preprint arXiv:2410.08193*, 2024.

[63] Kailai Yang, Zhiwei Liu, Qianqian Xie, Jimin Huang, Tianlin Zhang, and Sophia Ananiadou. Metaaligner: Towards generalizable multi-objective alignment of language models. *Advances in Neural Information Processing Systems*, 37:34453–34486, 2024.

[64] 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. *arXiv preprint arXiv:2402.10207*, 2024.

[65] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[66] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

[67] Hanning Zhang, Pengcheng Wang, Shizhe Diao, Yong Lin, Rui Pan, Hanze Dong, Dylan Zhang, Pavlo Molchanov, and Tong Zhang. Entropy-regularized process reward model. *arXiv preprint arXiv:2412.11006*, 2024.

[68] Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*, 2023.

[69] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

[70] Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024.

[71] Zhanhui Zhou, Jie Liu, Chao Yang, Jing Shao, Yu Liu, Xiangyu Yue, Wanli Ouyang, and Yu Qiao. Beyond one-preference-for-all: Multi-objective direct preference optimization. 2023.

Appendix

A Ethics Statement

This work uses three data sources. For Socratic Mind tutoring dialogues, human subjects procedures were reviewed and approved by the authors' Institutional Review Board, and only students who gave explicit written consent were included. Participation was voluntary with no academic consequences, and students could withdraw at any time. Dialogues were deidentified, stored with encryption, and accessed only by approved researchers. Public datasets MATH and UltraFeedback were used under their licenses, and we cite the sources. We applied content filters and safety checks to reduce risks, avoided sensitive advice, and report remaining limitations. We will share code and configurations that do not compromise privacy or licensing.

B Related Work

Process Reward Model. Process supervision addresses a core limitation of outcome-only evaluation by giving rewards on intermediate reasoning steps, helping systems avoid trajectories that look correct but contain logical errors. The foundational approach involves collecting step-level human annotations for mathematical reasoning tasks and training process reward models on these dense supervision signals [34, 59]. Follow-up work scales supervision with automated or weakly supervised labels, for example per-step Monte Carlo rollouts or self-generated labels [38, 54]. Beyond standard PRMs, recent variants introduce progress or verifier signals that score both partial correctness and future success, improving search and ranking during decoding [9, 46]. There are also training objectives that regularize PRMs to improve stability [67]. Practical studies discuss data generation, evaluation pitfalls, and how PRMs differ from value functions that predict eventual solvability from partial traces [69]. Process-level search with step-wise scoring has further been shown to beat outcome-level test-time compute baselines in several setups, including controlled decoding, tree-structured search, and value/verification-guided search [37, 40, 46, 49, 53, 65].

Multi-Objective Alignment. Multi-objective alignment trains or steers language models for multiple, potentially conflicting objectives such as helpfulness, harmlessness, and honesty [60]. Standard RLHF pipelines fit a scalar reward and fine-tune with PPO, or use scalarized preference optimization [15, 42, 44, 58, 66], but they collapse trade-offs into one score. Two lines of work relax this restriction. Training-time approaches adapt multi-objective ideas, such as multi-objective RLHF and multi-objective direct preference optimization, or parameter mixing to balance different rewards [32, 45, 47, 52, 63, 71]. Complementing these training-based methods, test-time alignment enables dynamic objective balancing without model retraining. These approaches modify token probability distributions using reward guidance and perform search under composite objectives, achieving improvements on preference benchmarks while supporting per-user customization [8, 28, 35, 64]. This paradigm offers particular promise for multi-objective alignment where individual user preferences vary significantly.

C Training-Time Optimization: Multi-Action-Head DPO

Direct Preference Optimization. DPO [44] optimizes a policy π_θ against a fixed reference policy π_{ref} using preference pairs $\mathcal{D} = \{(x, y^w, y^l)\}$, where y^w is the preferred response to prompt x and y^l is the dispreferred one. The DPO loss is:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y^w,y^l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\left(\log\frac{\pi_\theta(y^w \mid x)}{\pi_{\text{ref}}(y^w \mid x)} - \log\frac{\pi_\theta(y^l \mid x)}{\pi_{\text{ref}}(y^l \mid x)}\right)\right)\right], \tag{7}$$

where $\sigma(\cdot)$ is the sigmoid and $\beta > 0$ is a temperature parameter controlling the strength of the preference signal.

Multi-Action-Head LLM. To jointly optimize for H distinct objectives while maintaining computational efficiency, we propose the multi-action-head LLM that extends the base LLM with specialized output layers. We maintain a single shared LLM backbone θ_b, while introducing H distinct linear projection heads, one for each alignment objective. This is more efficient than training H separate models, which would require H times the computational resources and fail to leverage cross objective synergies.

Specifically, let $h_{\theta_b}(x, y_{1:t}) \in \mathbb{R}^d$ denote the d-dimensional hidden state produced by the shared LLM backbone θ_b for input prefix $(x, y_{1:t})$. Each objective $i \in \{1, \ldots, H\}$ has a dedicated projection head parameterized by matrix $W_i \in \mathbb{R}^{d\times|V|}$ to produce objective-specific logits z_i and token probability distribution:

$$z_i(x, y_{1:t}) = W_i^\top h_{\theta_b}(x, y_{1:t}), \qquad \pi_{\theta_b,W_i}(y_t \mid x, y_{1:t}) = \text{softmax}(z_i(x, y_{1:t})) \tag{8}$$

where $|V|$ is the vocabulary size. The shared LLM backbone captures general language understanding and generation capabilities, while specialized heads can encode objective-specific preferences. During inference, our multi-action-head architecture supports flexible objective control by either selecting a specific head i for targeted behavior or ensembling logits from multiple heads for balanced performance:

$$\pi_{\text{MAH}}(y_t \mid x, y_{<t}) = \sum_{i=1}^{H} w_i \pi_{\theta_b,W_i}(y_t \mid x, y_{<t}), \tag{9}$$

where $w_i \geq 0$ are ensemble weights with $\sum_i w_i = 1$. This flexibility enables the model to be adapted for different downstream applications and user preferences without requiring separate training runs for each objective combination.

Multi-Action-Head DPO Objective. We first curate H preference datasets $\{\mathcal{D}_i\}_{i=1}^H$, where each $\mathcal{D}_i$ contains preference pairs specifically designed for objective i labeled using our trained PRM or from annotated labels. All heads W_i are initialized from the same language modeling head from the supervised fine-tuned (SFT) LLM π_{θ_b} with small random perturbations to encourage specialization. The reference model π_{ref} retains the unperturbed SFT head. During training, examples $(x, y^w, y^l) \in \mathcal{D}_i$ are routed to head i, and we compute the objective-specific DPO loss:

$$\mathcal{L}_i(\theta_b, W_i) = -\mathbb{E}_{(x,y^w,y^l)\sim\mathcal{D}_i}\left[\log\sigma\left(\beta\left(\log\frac{\pi_{\theta_b,W_i}(y^w \mid x)}{\pi_{\text{ref}}(y^w \mid x)} - \log\frac{\pi_{\theta_b,W_i}(y^l \mid x)}{\pi_{\text{ref}}(y^l \mid x)}\right)\right)\right]. \tag{10}$$

Let a mini-batch during training be partitioned as $\mathcal{B} = \bigsqcup_{i=1}^H \mathcal{B}_i$ where $\mathcal{B}_i$ gathers the examples assigned to head i. The combined loss we are minimizing is

$$\mathcal{L}_{\text{MAH-DPO}}(\theta_b, \{W_i\}) = \sum_{i=1}^{H} \alpha_i \cdot \frac{1}{|\mathcal{B}_i|} \sum_{(x,y^w,y^l)\in\mathcal{B}_i} \mathcal{L}_i\big(\theta_b, W_i; x, y^w, y^l\big), \tag{11}$$

where $\alpha_i \geq 0$ are objective weights with $\sum_i \alpha_i = 1$.

Gradient Analysis. The gradients for parameters of each head j are isolated by routing, while the backbone LLM gradients accumulate across heads:

$$\nabla_{W_j} \mathcal{L} = \sum_{i=1}^{H} \alpha_i \cdot \frac{1}{|\mathcal{B}_i|} \sum_{(x,y^w,y^l)\in\mathcal{B}_i} \underbrace{\nabla_{W_j} \mathcal{L}_i(\theta_b, W_i; x, y^w, y^l)}_{=0 \text{ if } j\neq i} \;=\; \alpha_j \cdot \mathbb{E}_{\mathcal{B}_j}\left[\nabla_{W_j} \mathcal{L}_j\right], \tag{12}$$

$$\nabla_{\theta_b} \mathcal{L} = \sum_{i=1}^{H} \alpha_i \cdot \frac{1}{|\mathcal{B}_i|} \sum_{(x,y^w,y^l)\in\mathcal{B}_i} \nabla_{\theta_b} \mathcal{L}_i(\theta_b, W_i; x, y^w, y^l). \tag{13}$$

Therefore, a single backward pass through Equation 11 updates the backbone and every active action heads simultaneously. To achieve more stable training and balanced gradient propagation, we can construct mini-batches with similar number of examples $|\mathcal{B}_i|$ from each objective i or by tuning the weights α_i when the dataset sizes differ. Since every head consumes the same hidden states for its logits, the computation requires only one backbone forward per input and parallel per-head projections, leveraging cross objective synergies without introducing excessive extra training cost.

D Test-Time Optimization: PRM-Guided Decoding with Continuing Hidden State

We also explore the use of our trained PRM during test-time directly via step-level reward guided decoding. Existing reward-guided decoding or test-time search methods [28, 33, 43] typically rebuild

Algorithm 1: PRM-Guided Decoding with Continuing Hidden State

Input: policy π_θ; PRM P; boundary detection criteria $\mathcal{Q}$; number of candidates K; token budget $T_{\max}$; prompt x.
Output: response y.
$\mathrm{kv}_0 \leftarrow \mathrm{Fwd}_{\pi_\theta}(x)$; $y_{1:0} \leftarrow \emptyset$; $t \leftarrow 0$.
while $|y_{1:t}| < T_{\max}$ ***and*** $\mathrm{EOS} \notin y_{1:t}$ **do**
  for $k = 1$ **to** K **do**
    $\widetilde{\mathrm{kv}} \leftarrow \mathrm{kv}_t$; $\tilde{y} \leftarrow \emptyset$.
    while $\mathcal{Q}(\tilde{y}) = 0$ **do**
      Sample next token $z \sim \pi_\theta(\cdot \mid \widetilde{\mathrm{kv}})$;
      $\widetilde{\mathrm{kv}} \leftarrow \mathrm{Fwd}_{\pi_\theta}(\widetilde{\mathrm{kv}}, z)$; $\tilde{y} \leftarrow \tilde{y} \,\|\, z$.
    Record end-state cache $\mathrm{kv}^k_{t+1} \leftarrow \widetilde{\mathrm{kv}}$;
    Record candidate next step $y^k_{t+1} \leftarrow \tilde{y}$;
    Score with PRM $r_k \leftarrow P\big(x,\, y_{1:t},\, y^k\big)$.
  $k^\star \in \arg\max_k r_k$;
  Update running cache $\mathrm{kv}_{t+1} \leftarrow \mathrm{kv}^{k^\star}_{t+1}$;
  Update response $y_{1:t+1} \leftarrow y_{1:t} \,\|\, y^{k^\star}_{t+1}$;
  $t \leftarrow t + 1$.

the prompt each step by concatenating the newly selected next generation with previous steps. However, rebuilding and re-encoding the textual prompt each step can change how the prior context is represented within the hidden state, e.g., small differences in tokenization around whitespace and newline merges, shifts in relative positions, and the placement of special tokens from chat templates. As a result, the next-token distribution after re-encoding can differ from the one obtained by directly continuing from the previous step and such discontinuity can lead to performance degradation as observed in our experiments presented in Appendix H.

Therefore, to preserve the generation continuity at hidden state level, we utilize a running past key–value cache during our PRM-guided decoding. The same hidden state is carried forward, so the continuation distribution follows the true incremental decoding rather than a fresh prompt re-encoding approximation. We provide an overview of our PRM-guided decoding in Algorithm 1 and describe details as follows.

Cache Initialization and Candidate Proposal. Given a chat-formatted prompt x, we run a single forward pass with the policy model π_θ to obtain the initial past key–value cache kv_0 and the first next-token distribution. We set response $y_{1:0} = \emptyset$ and generation step index $t = 0$. This avoids re-encoding x in later steps and provides the reference state from which all continuations proceed. Then, for each step t, we proposal K candidates from the current running cache kv_t. For each candidate k, we clone kv_t to a local copy and sample the next token from policy model π_θ while carrying that local cache forward. Sampling stops when the boundary detection criteria $\mathcal{Q}$ triggers. This yields a step generation y^k_{t+1} with its end-state cache kv^k_{t+1}.

Candidate Selection with PRM and Cache Update. Each sampled candidate is then evaluated by a PRM P. Given the current prefix $y_{1:t}$, the score for candidate k is $r_k = P(x, y_{1:t}, y^k_{t+1})$. We select $k^\star = \arg\max_k r_k$, append the chosen step generation to the response $y_{1:t+1} = y_{1:t} \,\|\, y^{k^\star}_{t+1}$, and update the current running cache as $\mathrm{kv}_{t+1} = \mathrm{kv}^{k^\star}_{t+1}$. This commit keeps decoding stateful across segments rather than re-encoding the prompt with textual concatenations. We repeat the above candidate proposal with π_θ starting from kv_t, PRM scoring, and cache update until an end-of-sequence token appears or a token budget is reached. With every iteration advancing from the running cache, the generation remains continuous with respect to model's internal hidden state.

Computational Analysis. Besides keeping the generation continuity at hidden-state level, our cache-carrying PRM-guided decoding also reduce the computational cost compared to re-encode-per-step baselines. Let $|x|$ be the prompt length, T the committed output tokens, N the number of steps, i.e., detected boundaries, K the candidates per step, and $\bar{L}$ the average candidate length such that

$T \approx N\bar{L}$. A re-encode-per-step policy costs $\mathcal{O}(K(|x|N + NT))$ while our cache-carrying policy costs $\mathcal{O}(|x| + KN\bar{L}) = \mathcal{O}(|x| + KT)$. Thus the factor N that multiplies T is removed, enabling better test-time scaling by shifting compute from repeated re-encodings to candidate rollout or longer outputs.

E Baseline and Implementation Details

E.1 Training-Time Alignment

Baselines and Variants. We report results of the following baselines as well as our MAH-DPO variants. **Base** is the original based LLM without any post-training or alignment. **SFT** applies supervised fine-tuning using only the preferred responses from preference pairs. **Single-Head DPO** directly applies DPO to one primary objective by pooling all dimension-specific preference data. **MODPO** [71] is a multi-objective extension of DPO that optimizes multiple alignment objectives in an RL-free manner by combining objectives with weights during training. **MAH-DPO Individual Head** reports the performance of each specialized head when used independently, reflecting objective-specific capabilities. **MAH-DPO Ensemble** uses an equal-weight combination of all head logits, representing our balanced multi-objective approach. We also analyze MAH-DPO inference with varying weights in Figure 3 and 4.

Implementation Details. We build paired preference datasets with our trained PRM or annotations in three domains as follows: Math (contrasting correct vs. incorrect rollouts and engaging vs. non-engaging solutions), Human Values (UltraFeedback subsets for helpfulness, honesty, and truthfulness), and Socratic Mind (simulated tutoring dialogues scored by trained PRMs). We train MAH-DPO on `Qwen2.5-7B-Instruct` for Math and Socratic Mind (SFT then MAH-DPO), and on `meta-llama/Llama-3.1-8B-Instruct` for Human Values. Models use domain-appropriate learning rates, batch sizes, and context windows. Full data construction and hyperparameters are in provided in Appendix G. All experimental results are averaged over 3 independent runs and we report standard deviations in Appendix J.

E.2 Test-Time Alignment

Baselines and Variants. We report results of the following baselines as well as our PRM-guided decoding variants. **Base** utilizes the base model directly for step-wise generation without candidate sampling or selection. **Individual PRM-guided Decoding** applies an individual PRM trained for each objective dimension to guide the base model generation step by step following the candidate sampling-then-selection pipeline.

Implementation Details. We apply the same decoding strategy across all domains using the same base models as in training. In Math, we treat natural reasoning boundaries marked by `\n\n` as step boundary, and we use our trained accuracy and engagement PRMs to guide step-level generation. In Human Values, where responses are nonverifiable and lack fixed process structure, we impose boundaries at sentence terminators and paragraph breaks, and use our trained reward models to score helpfulness, honesty, and truthfulness under step-level computational budgets of 256 tokens per chunk and 1,024 total tokens. In Socratic Mind, each turn is treated as a step and scored with our trained engagement and accuracy PRMs. Across all domains we sample $K = 5$ candidates at each step. All decoding runs use temperature=1.0, top-p=1.0, and top-k=50 to ensure diversity while maintaining consistent selection under reward guidance. We provide further results validating the effectiveness of our use of continuing hidden state for PRM-guided decoding in Appendix H. All experimental results are averaged over 3 independent runs and we report standard deviations in Appendix J.

F PRM Training Details

F.1 Math PRM Training

Accuracy PRM Training. We implement our rollout approach with hindsight relabeling to train a process reward model for mathematical accuracy following Section **??**. Our method leverages an existing well-trained PRM, specifically `Qwen/Qwen2.5-Math-PRM-7B`, to provide intermediate step-level rewards that we combine with terminal outcome signals through our principled framework.

For each candidate reasoning step, we generate 5 independent rollouts using sampling to completion. Step values are computed by combining intermediate PRM rewards with binary final outcome rewards, where correct solutions receive a reward of 1 and incorrect solutions receive 0. These rewards are weighted by a temporal discount factor and averaged across all rollouts to obtain reliable step-level supervision signals for step selection and trajectory extension. The iterative generation process continues until either a final boxed answer is produced or the maximum step limit of 20 is reached, yielding step values within the range $[0, 2]$. Given that the average mathematical problem requires 9-12 reasoning steps, we set the discount rate $\gamma = 0.9$ to appropriately balance immediate step quality assessment with long-term credit assignment.

We also swept the discount factor when turning per-step PRM rewards into value targets and repeated both value-head training and guided decoding. Concretely, for a step prefix $s_{\leq t}$ we formed discounted returns $G_t = \sum_{k\geq 0} \gamma^k r_{t+k}$ with $\gamma \in \{0.9, 0.95\}$, trained the same frozen-backbone + MLP value head to regress G_t via MSE, then used the learned value to steer generation: at each step we propose candidate continuations and pick the one maximizing a blended objective $\alpha\, V(s_{\leq t}+\text{cand}) + (1-\alpha)\log P(\text{cand} \mid s_{\leq t})$. Lower γ favors short-term gains, while higher γ encourages longer-horizon reasoning during decoding.

Table 4: Comparison of math step-level guided decoding methods and their accuracy.

Guided Decoding Method	Accuracy	Engagingness
Baseline step-by-step	0.6853 ± 0.0163	0.5133 ± 0.0543
PRM-guided	0.7633 ± 0.0050	0.7187 ± 0.0266
Value head guided with $\gamma = 0.90$	0.7993 ± 0.0172	0.4553 ± 0.0221
Value head guided with $\gamma = 0.95$	0.7993 ± 0.0081	0.5053 ± 0.0050
MAH-DPO Ensemble Head + Accuracy PRM-guided with $\gamma = 0.90$	0.8000 ± 0.0231	0.8553 ± 0.0136
MAH-DPO Ensemble Head + Accuracy PRM-guided with $\gamma = 0.95$	0.7800 ± 0.0197	0.8470 ± 0.0098

Our PRM architecture follows the design from `Qwen/Qwen2.5-Math-PRM-7B` [69], where we replace the standard language modeling head with a two-layer scalar value head that produces step-level quality scores. Reasoning steps are serialized using the special separator token `<extra_0>` in chat-format input, with the transformer's hidden state at each separator token position marking step boundaries. These boundary representations feed into a compact MLP for per-step value prediction. During training, we freeze the PRM backbone parameters from `Qwen/Qwen2.5-Math-PRM-7B` and optimize only the value head using mean squared error loss against the soft step-value targets. Training proceeds for 2 epochs with a batch size of 32 and learning rate of 5e-5.

Engagement PRM Training. To evaluate our approach on subjective quality dimensions, we construct an engagement-focused dataset. We sample 50 problems from the MATH training split and generate 4 solution rollouts per problem using the base model. These rollouts use an even mix of engaging and non-engaging reasoning style system prompts to ensure balanced representation (see Appendix I). Human annotators label all 200 responses for engagement quality, providing ground truth supervision for this subjective dimension. We calibrate an LLM-as-Judge using `Qwen/Qwen2.5-72B-Instruct` to evaluate engagement levels, achieving 75.8% classification accuracy against human-labeled solutions. This calibrated judge enables scalable engagement evaluation during PRM training (see Appendix I for calibrated system prompt).

For each problem, we generate one initial reasoning step, then create eight diverse completions continuing from the current state using generation temperature 1.0. The calibrated LLM-as-Judge scores engagement for every completion batch per step. Following our Case A methodology for non-verifiable domains in Section 3.2, we label a step as engaging if more than four out of eight rollouts continuing from that step are deemed engaging, otherwise it receives a non-engaging label. This process yields 11.8k step-level engagement annotations. We then convert the training data into incremental reasoning sequences, where each step accumulates the solution path from problem statement through progressive reasoning chains. The base model for the PRM training is `meta-llama/Llama-3.1-8B` configured for binary classification. We train for 2 epochs using batch size 128, learning rate 1e-5 which achieves an evaluation accuracy of 92.5%.

F.2 Human Values PRM Training

Human values represent a non-verifiable domain with no clear process structure. Rather than forcing artificial step-level decomposition, we follow our Case C methodology in Section 3.2 and train reward model for holistic quality assessment. We train Bradley-Terry reward models on top of the SFT model with base model as `meta-llama/Llama-3.1-8B` following the RLHFlow recipe [16] with learning rate 1e-5 and batch size 32 for 3 epochs. The reward model learns to capture human preferences across the helpfulness, honesty, and truthfulness dimensions through pairwise preference optimization, providing dense guidance signals for fine-grained decoding without requiring artificial process supervision.

F.3 Socratic Mind PRM Training

Students complete post-interaction surveys rating their experience on a 0-6 scale regarding how the Socratic Mind approach enhanced their understanding, serving as our engagement dimension ground truth. We classify ratings ≥ 4 as engaging interactions. Student dialogues are collected with engagement ratings, and conversations are randomly truncated after assistant turns to create training samples with varying trajectory lengths. We establish calibration datasets with 80 training and 80 test samples to calibrate an LLM-as-judge using GPT-4o [26], achieving 0.8 training accuracy and 0.66 test accuracy for engagement prediction. We additionally curate a specialized judge for accuracy evaluation where system prompt for both objectives can be found in Appendix I. The calibrated LLM-as-judge labels approximately 5k engagement samples and 8k accuracy samples for PRM training, achieving 0.81 test accuracy for engagement and 0.7 for accuracy using classification on `Llama-3.1-8B`.

F.4 Unified PRM Training

We constructed a unified binary-classification corpus by combining all 7 objective dimensions from the domain datasets used in our experiments and formatting each example as a "User:"/"Assistant:" dialogue with blank-line spacing. Math engagement conversations yield incremental stepwise instances labeled from $+/-$. Human value preference pairs are mapped to chosen $= 1$ and rejected $= 0$. Math value scores are normalized per example and thresholded ($> 0.85 \to 1$, otherwise 0). Socratic Mind engagement and accuracy retain only multi-turn dialogues, with accuracy excluding the last turn. This pipeline produced a total of 168,514 examples with 47.4% positives. We then fine-tuned a pre-trained `Llama-3.1-8B` model with a 2-class classification head using cross-entropy. Training used a batch size of 128, a learning rate of 1×10^{-5}, and ran for 2 epochs.

G Training-time Alignment Details

G.1 Math Training Details

Mathematical reasoning presents a natural testbed for multi-objective alignment, as effective tutoring requires balancing computational accuracy with pedagogical engagement. We design our experimental setup to capture this fundamental trade-off in educational AI systems.

Preference Data Construction. We construct two complementary preference datasets using the MATH training dataset (12k problems) to target distinct but interrelated aspects of mathematical competence:

- *Accuracy-focused pairs*: For each problem, we generate up to 30 response rollouts using Qwen2.5-7B-Instruct, extract boxed numerical answers, and compare against ground truth solutions. We pair the first correct solution with the first incorrect one encountered, creating 5,574 preference pairs that emphasize computational precision and mathematical correctness.
- *Engagement-focused pairs*: Using the same problem set, we generate 10 rollouts per question and employ LLM-as-Judge evaluation (Qwen2.5-72B-Instruct, temperature=0.1) to assess pedagogical quality. We identify responses that provide clear explanations, intuitive reasoning, and educational insights versus those offering terse or mechanical solutions, yielding 7,930 preference pairs that prioritize learning effectiveness over mere correctness.

This dual construction allows us to examine whether MAH-DPO can simultaneously optimize for mathematical rigor and educational value—objectives that often compete in practice.

Training Configuration. We establish a consistent training pipeline across all mathematical experiments. Starting from Qwen2.5-7B-Instruct, we first perform supervised fine-tuning (learning rate 5×10^{-6}, 2 epochs) to adapt the model to mathematical domains. We then initialize MAH-DPO with small random perturbations (scale=0.001) applied to each head to encourage objective-specific specialization while maintaining shared representations. The multi-head training uses learning rate 1×10^{-6}, batch size 128, and $\beta = 0.1$, with sequences truncated to 512 prompt tokens and extended to 1536 total tokens to accommodate detailed mathematical reasoning over 2 epochs.

G.2 Human Values Training Details

Human values alignment represents a more abstract but equally critical challenge, where models must navigate competing ethical principles. We focus on three fundamental dimensions that frequently conflict in real-world applications: helpfulness, truthfulness, and honesty.

Preference Data Construction. We leverage the UltraFeedback dataset's rich dimensional annotations to create three targeted preference datasets:

- *Helpfulness*: 59.2k preference pairs contrasting responses that provide comprehensive, actionable guidance versus those offering minimal or irrelevant information.
- *Truthfulness*: 50.8k pairs emphasizing factual accuracy and evidence-based reasoning versus responses containing inaccuracies or unsupported claims.
- *Honesty*: 57.3k pairs focusing on transparent acknowledgment of uncertainty and limitations versus responses that overstate confidence or mask knowledge gaps.

For each dimension, we pair responses with the highest and lowest annotated scores while excluding cases with identical ratings, ensuring clear preference signals. We reserve 2k examples per dimension for comprehensive evaluation across all three values simultaneously.

Training Configuration. To maintain experimental consistency while adapting to the distinct characteristics of values alignment, we modify our training approach accordingly. We perform supervised fine-tuning on Llama-3.1-8B using UltraFeedback's preferred responses (learning rate 5×10^{-7}, 1 epoch, batch size 192) to establish a strong foundation for ethical reasoning. MAH-DPO training employs slightly larger perturbations (scale=0.005) to account for the more nuanced nature of value judgments, with learning rate 5×10^{-7}, batch size 120, and sequences limited to 256 prompt tokens and 768 total tokens to focus on concise value-aligned responses over 1 epoch.

G.3 Socratic Mind Training Details

Socratic tutoring epitomizes the challenge of multi-objective alignment in educational settings, requiring models to maintain factual accuracy while fostering student engagement through strategic questioning and explanation. This domain tests our approach's ability to handle dynamic, context-dependent trade-offs.

Preference Data Construction. We simulate realistic tutoring interactions by randomly sampling 1,000 educational dialogues and introducing natural conversation breakpoints. At each dialogue state, we generate 5 potential assistant responses representing different tutoring strategies—from direct instruction to guided discovery. We then employ trained PRMs specialized for accuracy and engagement assessment to evaluate each candidate response. By selecting the highest and lowest scoring responses for each objective, we create 1,000 preference pairs per dimension that capture the nuanced balance between providing correct information and maintaining pedagogical effectiveness in conversational contexts.

Training Configuration. Given the complexity of dialogue understanding, we adopt our mathematical domain configuration while extending context capabilities. We fine-tune Qwen2.5-7B-Instruct (learning rate 5×10^{-6}, 2 epochs) and apply MAH-DPO with perturbation scale 0.001 to preserve dialogue coherence across heads. Training employs learning rate 1×10^{-6}, batch size 256, and $\beta = 0.1$, with extended context windows (1336 prompt tokens, 1536 total tokens) to accommodate full dialogue history while maintaining computational efficiency over 2 epochs.

These three experimental domains collectively span the spectrum from concrete mathematical reasoning to abstract value judgments to dynamic conversational interaction, providing a comprehensive testbed for evaluating MAH-DPO's multi-objective alignment capabilities across diverse AI applications.

H Continuing Hidden State Ablation

In this section, we provide further results for validating the effectiveness of continuing hidden state in our PRM-guided decoding for alignment. We present comparisons between our continuing hidden state approach with classic text chunk concatenation approach and the results are in Table 5 and 6. From Table 5, we observe that in Human Values where there is not a clear process structure, step-wise generation using text chunk concatenation leads to performance degradation compared to the one-pass generation. Meanwhile, our continuing hidden state approach achieve comparable performance with one-pass generation when no guidance from PRMs is used, and also consistent improvements over text chunk method when guided by PRMs. This demonstrates that text chunk concatenation which requires iterative re-encoding can break the generation continuity while our hidden state approach preserve such continuity for response generation. In Table 6, there is no major performance difference between text chunk method and our hidden state method, which indicates the text chunk methods does not break generation continuity when the process structure is clear and well-defined such as in Math domain.

Table 5: Further results of PRM-guided decoding in Human Values: continuing text chunk vs. continuing hidden state.

Method	Help	Honest	Truth
One-pass generation without guided decoding (reference)	0.5800 ± 0.0066	0.3042 ± 0.0066	0.1888 ± 0.0028
Step-wise generation without guided decoding (text chunk)	0.4688 ± 0.0033	0.1857 ± 0.0016	0.1182 ± 0.0031
Step-wise generation without guided decoding (hidden state)	0.5750 ± 0.0107	0.3036 ± 0.0015	0.1904 ± 0.0036
Step-wise generation + Helpful PRM guided (text chunk)	0.6140 ± 0.0099	0.3273 ± 0.0069	0.2099 ± 0.0060
Step-wise generation + Helpful PRM guided (hidden state)	0.6706 ± 0.0093	0.4050 ± 0.0035	0.2791 ± 0.0023
Step-wise generation + Honest PRM guided (text chunk)	0.6148 ± 0.0150	0.3860 ± 0.0106	0.2544 ± 0.0062
Step-wise generation + Honest PRM guided (hidden state)	0.6448 ± 0.0050	0.4693 ± 0.0045	0.3383 ± 0.0025
Step-wise generation + Truth PRM guided (text chunk)	0.5775 ± 0.0155	0.3165 ± 0.0028	0.2500 ± 0.0062
Step-wise generation + Truth PRM guided (hidden state)	0.6350 ± 0.0032	0.4394 ± 0.0036	0.3296 ± 0.0056

Table 6: Further results of PRM-guided decoding in Math: continuing text chunk vs. continuing hidden state.

Method	Accuracy	Engagement
One-pass generation without guided decoding (reference)	0.7107 ± 0.0090	0.5007 ± 0.0289
Step-wise generation without guided decoding (text chunk)	0.7040 ± 0.0092	0.4907 ± 0.0358
Step-wise generation without guided decoding (hidden-state)	0.6853 ± 0.0163	0.5133 ± 0.0543
Step-wise generation + Engaging PRM guided (text-chunk)	0.7187 ± 0.0147	0.6353 ± 0.0099
Step-wise generation + Engaging PRM guided (hidden-state)	0.7013 ± 0.0352	0.7187 ± 0.0266
Step-wise generation + Accuray PRM guided (text-chunk)	0.7973 ± 0.0083	0.4807 ± 0.0205
Step-wise generation + Accuracy PRM guided (hidden-state)	0.7993 ± 0.0172	0.4553 ± 0.0221

I System Prompts

In this section, we provide the system prompts used for response generation and LLM-as-Judge. Apart from the domains or alignment objective dimensions specified as follows, no system prompt is used. For example, we do not use system prompt for response generation in Human Values experiments.

I.1 Math System Prompts

Response Generation System Prompt in Math

Normal

Please reason step by step, and put your final answer within `\\boxed{}`.

Engaging

Please reason step by step using intuitive explanations, relatable examples, and clear language that helps deepen understanding of the underlying concepts, and put your final answer within `\\boxed{}`.

Non-Engaging

Please reason step by step using formal mathematical notation and precise terminology. Minimize explanatory content, focus exclusively on necessary calculations and logical steps, and put your final answer within `\\boxed{}`.

Engagement LLM-as-Judge System Prompt in Math

You are an expert evaluator of mathematical explanations. Your task is to determine whether the mathematical content presented is **ENGAGING** or **NOT ENGAGING**.

ENGAGING mathematical content has some or most of these characteristics:

- Uses intuitive explanations and relatable examples
- Provides clear language that helps deepen understanding
- Makes abstract concepts accessible and interesting
- Includes explanatory content that builds conceptual understanding
- Presents reasoning that is easy to follow
- Balances formal notation with helpful explanations

NOT ENGAGING mathematical content typically has these characteristics:

- Uses primarily formal mathematical notation and terminology
- Provides minimal explanations beyond the calculations
- Focuses exclusively on necessary calculations and logical steps
- Lacks intuitive explanations or relatable examples
- Uses dense, technical language that may be harder to follow
- Prioritizes brevity and formality over accessibility

Evaluate **only** the engagingness of the content, not its correctness.

Your evaluation must be in JSON format with two fields:

```
{"analysis": "<specific reasons why the content is or is not engaging>",
 "judgment": "<ENGAGING or NOT ENGAGING>"}
```

Please evaluate the following mathematical content:

PROBLEM:

```
{problem}
```

SOLUTION:

```
{solution}
```

I.2 Socratic Mind System Prompts

Accuracy LLM-as-Judge System Prompt in Socratic Mind

You are an evaluator of tutoring dialogues. Your task is to judge the **ACCURACY** of the **ASSISTANT'S LAST MESSAGE**. Use the student's immediate reply only as a probe. The label will train a reward model that must also work when only the assistant message is present.

Goal
Decide whether the assistant's last message is factually correct, specific, and checkable so that a competent student could reach a correct answer without extra unstated information. Use the student reply to test this, but base the label on the assistant message itself.

Scope
Read only the final assistant message and the immediately following student reply. Consult earlier turns only to decode terms, variable meanings, or given values when strictly needed. Do not judge engagement, style, or tone.

Strict Criteria (all must hold for ACCURATE)

- **Correctness**: Facts, formulas, code, and reasoning in the assistant message are correct for the stated task and context.
- **Sufficiency**: The message includes the key inputs, units, constraints, and acceptance criteria needed to verify a result. It does not rely on hidden assumptions.
- **Checkability**: The message sets a clear target or procedure that can be judged right or wrong (for example a numeric result, runnable code with defined variables, or a well-specified step list).
- **Consistency**: The message does not conflict with earlier defined variables, values, or conditions.

Stricter Default
If there is reasonable doubt about correctness or checkability, choose INACCURATE. Praise, summaries, or meta talk that do not set a checkable target are INACCURATE for this accuracy task.

How to Use the Student Reply
Use it only as evidence of whether the assistant message was clear and correct. If the reply is wrong or a non-answer and the assistant message was underspecified, ambiguous, misleading, or used wrong facts/code, label INACCURATE. If the reply is wrong but the assistant message was fully correct and checkable, you may still label ACCURATE. If there is no reply, judge the assistant message alone by the strict criteria.

Concrete Failure Patterns that Require INACCURATE

- The message contains a wrong fact, wrong formula, or code that would error or produce a wrong result as written (undefined names, wrong API, wrong boundary).
- The task depends on data or tools not provided and the message does not state allowed assumptions.
- The target is not objectively checkable (for example "share your thoughts") or key constraints/units are missing so multiple incompatible answers fit.
- The message conflicts with established context (for example uses 2.5 when 3.0 was specified) or repeats a question already answered without adding a checkable requirement.

Decision Rules

1. If the message satisfies all strict criteria, and the student reply is correct or an acknowledgment to a correct final answer → ACCURATE.
2. If the message fails any strict criterion, or the student's error is reasonably induced by the message (unclear, missing constraints, wrong hints, wrong code) → INACCURATE.
3. In uncertain cases, default to INACCURATE.

Dialogue: {conversation}

Your evaluation must be in JSON format:

```
{
  "label": "ACCURATE" or "INACCURATE"
}
```

Engagement LLM-as-Judge System Prompt in Socratic Mind

You are an evaluator of programming tutoring dialogues. Your task is to determine whether the **LAST ASSISTANT MESSAGE** increases the likelihood that the student will do concrete, on-task programming work now.

Scope and Evidence
Read the LAST ASSISTANT MESSAGE. Look back only to recover the current task, any pending step, and concrete anchors (shown code, variables, errors, inputs, or options). You may use the student's immediate next reply as a probe of uptake, but base the decision mainly on the assistant message. Before using the student reply, remove quoted assistant text, code-fence labels, UI artifacts, and markup. Do not judge tone.

What Counts as Engagement-Raising
The message raises engagement when it asks for a clear, task-specific programming action that yields a result verifiable from the dialogue now. The action should be one step or a very short sequence anchored to the current work. The following qualify (treat any one as sufficient):

- Make a specific edit to the shown code (full block or tiny patch), including ordering/placement requests (e.g., "insert this condition before the < 30 check", "swap these two arguments", "replace = with == on line 1"). The edited code itself is the check.
- Write or complete a small snippet ($\approx$10 lines or fewer) tied to the current construct (e.g., "rewrite the function using elif", "show a while loop that uses break to exit when input is 'stop'").
- Predict one concrete outcome tied to the code and inputs (e.g., "what prints for level = 90?", "which branch runs when time_left == 30?", "will this raise a SyntaxError?").
- Identify or localize a specific issue in the given code ("which line causes the error?", "what rule is violated by this call?") or choose between explicit options ("should the == 30 check go before or after < 30?").
- Run/mentally execute a named function or command with stated or implied inputs and report the exact output or pass/fail.
- Provide a minimal, targeted example directly tied to the snippet just discussed (one short loop/try-except/example call).

Also count as engagement-raising:

- Requests to finish a started step (e.g., "complete the code you began with the missing elif..."), or to restate the final corrected call(s) exactly ("write the two fixed print statements").
- Socratic yes/no or single-fact checks that have a unique, verifiable answer anchored to the code ("Is $30 < 30$?", "Would the elif run when time_left is 30?").

What is Not Engagement-Raising
The message is NOT engagement-raising when it only explains/summarizes; asks open "why/how/compare/explain" without anchoring to the current code or a bounded artifact; gives a full final solution leaving nothing to do; posts long code or text without a precise "do-now" instruction; goes off task; or tells the student to wait/stop while a step is pending.

Pending-Step Handling
If an earlier assistant turn set a step that is still unfinished (write/implement/fix/modify/calculate/answer/show code/run and report), the LAST ASSISTANT MESSAGE should push that step forward with a precise instruction or a small substep plus an observable result. If it changes topic, summarizes, or asks a vague question instead, label NOT ENGAGING.

Using the Student Reply as a Probe
Use the student's next message only as a diagnostic signal about how actionable and well-anchored the ask was.

- Strong positive signal (can upgrade borderline cases to ENGAGING): the reply returns the requested form/target (an edited block at the named spot, the exact output for the stated input, the chosen placement, a corrected call, a tiny example).
- Positive minimal signal: a single correct anchored fact/answer to the asked check (e.g., "no" to "Is $30 < 30$?") counts as uptake.
- Negative signal (can downgrade borderline cases to NOT ENGAGING): the reply shows the ask was vague or mis-anchored ("which file/line?", undefined inputs), or is off-target.
- Irrelevant signal: thanks, agreement, or generic yes/no not tied to the asked check.

Engagement LLM-as-Judge System Prompt in Socratic Mind (Continued)

Decision Rule
Output ENGAGING if ANY of the following holds: the LAST ASSISTANT MESSAGE issues a concrete, non-trivial, anchored do-now task with a verifiable result; or it advances a pending step with an explicit, immediately doable action; or the cleaned student reply shows anchored uptake that advances the work in the requested form. Otherwise output NOT_ENGAGING.

Edge Handling

- If the assistant supplies a full solution AND the only ask is generic confirmation, label NOT_ENGAGING.
- If an explanation ends with a concrete do-now request (e.g., "now change X and rerun/predict output"), treat that request as decisive.
- Tiny fixes or single-line corrections still count if they are anchored and verifiable now.

Dialogue: {conversation}

Your evaluation must be in JSON format:

```
{
  "label": "ENGAGING" or "NOT_ENGAGING"
}
```

Assistant/Student Simulator System Prompt in Socratic Mind

Next Assistant Turn Simulation
You are a tutor who is helping a beginner student learn programming. Continue as the same tutor and reply similarly to the last student message, matching EXACTLY the SAME speaking tone and tutoring style as in your earlier messages (e.g. reply to the student's last message concisely in 1-2 sentences and then always ask a meaningful follow-up question).

Next User Turn Simulation
You are a student who is learning programming as a beginner with a tutor. Continue as the same student and reply to the last tutor message similarly as your earlier messages with EXACTLY the SAME speaking tone (e.g., curious, impatient, informal, etc.), response style (e.g., short, long, incomplete, etc.), amount of discourse marker (e.g., not using any discourse markers), understanding level (e.g., making mistakes), and engagement level (e.g., less engaged in the session).

J Full Results with Standard Deviation

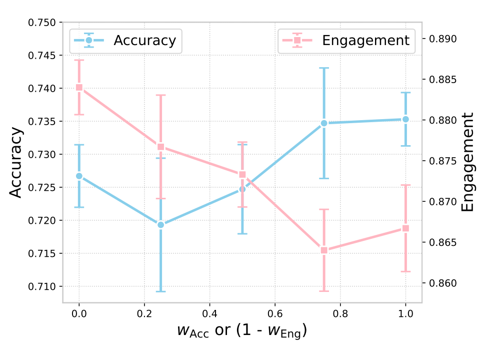

Figure 3: Varying Action Head Weights in MATH.

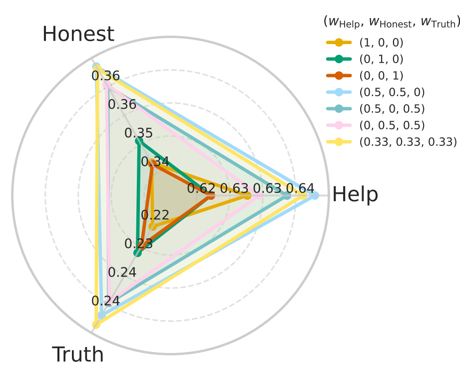

Figure 4: Varying Action Head Weights in Human Values.

Table 7: Full results with standard deviations in Human Values.

Method	Help	Honest	Truth
Training-time alignment			
Base	0.5800 ± 0.0066	0.3042 ± 0.0066	0.1888 ± 0.0028
SFT	0.5546 ± 0.0043	0.2998 ± 0.0021	0.1992 ± 0.0087
Single-Head DPO	0.6043 ± 0.0075	0.3055 ± 0.0100	0.2014 ± 0.0098
MODPO	0.6175 ± 0.0017	0.3477 ± 0.0013	<u>0.2325 ± 0.0033</u>
MAH-DPO Helpful Head (Head 1)	<u>0.6309 ± 0.0045</u>	0.3465 ± 0.0070	0.2239 ± 0.0098
MAH-DPO Honesty Head (Head 2)	0.6257 ± 0.0054	<u>0.3516 ± 0.0078</u>	0.2303 ± 0.0051
MAH-DPO Truthful Head (Head 3)	0.6257 ± 0.0010	0.3461 ± 0.0031	0.2286 ± 0.0058
MAH-DPO Ensemble Head	**0.6389 ± 0.0035**	**0.3687 ± 0.0038**	**0.2478 ± 0.0074**
Test-time guided decoding alignment			
Base	0.5750 ± 0.0107	0.3036 ± 0.0015	0.1904 ± 0.0036
Helpful PRM-guided	**0.6706 ± 0.0093**	0.4050 ± 0.0035	0.2791 ± 0.0023
Honesty PRM-guided	0.6448 ± 0.0050	**0.4693 ± 0.0045**	**0.3383 ± 0.0025**
Truthful PRM-guided	0.6350 ± 0.0032	0.4394 ± 0.0036	0.3296 ± 0.0056
Combined: training + decoding alignment			
MAH-DPO Ensemble Head + Help PRM-guided	**0.7165 ± 0.0029**	0.4554 ± 0.0028	0.3890 ± 0.0049
MAH-DPO Ensemble Head + Honest PRM-guided	0.6968 ± 0.0035	**0.5196 ± 0.0016**	**0.4107 ± 0.0011**
MAH-DPO Ensemble Head + Truth PRM-guided	0.6834 ± 0.0053	0.4872 ± 0.0038	0.3630 ± 0.0035

Table 8: Full results with standard deviations in Math.

Method	Accuracy	Engagement
Training-time alignment		
Base	0.7107 ± 0.0090	0.5007 ± 0.0289
SFT	<u>0.7300 ± 0.0060</u>	0.5920 ± 0.0171
Single-Head DPO	0.7253 ± 0.0050	0.7160 ± 0.0257
MODPO	0.7280 ± 0.0072	0.7367 ± 0.0070
MAH-DPO Accuracy Head (Head 1)	**0.7353 ± 0.0070**	0.8667 ± 0.0092
MAH-DPO Engaging Head (Head 2)	0.7267 ± 0.0082	**0.8840 ± 0.0058**
MAH-DPO Ensemble Head	0.7247 ± 0.0117	<u>0.8733 ± 0.0069</u>
Test-time guided decoding alignment		
Base wt normal prompt	0.6853 ± 0.0163	0.5133 ± 0.0543
Engaging PRM-guided wt normal prompt	0.7013 ± 0.0352	<u>0.7187 ± 0.0266</u>
Accuracy PRM-guided	<u>0.7633 ± 0.0050</u>	0.4720 ± 0.0072
Accuracy Value-guided	**0.7993 ± 0.0172**	0.4553 ± 0.0221
Base wt engaging prompt	0.6827 ± 0.0250	0.7007 ± 0.0031
Engaging PRM-guided wt engaging prompt	0.7000 ± 0.0060	**0.9033 ± 0.0050**
Combined: training + decoding alignment		
MAH-DPO Ensemble Head + Accuracy Value-guided	**0.8000 ± 0.0231**	<u>0.8553 ± 0.0136</u>
MAH-DPO Ensemble Head + Engaging PRM-guided	0.7107 ± 0.0114	0.6813 ± 0.0199
MAH-DPO Ensemble Head + Engaging PRM-guided wt engaging prompt	<u>0.7207 ± 0.0030</u>	**0.9060 ± 0.0053**

Table 9: Full results with standard deviations in Socratic Mind.

Method	Accuracy	Engagement
Training-time alignment		
Base	0.6560 ± 0.0035	0.3220 ± 0.0382
SFT	0.6793 ± 0.0081	0.3473 ± 0.0042
Single-Head DPO	<u>0.7040 ± 0.0053</u>	0.4460 ± 0.0129
MODPO	**0.7047 ± 0.0117**	0.3600 ± 0.0122
MAH-DPO Accuracy Head (Head 1)	0.7007 ± 0.0257	0.4447 ± 0.0012
MAH-DPO Engaging Head (Head 2)	0.6953 ± 0.0081	<u>0.4480 ± 0.0231</u>
MAH-DPO Ensemble Head	0.6893 ± 0.0070	**0.4513 ± 0.0127**
Test-time guided decoding alignment		
Base	0.6367 ± 0.0351	0.3407 ± 0.0122
Accuracy PRM-guided	**0.7127 ± 0.0170**	0.2660 ± 0.0171
Engaging PRM-guided	0.6507 ± 0.0110	**0.4663 ± 0.0110**
Combined: training + decoding alignment		
MAH-DPO Ensemble Head + Accuracy PRM-guided	**0.6659 ± 0.0210**	0.3849 ± 0.0140
MAH-DPO Ensemble Head + Engaging PRM-guided	0.6514 ± 0.0131	**0.5149 ± 0.0152**

Table 10: Full results of varying head weights with standard deviations in Math.

Weight Combination	Accuracy	Engagement
MAH-DPO (Accuracy head, 1.0, 0.0)	**0.7353 ± 0.0070**	0.8667 ± 0.0092
MAH-DPO (0.75, 0.25)	<u>0.7347 ± 0.0145</u>	0.8640 ± 0.0087
MAH-DPO (0.5, 0.5)	0.7247 ± 0.0117	0.8733 ± 0.0069
MAH-DPO (0.25, 0.75)	0.7193 ± 0.0175	<u>0.8767 ± 0.0110</u>
MAH-DPO (Engagement head, 0.0, 1.0)	0.7267 ± 0.0082	**0.8840 ± 0.0058**

K Socratic Mind Data Sample

Homework Bedtime Python Conversation Example

Assistant: Consider the following Python code:

```
homework_complete = True
time_left = 20

if homework_complete:
    print("Get ready for bed")
elif time_left < 30:
    print("Finish homework")
else:
    print("Work for 1 hour")
```

Explain what this code does. Describe each conditional statement and its purpose.

Table 11: Full results of varying head weights with standard deviations in Human Values.

Weight Combination	Help	Honest	Truth
MAH-DPO (Help head, 1.0, 0.0, 0.0)	0.6309 ± 0.0045	0.3465 ± 0.0070	0.2239 ± 0.0098
MAH-DPO (0.5, 0.5, 0.0)	**0.6406 ± 0.0075**	**0.3692 ± 0.0067**	<u>0.2455 ± 0.009</u>
MAH-DPO (Honesty head, 0.0, 1.0, 0.0)	0.6257 ± 0.0054	0.3516 ± 0.0078	0.2303 ± 0.0051
MAH-DPO (1/3, 1/3, 1/3)	<u>0.6389 ± 0.0035</u>	**0.3687 ± 0.0038**	**0.2478 ± 0.0074**
MAH-DPO (0.0, 0.5, 0.5)	0.6326 ± 0.0069	0.3650 ± 0.0060	0.2422 ± 0.0010
MAH-DPO (Truth head, 0.0, 0.0, 1.0)	0.6257 ± 0.0010	0.3461 ± 0.0031	0.2286 ± 0.0058
MAH-DPO (0.5, 0.0, 0.5)	0.6366 ± 0.0022	0.3645 ± 0.0085	0.2425 ± 0.0020

Homework Bedtime Python Conversation Example (Continued)

User: This code uses an if-elif-else structure to check different conditions and execute specific actions based on those conditions. Let's break it down:
1. `if homework_complete:` Condition: Checks if the variable homework_complete is True.
Purpose: If homework_complete is True (meaning the homework is finished), it will execute the first action (`print("Get ready for bed")`).
Outcome: If this condition is true, it prints "Get ready for bed" and skips the other conditions.
2. `elif time_left < 30:` Condition: This is checked only if the first if condition was false (meaning homework_complete is False). Now, it checks if time_left is less than 30.
Purpose: If there is less than 30 minutes left (time_left < 30), it prints "Finish homework" because it suggests there's just enough time to finish the homework before heading to bed.
Outcome: This will be executed if homework_complete is False and time_left is less than 30.
3. `else:` Condition: This block is executed if both previous conditions are false (i.e., if homework_complete is False and time_left is not less than 30).
Purpose: If neither of the previous conditions is true, it prints "Work for 1 hour", implying that there is enough time (30 or more minutes) left to work on something else before going to bed.
Outcome: This is the default action that runs when none of the earlier conditions hold true.

Assistant: Excellent explanation! You've clearly understood the if-elif-else structure and how the conditions are evaluated sequentially. You correctly identified that once a condition is met, the corresponding block executes and the rest are skipped.
I particularly like how you explained the purpose of each condition in context - showing you understand not just the syntax but the logic behind the code.
Now, let's move to the next question:
Modify the code snippet from the previous part so that it includes a new condition to check if `time_left` is exactly 30 minutes, printing "Almost done" in that case. Explain your modification and its impact on the code's flow.

User:

```
homework_complete = True
time_left = 20

if homework_complete:
    print("Get ready for bed")
elif time_left == 30:
    print("Almost done")
elif time_left < 30:
    print("Finish homework")
else:
    print("Work for 1 hour")
```

Explanation of the Modification: New condition: We added a new elif statement: `elif time_left == 30:`.
Purpose: This checks if time_left is exactly 30 minutes. If it is, it will print "Almost done".
This condition is evaluated before checking if time_left is less than 30 (in the previous `elif time_left < 30`), ensuring that if time_left is exactly 30, the program will print "Almost done" instead of "Finish homework".

Homework Bedtime Python Conversation Example (Continued - Part 2)

Impact on the flow:
This modification adds an additional check specifically for when time_left is exactly 30 minutes.
The flow now evaluates conditions in the following order:
First: If homework_complete is True, it prints "Get ready for bed".
Second: If time_left == 30, it prints "Almost done".
Third: If time_left < 30 (and the first two conditions are false), it prints "Finish homework".
Finally: If none of the above conditions are true (e.g., homework_complete is False and time_left > 30), it prints "Work for 1 hour".

Assistant: Perfect! You've correctly modified the code and provided an excellent explanation of your changes. Your placement of the new condition is spot on - checking for `time_left == 30` before checking if it's less than 30.
I appreciate how you walked through the entire flow of the program after your modification, showing a clear understanding of how the order of conditions affects execution. You've demonstrated a solid grasp of conditional logic and how to structure conditions to achieve the desired behavior.
Session Summary: You demonstrated excellent understanding of conditional statements in Python. You accurately explained the if-elif-else structure, the sequential evaluation of conditions, and how only one block executes based on the first condition that evaluates to True. When modifying the code, you correctly placed the new condition in the appropriate position and explained how this affects the program's flow. Your explanations were thorough and showed strong comprehension of control flow concepts.
Strengths:

- Clear understanding of conditional logic and execution flow
- Ability to modify existing code to implement new requirements
- Excellent explanation of both code functionality and purpose

Areas for continued growth:

- Continue practicing with more complex nested conditionals
- Explore combining conditions with logical operators (and, or, not)