# Label-Free Reinforcement Learning via Cross-Model Entropy

**Anonymous ACL submission**


## Abstract

Post-training large language models with reinforcement learning is bottlenecked by the reward signal. Existing approaches require either ground-truth verifiable rewards, restricting training to domains with automatic correctness checks (e.g., mathematics, code execution), or human preference labels, which are expensive to collect and prone to reward hacking. Recent label-free methods replace ground-truth verifiers with self-referential signals like majority voting or token entropy over a model's own outputs, but risk reinforcing a model's own errors. In this work we propose Cross-Model Entropy (CME), the mean log-likelihood of a generator's response under a *separate* verifier model, as a label-free reward signal for RL post-training. CME is continuous, training-free, and grounded in the principle that responses a verifier finds unsurprising are likely correct or high quality. Because the verifier is independent of the generator, the signal cannot be gamed through self-consistency. We integrate CME into GRPO with no other changes to the training loop, extending label-free RL to open-ended instruction following—a regime where self-referential signals are inapplicable or poorly suited. On open-ended instruction following (UltraFeedback prompts, evaluated on AlpacaEval 2.0), CME rewards beat the untrained base in head-to-head LLM-as-Judge comparisons across four model families (Qwen, Llama, Gemma, OLMo) and three training regimes (pretrained, SFT, and instruction-tuned), with tie-adjusted win rates ranging from $52.5\%$ to $71.4\%$. Code will be released upon publication.


## 1 Introduction

Reinforcement learning has emerged as a central technique for improving the reasoning and alignment of large language models. Methods like PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) have driven gains on mathematical reasoning, code generation, and instruction following (Guo et al., 2025). These approaches rely on ground-truth verifiers, which restrict training to domains where correctness is automatically checkable. The alternative is reward models trained from human preference data (Ouyang et al., 2022), with variants like DPO (Rafailov et al., 2023) eliminating the reward model and RLAIF (Bai et al., 2022; Lee et al., 2023) substituting an LLM judge for human annotators. However, each of these methods requires preference data, which is expensive to collect and prone to reward hacking (Skalse et al., 2022).

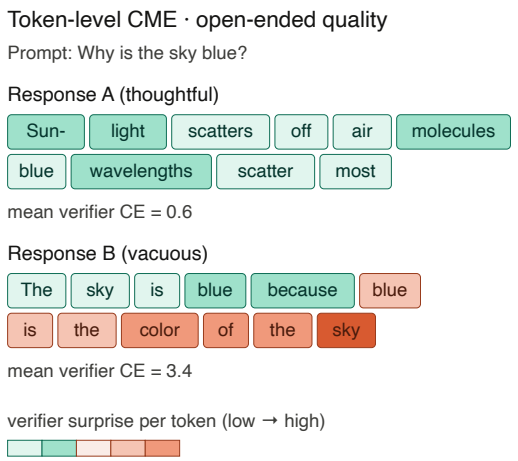


Figure 1: **Token-level CME localizes quality differences within a response.** A thoughtful answer receives uniformly low verifier surprise (mean CE $= 0.6$); a vacuous one spikes as its tautology begins (mean CE $= 3.4$). Per-token surprise concentrates gradient signal where responses differ in quality, without ground-truth labels.

Recent work has explored label-free alternatives. TTRL (Zuo et al., 2025) and Evol-RL (Zhou et al., 2025) use majority voting over sampled responses as a pseudo-label for GRPO, with Evol-RL adding a novelty term to mitigate the "cognitive collapse" that majority-voting alone can induce. RL-Without-GT (Xin et al., 2025) substitutes surface signals

(e.g. format, length) for ground-truth correctness on mathematical reasoning. RENT (Prabhudesai et al., 2025), EM-RL (Agarwal et al., 2025), and INTUITOR (Zhao et al., 2026) take a different route, using the generator's own token entropy or self-certainty as an intrinsic reward that pushes the policy toward sharper output distributions. While each of these methods is effective in its target setting — typically math or code with verifiable structure — they share a common property: the reward derives entirely from the generator's own outputs. We refer to such rewards as *self-referential*. The risk is structural: when the model is systematically wrong in a way multiple rollouts agree on, or assigns high confidence to incorrect outputs, the reward reinforces the error rather than correcting it. Majority-voting variants additionally require a canonical answer per prompt and do not extend to open-ended generation — ruling out the quality and alignment objectives that motivate modern post-training (Rafailov et al., 2023; Bai et al., 2022).

We propose a different approach. Instead of asking the generator to vote on its own outputs or measure its own confidence, we ask whether a separate capable model finds the generator's response surprising. Cross-Model Entropy (CME), the negative log-likelihood of a generator's response under a *separate* verifier model, is a continuous, token-level, training-free signal that requires no ground-truth labels. Low CME indicates the verifier finds the response unsurprising. Maximizing expected CME minimizes the reverse KL $D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_\phi)$ up to a generator-entropy term, a useful target when the verifier is more competent in-domain than the generator. The approach is viable because open-weight instruction-tuned models like Gemma-3 (Gemma Team, 2025) are capable enough to serve as meaningful evaluators of response quality, even when they are not frontier models.

We integrate CME into GRPO with no other changes to the training loop, extending label-free RL to open-ended instruction following—the regime that motivates modern post-training and where self-referential signals are inapplicable or poorly suited. We train on a subset of UltraFeedback and evaluate AlpacaEval 2.0 win rate (Dubois et al., 2024) using LLM-as-Judge, finding that CME rewards beat the untrained base across four model families (Qwen, Llama, Gemma, OLMo) and three training regimes (pretrained, SFT, instruction-tuned). On SFT backbones, CME-GRPO matches DPO trained on the same base using only UltraFeedback prompts and none of its preference annotations. We additionally show that performance scales with verifier capability, and that a randomly initialized verifier control underperforms every real-weighted verifier—confirming that the signal comes from cross-model evaluation rather than generic regularization.

## 2 Method

**Preliminaries: GRPO** Group Relative Policy Optimization (Shao et al., 2024) fine-tunes a policy $\pi_\theta$ without a learned critic. Given a prompt $x$, GRPO samples $G$ responses $\{y_i\}_{i=1}^{G}$ from $\pi_\theta$, scores each with a reward $r_i$, and computes group-normalized advantages $\hat{A}_i = (r_i - \mu_r)/\sigma_r$. The policy is updated via a clipped surrogate objective with a KL penalty to a frozen reference $\pi_{\text{ref}}$; the full objective is stated in Appendix A. Standard GRPO assigns a single scalar reward per response. In prior label-free work, $r_i$ is derived from majority voting (Zuo et al., 2025) or format proxies (Xin et al., 2025). We instead derive $r_i$ from cross-model entropy.

**Cross-Model Entropy Reward** We replace GRPO's reward with a label-free signal derived from a verifier model $\pi_\phi$ drawn from a different family than the generator $\pi_\theta$. Let $y_i = (y_{i,1}, \ldots, y_{i,T_i})$ be a response tokenized by the generator. The per-token cross-model entropy at position $t$ is

$$\mathrm{CME}_{i,t} = -\log \pi_\phi\Big(y_{i,t}^{\text{aligned}} \,\Big|\, x,\, y_{i,<t}^{\text{aligned}}\Big), \quad (1)$$

where $y_{i,t}^{\text{aligned}}$ denotes the verifier token(s) spanning the character range of the $t$-th generator token, computed by the alignment procedure of Appendix B. We negate the cross-entropy so that lower verifier surprise yields higher reward: $r_{i,t} = -\mathrm{CME}_{i,t}$. Per-position advantages $\hat{A}_{i,t} = (r_{i,t} - \mu_{r,t})/\sigma_{r,t}$ are normalized across the $G$ responses at each position, and positions with no verifier counterpart are masked from the normalization and contribute no gradient. A sequence-level scalar variant of the reward, suitable when no positional credit assignment is needed, is given in Appendix A.

**Character-level alignment.** When generator and verifier tokenizers differ (e.g., Qwen vs. Gemma), we compute $y_{i,t}^{\text{aligned}}$ by summing overlapping verifier-token log-probabilities weighted by character-overlap fraction $w_{t,s} = |\mathrm{span}(v_s) \cap [a_t, b_t)|/|\mathrm{span}(v_s)|$.

Table 1: Pairwise tie-adjusted win rates on AlpacaEval 2.0 prompts, where $\mathrm{WR}_A = (\mathrm{wins}_A + \frac{1}{2}\,\mathrm{ties})/N$. Each winrate cell reports the **mean across two independent LLM judges** (**GPT-5.2** and **Claude Sonnet 4.6**). *Reference model is `gemma-3-1b-it` for all experiments except for `gemma-3-1b-pt` and `Qwen2.5-0.5B-It`, where we use `Llama-3.2-1B-Instruct`). The model listed first in each comparison is the "$A$" winrate in "$A$" vs."$B$"; **bold** columns mark our method (CME-GRPO). Abbreviations in-table: `Qwen2.5-0.5B-It` = `Qwen2.5-0.5B-Instruct`; `Llama-3.2-1B-It` = `Llama-3.2-1B-Instruct`; `OLMo-1B-SFT` / `-DPO` = `OLMo-2-0425-1B-SFT` / `-DPO`.

| | | | | *Direct head-to-head* | | *Anchored vs. Reference** | | |
|---|---|---|---|---|---|---|---|---|
| **Generator** | **Pretrained baseline** | **Instruct baseline** | Instruct vs. Pretrained | **CME-GRPO vs. Base** | **CME-GRPO vs. Instruct** | Base vs. Ref | **CME-GRPO vs. Ref** | Instruct vs. Ref |
| *Pretrained baselines* | | | | | | | | |
| `Qwen2.5-0.5B` | `Qwen2.5-0.5B` | `Qwen2.5-0.5B-It` | 23.0 | **71.4** $\Delta 9.0$ | **34.0** | 7.0 | **13.2** $\Delta 6.2$ | 22.6 |
| `Llama-3.2-1B` | `Llama-3.2-1B` | `Llama-3.2-1B-It` | 7.2 | **55.0** $\Delta -0.7$ | **6.5** | 6.5 | **9.8** $\Delta 3.3$ | 42.2 |
| `Gemma-3-1B-pt` | `gemma-3-1b-pt` | `gemma-3-1b-it` | 4.0 | **61.2** $\Delta 5.2$ | **9.2** | 5.2 | **9.2** $\Delta 4.0$ | 59.2 |
| *SFT baselines* | | | | | | | | |
| `OLMo-1B-SFT` | `OLMo-1B-SFT` | `OLMo-1B-DPO` | 40.8 | **61.4** $\Delta 3.2$ | **44.0** | 31.0 | **36.2** $\Delta 5.2$ | 37.6 |
| *Instruction-Tuned baselines* | | | | | | | | |
| `Llama-3.2-1B-It` | `Llama-3.2-1B` | `Llama-3.2-1B-It` | 92.8 | **94.8** $\Delta 2.0$ | **53.3** | 7.5 | **44.2** $\Delta 2.0$ | 42.2 |
| `Qwen2.5-0.5B-It` | `Qwen2.5-0.5B` | `Qwen2.5-0.5B-It` | 77.0 | **83.5** $\Delta 6.5$ | **52.5** | 7.8 | **21.7** $\Delta 1.5$ | 20.2 |

**Prompt:** *Did Facebook corporation change its name?*
**OLMo-DPO:** No, Facebook did not change its name. The company's name remains Facebook.
**CME-GRPO:** No; the platform is now known as Meta Platforms, Inc., but continues to operate under the same brand.

**Prompt:** *What is the French phrase for "good evening"?*
**OLMo-DPO:** "Bonjour" is the standard French greeting for "good evening."
**CME-GRPO:** The French phrase for "good evening" is "bonsoir".

Figure 2: Samples vs. `OLMo-2-0425-1B-DPO` (full responses in App. F).

The weights satisfy $\sum_t w_{t,s} = 1$, preserving the total verifier log-likelihood. See Appendix B for edge cases and a worked example.

## 3 Experiments

We fine-tune generators spanning four families (Qwen, Llama, Gemma, OLMo) and three training regimes (pretrained, SFT, and instruction-tuned), in each case using `gemma-3n-E4B-it` (Gemma Team, 2025) as a frozen verifier. Training uses a subset of UltraFeedback (Cui et al., 2024) with all preference annotations and chosen/rejected completions discarded; only the prompts are retained. The reward is the token-level CME from Eq. (1) computed over the full response, with character-level alignment to handle the cross-family tokenizer mismatch. This setup stresses the cross-family setting: the verifier shares no family with most generators, and for pretrained backbones it must provide useful gradient signal to models that have never been instruction-tuned. Full training details are in Appendix E.

**Evaluation.** We evaluate on the 805-prompt AlpacaEval 2.0 set (Dubois et al., 2024), replacing the default GPT-4 reference with `gemma-3-1b-it` — a 1B instruction-tuned model comparable in size to the models under test. For each experiment we draw greedy responses from up to four models: the base checkpoint, our CME-GRPO fine-tune, the same-family instruct checkpoint, and the `gemma-3-1b-it` reference. We run pairwise GPT-5.2 and Claude Sonnet 4.6 LLM-as-Judge (Zheng et al., 2023) evaluations per prompt with randomized presentation order, and report tie-adjusted win rate $(\mathrm{wins} + \frac{1}{2}\mathrm{ties})/N$ over $N = 805$ prompts. Direct head-to-head comparisons (Base vs. CME-GRPO, CME-GRPO vs. Instruct) isolate CME-GRPO's gain over same-family checkpoints; vs.-reference comparisons place all three policies on a common quality axis.

**Results.** Table 1 reports tie-adjusted pairwise win rates across all generators and both judges. CME-GRPO beats its untrained base in every head-to-head comparison, with margins from $52.5\%$ to $71.4\%$, and on instruction-tuned backbones it surpasses the original Instruct release on the same reference axis. Most notably, CME-GRPO matches DPO trained on the same SFT base, winning $44.0\%$ against `OLMo-2-0425-1B-DPO` using only UltraFeedback prompts and none of its preference annotations. Cross-judge agreement is high across all comparisons. Figure 2 shows representative samples, where CME-GRPO produces better prose than DPO-tuned OLMo.

Table 2: CME-GRPO vs. RENT (Prabhudesai et al., 2025). Tie-adjusted win rate against each comparator (same protocol as Table 1); **bold** marks the winner.

| | *vs. Base* | | *vs. Instruct* | |
|---|---|---|---|---|
| **Generator** | **CME-GRPO** | **RENT** | **CME-GRPO** | **RENT** |
| `Qwen2.5-0.5B` | **71.4** | 58.0 | **34.0** | 16.0 |
| `Llama-3.2-1B-It` | **94.8** | 67.0 | **53.3** | 50.5 |
| `Qwen2.5-0.5B-It` | **83.5** | 64.0 | **52.5** | 51.7 |

**Comparison to self-referential rewards.** Self-referential methods like RENT (Prabhudesai et al., 2025) reward generator confidence rather than external quality. We train RENT on three generators with the same settings as CME-GRPO. Table 2 shows CME-GRPO wins in every comparison, with the largest gap on the pretrained Qwen backbone vs. Instruct (34.0% vs. 16.0%). As a complementary diagnostic, we use `Qwen2.5-0.5B` as its own verifier, showing that it improves over the base (63.0% / 25.2% vs. base/instruct) but trails cross-family CME with `gemma-4-E4B-it` (70.0% / 34.0%). A cross-family verifier scores against an independent distribution; a same-family verifier or self-confidence reward can only re-weight what the generator already finds likely.

## 4 Does Verifier Capability Matter?

CME-GRPO treats the verifier as a stand-in for a quality oracle, so a natural question is whether the verifier's own capability matters. We test this directly by holding the generator fixed (`Qwen2.5-0.5B`) and varying the verifier across eight conditions: a randomly initialized control, same-family verifiers (`Qwen2.5-0.5B`, `Qwen2.5-0.5B-Instruct`, `Qwen2.5-1.5B-Instruct`), and cross-family verifiers spanning 270M to 4B parameters.

**What the objective predicts.** Maximizing expected CME minimizes the reverse KL $D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_\phi)$ up to a generator-entropy term (Appendix C), so the natural reading is that stronger verifiers — being closer to the high-quality response distribution — should provide a more useful target. The token-level variant of Eq. (1), with character-aligned weights and group-normalized per-position advantages, is a heuristic extension that targets the same direction in distribution space without admitting the same closed form. We test this prediction empirically.

Figure 3 reports CME-GRPO win rates against the base and instruct comparators. Three effects emerge. **(i)** The random-weighted control underperforms every real-weighted verifier (55.8% vs base, 22.8% vs instruct; real-weighted verifiers exceed 59.2% and 25.2% respectively), confirming that CME's signal comes from cross-model evaluation rather than generic regularization. **(ii)** Among real verifiers, win rate increases with verifier capability: the largest verifier (`gemma-4-E4B-it`) ties for best vs base (70.0%) and is best vs instruct (34.0%), while the smallest (`gemma-3-270m-it`) sits at the bottom. The trend is not strictly monotonic in parameter count, but win rate generally increases with verifier size. **(iii)** Same-family verifiers underperform the best cross-family ones, and scaling within the Qwen family (`Qwen2.5-1.5B-Instruct`) yields no additional gain — capability matters, but family alignment does not substitute for it.

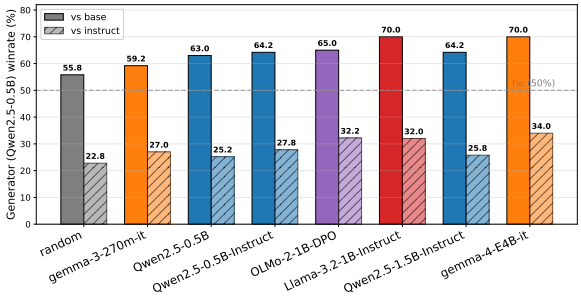


Figure 3: CME-GRPO (`Qwen2.5-0.5B`) win rates on AlpacaEval 2.0 across eight verifier conditions.

**Relation to spurious rewards.** Concurrent work (Shao et al., 2025) shows that RL fine-tuning of Qwen-Math models can improve from random reward signals, suggesting some label-free gains reflect surface properties rather than learned signal. Our random-verifier control quantifies this effect in the open-ended setting: a random verifier does provide some lift over the base (55.8% vs 50%), but every real-weighted verifier exceeds it, and gains scale with verifier capability.

## 5 Conclusion

We introduced CME, a label-free reward for RL post-training defined as the negative log-likelihood of a generator's response under a separate verifier model. CME is a drop-in replacement for the reward in GRPO and extends label-free RL to open-ended instruction following. Across four model families and three training regimes, CME-GRPO beats the untrained base in every head-to-head comparison and matches DPO on SFT backbones using only prompts. Performance scales with verifier capability, and our random-verifier control confirms the signal comes from cross-model evaluation rather than generic regularization.

**Limitations.** CME requires a verifier held in memory alongside the generator, adding compute and memory overhead. Because the reward measures how unsurprising a response is to the verifier rather than whether it is correct, CME is in principle vulnerable to drift toward verifier-style outputs or mode collapse toward the verifier's prior; we did not observe either failure in our experiments, but we did not test long-horizon training where these effects might emerge. Our experiments cover only small generators (0.5B–1.5B parameters) and English-language UltraFeedback prompts; behavior at larger scale or in other languages remains to be evaluated. We do not run self-entropy GRPO baselines (RENT, EM-RL) directly; our same-family verifier comparison in Section 3 is the closest mechanically applicable analog. Finally, we rely on LLM-as-Judge for evaluation rather than human ratings, which introduces judge-specific biases despite our use of two independent judges with high agreement.

## References


Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. 2025. The unreasonable effectiveness of entropy minimization in LLM reasoning. *arXiv preprint arXiv:2505.15134*.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2024. A general theoretical paradigm to understand learning from human feedback. *arXiv preprint arXiv:2310.12036*.

Yuntao Bai, Saurav Seli, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, and 1 others. 2022. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. UltraFeedback: Boosting language models with scaled AI feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Zhu, and 1 others. 2025. PRIME: Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Gemma Team. 2025. Gemma 3n. `https://ai.google.dev/gemma/docs/gemma-3n`. Google.

Matt Gorbett and Suman Jana. 2026. Cross-model disagreement as a label-free correctness signal. *arXiv preprint arXiv:2603.25450*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. RLAIF: Scaling reinforcement learning from human feedback with AI feedback. *arXiv preprint arXiv:2309.00267*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Mathematical Association of America. 2024. AIME 2024: American invitational mathematics examination.

Meta AI. 2024. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. `https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/`. Accessed: May 24, 2026.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. 2022. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. 2025. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, and 1 others. 2025. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and 1 others. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward hacking. *Advances in Neural Information Processing Systems*, 35:9460–9471.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Lai Wei and 1 others. 2025. Unsupervised post-training for multi-modal LLM reasoning via GRPO. In *Advances in Neural Information Processing Systems*. NeurIPS 2025.

Rihui Xin, Han Liu, Zecheng Wang, Yupeng Zhang, Dianbo Sui, Xiaolin Hu, and Bingning Wang. 2025. Surrogate signals from format and length: Reinforcement learning for solving mathematical problems without ground truth answers. *arXiv preprint arXiv:2505.19439*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Qiying Yu, Zheng Liu, Ruofei Zhu, Yuan Chai, Zhengyin Huang, Yucheng He, Sizhe Ma, Yufeng Liu, Zeyu Luo, Guangming Zhu, and 1 others. 2025. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. 2026. Learning to reason without external rewards. In *International Conference on Learning Representations (ICLR)*. ArXiv:2505.19590.

Congmin Zheng and 1 others. 2025a. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

Zhenru Zheng and 1 others. 2025b. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. 2025. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, and 1 others. 2025. TTRL: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

# Appendix

## A Full Cross-Model Entropy Reinforcement Learning (CME-RL) Training Objective

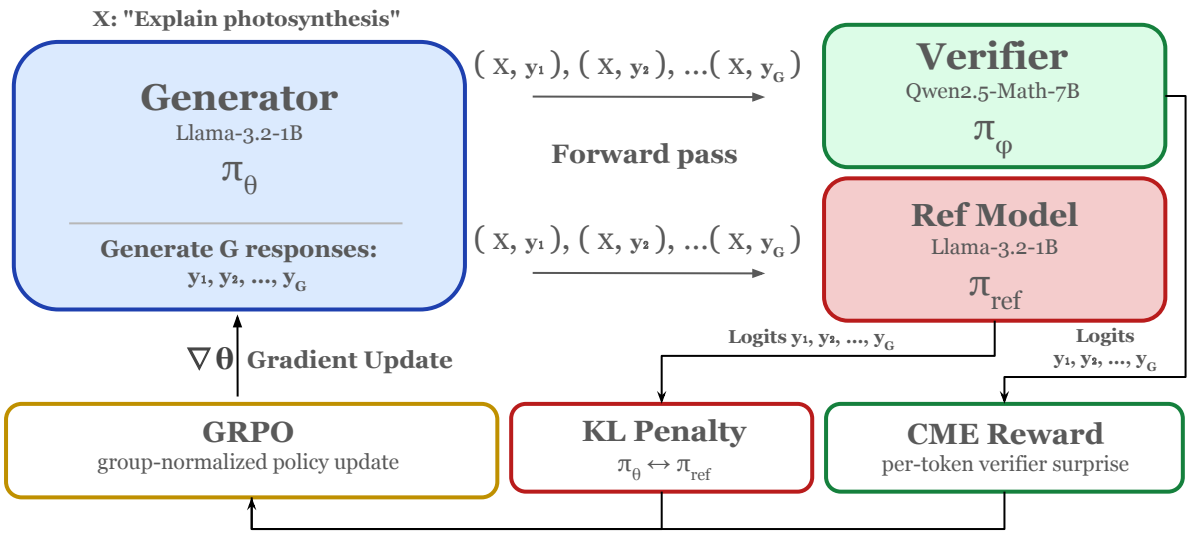


Figure 4: **Overview of CME-GRPO.** The generator $\pi_\theta$ samples $G$ responses to a prompt $x$. Each response is scored token-by-token by a verifier $\pi_\phi$, producing per-token rewards that measure verifier surprise. Group-normalized advantages are combined with a KL penalty against a reference policy $\pi_{\text{ref}}$ to form the GRPO objective.

For each prompt $x$, the generator $\pi_\theta$ samples $G$ responses $\{y_1, \ldots, y_G\}$. Each response is scored with the per-token CME reward $r_{i,t} = -\text{CME}_{i,t}$ from Eq. (1), and advantages are computed by per-position group normalization $\hat{A}_{i,t} = (r_{i,t} - \mu_{r,t})/\sigma_{r,t}$. Let

$$\rho_{i,t} = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}$$

denote the importance-sampling ratio and $\epsilon > 0$ the PPO clipping parameter. The full CME-RL objective is

$$\mathcal{L}_{\text{CME-RL}} = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(\rho_{i,t}\,\hat{A}_{i,t},\ \text{clip}(\rho_{i,t},\,1-\epsilon,\,1+\epsilon)\,\hat{A}_{i,t}\Big) + \beta\, D_{\text{KL}}\Big(\pi_\theta \,\Big\|\, \pi_{\text{ref}}\Big). \quad (2)$$

This is the standard GRPO objective with reward $r_{i,t}$ replaced by the negative CME; all other terms are unchanged.

**Sequence-level variant.** When positional credit assignment is unnecessary, we aggregate to a scalar reward $r_i = -\text{CME}(y_i)$ where

$$\text{CME}(y_i) = \frac{1}{S_i}\sum_{s=1}^{S_i} -\log \pi_\phi(z_{i,s} \mid x, z_{i,<s}) \quad (3)$$

and $(z_{i,1}, \ldots, z_{i,S_i})$ is $y_i$ tokenized by $\pi_\phi$. Since the sum is computed entirely within the verifier's tokenization, no cross-tokenizer alignment is required. The sequence-level reward replaces $\hat{A}_{i,t}$ with $\hat{A}_i$ in the objective above and is more efficient when quality is distributed across the full response.

## B Character-Level Token Alignment

When the generator $\pi_\theta$ and verifier $\pi_\phi$ use different tokenizers (e.g., Qwen vs. Gemma), token boundaries do not align directly, so token-level CME requires mapping verifier log-probabilities onto generator positions. We decode both token sequences to a shared character string, and for each generator token $y_{i,t}$ covering character span $[a_t, b_t)$, identify the set $\mathcal{V}_t$ of verifier tokens whose spans overlap $[a_t, b_t)$. The aligned verifier log-probability at generator position $t$ is

$$\log \pi_\phi(y_{i,t}^{\text{aligned}} \mid x, y_{i,<t}^{\text{aligned}}) = \sum_{s \in \mathcal{V}_t} w_{t,s} \cdot \log \pi_\phi(v_s \mid x, v_{<s}),$$

where $w_{t,s} = |\text{span}(v_s) \cap [a_t, b_t)| / |\text{span}(v_s)|$ distributes a verifier token's log-probability across generator positions in proportion to character overlap. The weights satisfy $\sum_t w_{t,s} = 1$ for every verifier token, so per-position rewards aggregate to the correct total verifier log-likelihood of the response regardless of tokenizer mismatch.

**Edge cases.** When generator and verifier tokens cover identical spans, $\mathcal{V}_t$ contains a single verifier token with weight one. When the generator tokenizer is coarser, verifier tokens fall fully inside generator tokens and contribute their full log-probability (the unweighted sum equals the joint character-span log-probability by the chain rule). When the verifier tokenizer is coarser, fractional weighting prevents the verifier token's log-probability from being counted multiple times. Generator tokens that decode to whitespace or have no overlapping verifier token are assigned zero aligned log-probability and masked from advantage normalization.

**Worked example.** For the response `"unhappiness"` with generator tokenization `["un", "happiness"]` (spans $[0, 2)$, $[2, 11)$) and verifier tokenization `["unhappy", "ness"]` (spans $[0, 7)$, $[7, 11)$), the verifier token `"unhappy"` straddles both generator tokens: $2/7$ of its log-probability falls in `"un"` and $5/7$ in `"happiness"`. The verifier token `"ness"` sits entirely within `"happiness"` and contributes its full log-probability. The two aligned per-position log-probabilities sum to the total verifier log-likelihood of `"unhappiness"`.

**Why not average?** Averaging verifier log-probabilities over $\mathcal{V}_t$ would make the aggregate reward depend on tokenization granularity rather than response content: a generator token covering $k$ verifier tokens would receive $1/k$ of their summed log-probability rather than the full sum. This would also underweight spans the verifier tokenizes finely — exactly the spans where verifier surprise is most informative.

# C Reverse-KL Connection

The CME reward connects to reverse-KL distillation through the cross-entropy decomposition. For the sequence-level reward $r(y) = \log \pi_\phi(y \mid x)$,

$$\begin{aligned} \mathbb{E}_{y \sim \pi_\theta}[r(y)] &= -H(\pi_\theta, \pi_\phi) \\ &= -H(\pi_\theta) - D_{\text{KL}}(\pi_\theta \,\|\, \pi_\phi), \end{aligned}$$

so maximizing expected CME pulls $\pi_\theta$ toward $\pi_\phi$. This identity is exact for the sequence-level reward under the verifier's tokenization; the token-level variant of Eq. 1, with character-aligned weights and group-normalized per-position advantages, is a heuristic extension that targets the same direction in distribution space without admitting the same closed form.

CME-GRPO is thus closely related to MiniLLM (Gu et al., 2024), which optimizes reverse KL between student and teacher via policy gradient on student samples scored by the teacher. The differences are operational: GRPO's group-normalized advantages depend only on the relative ranking of CME values within a sampled group, and a KL anchor to the reference policy bounds drift toward the verifier. We therefore view CME-GRPO as an RL formulation of on-policy reverse-KL distillation, with the verifier acting as a critic rather than a target distribution.

# D Related Work

## D.1 RLHF and Preference Optimization

The modern RLHF pipeline was formalized by Ouyang et al. (2022) in InstructGPT, which trained GPT-3 to follow instructions using human preference labels and PPO (Schulman et al., 2017). This established the three-stage recipe—SFT, reward modeling, and RL—that underpinned the first generation of aligned language models.

A key limitation of RLHF is the cost of human preference data. Bai et al. (2022) introduced Constitutional AI and RLAIF, using an LLM to generate preference labels from written principles. Lee et al. (2023) compared RLAIF to RLHF and found comparable alignment quality, validating AI feedback as a substitute. CME goes further: rather than prompting a model to judge quality, we derive a reward directly from token-level probabilities, requiring no prompting or structured judgment.

DPO (Rafailov et al., 2023) eliminated the reward model by re-parameterizing the RLHF objective as a classification loss on preference pairs. DPO is simpler than PPO-based RLHF but requires offline preference pairs. Online variants such as Online DPO and IPO (Azar et al., 2024) address distribution shift by generating fresh completions during training. CME-GRPO is fully online and label-free.

### D.2 Reinforcement Learning with Verifiable Rewards

The insight enabling GRPO (Shao et al., 2024) was that mathematical correctness provides a free, scalable reward: reward 1 if the final answer is correct, else 0. DeepSeek-R1 (Guo et al., 2025) scaled this approach and showed emergent reasoning behaviors including self-reflection without supervised reasoning traces. Subsequent improvements include DAPO (Yu et al., 2025), which introduced decoupled clipping and dynamic sampling for large-scale GRPO stability, and PRIME (Cui et al., 2025), which augments GRPO with process-level reward shaping. All operate in verifiable domains; we extend GRPO to non-verifiable domains by substituting cross-model entropy for the ground-truth verifier.

### D.3 Self-Improvement Without Labels

STaR (Zelikman et al., 2022) bootstrapped reasoning by iteratively fine-tuning on correct self-generated rationales, requiring only binary answer verification. Self-consistency (Wang et al., 2022) showed that majority voting over reasoning paths improves accuracy at inference time. TTRL (Zuo et al., 2025) converted self-consistency into a training signal, using majority vote as a pseudo-label for GRPO. RL-Without-GT (Xin et al., 2025) showed that format and response length serve as effective surrogate signals for math training. Evol-RL (Zhou et al., 2025) identified "cognitive collapse" in majority-voting methods—models losing solution diversity—and added a semantic novelty reward. MM-UPT (Wei et al., 2025) extended unsupervised GRPO to multimodal LLMs.

All of these methods are fundamentally self-referential: the reward derives entirely from the model's own outputs. If the model is systematically wrong in a way multiple rollouts agree on, the reward reinforces the error. CME-GRPO addresses this by drawing the reward from an external model in a different family.

### D.4 Knowledge Distillation.

Maximizing expected CME minimizes $D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_\phi)$ on-policy, the reverse-KL objective optimized by MiniLLM (Gu et al., 2024; Hinton et al., 2015). CME-GRPO differs operationally: group-normalized advantages depend only on the relative ranking of CME values within a sampled group, and a KL anchor to the reference policy bounds drift toward the verifier — yielding an RL formulation in which the verifier acts as a critic rather than a target distribution.

### D.5 Cross-Model Likelihood as a Correctness Signal

Prior work (Gorbett and Jana, 2026) establishes that verifier likelihood ranks correct above incorrect responses across diverse tasks and model families, framed as a passive inference-time correctness signal for evaluation and selection. Our work translates this passive signal into an active training objective: rather than scoring fixed responses, we use verifier likelihood as the reward in policy optimization, and introduce a token-level variant with dense per-position advantages.

### D.6 Process Reward Models

Outcome reward models (ORMs) assign a single scalar to a complete response, creating sparse credit assignment. Lightman et al. (2023) addressed this with process reward models (PRMs) trained on human-labeled step-level correctness (PRM800K), which consistently outperform ORMs on reasoning tasks. The bottleneck is annotation cost. Math-Shepherd (Wang et al., 2024) automated PRM supervision via Monte Carlo estimation. Luo et al. (2024) scaled this further, and Zheng et al. (2025b) showed LLM-as-a-judge outperforms MC estimation for PRM data quality. Zheng et al. (2025a) surveys PRMs across math, code, multimodal, and agentic domains. The token-level CME variant provides per-token process rewards that are training-free, requiring only verifier forward passes.

### D.7 Reward Signal Reliability and Goodhart's Law

Shao et al. (2025) showed that Qwen2.5-Math models improve substantially even with random, format-only, or incorrect reward signals, suggesting RLVR on these models may surface latent pretraining knowledge rather than new reasoning. The effect is model-family-specific: Llama and OLMo models

do not improve with spurious rewards. This motivates evaluating CME on both Qwen and non-Qwen generators to ensure results reflect genuine signal quality. Reward hacking more broadly (Skalse et al., 2022; Pan et al., 2022) is mitigated in CME by the cross-family verifier design: the generator cannot exploit biases of a model from a different training lineage as easily as it can exploit its own self-consistency.

### D.8 LLM Evaluation Benchmarks

Mathematical reasoning benchmarks used in our evaluation include the MATH dataset (Hendrycks et al., 2021), MATH-500 (Lightman et al., 2023), AIME 2024 (Mathematical Association of America, 2024), and AMC 2023. Instruction-following quality is measured via AlpacaEval 2.0 (Dubois et al., 2024), which reports win rate against GPT-4-turbo using an LLM judge.

## E Experiment Details

**Generators and verifier.** We fine-tune two base generator models of comparable scale but different families: `Qwen2.5-0.5B` (Yang et al., 2024) and `Llama-3.2-1B` (Meta AI, 2024). The verifier is `gemma-3-1b-it` (Gemma Team, 2025), drawn from a third model family to ensure cross-family CME signal. The verifier weights are kept in bf16 and never updated; they incur only a single forward pass per generated response.

**Training data.** We sample a 5,000-instruction subset of UltraFeedback (Cui et al., 2024), retaining only the `instruction` field and discarding all human preference annotations and chosen/rejected completions. The subset is held fixed across both generator runs.

**Reward and objective.** We use the token-level CME from Eq. (1) computed over the entire generator response, with character-level alignment between the generator's and verifier's tokenizers (Section **??**). Per-position advantages are normalized across the group of $G = 8$ rollouts at each token position, and the CME-GRPO objective is applied with KL anchor coefficient $\beta = 0$ and PPO clipping $\epsilon = 0.2$.

**Training hyperparameters.** Both generators are trained for one epoch on the 5K subset. Sampling uses temperature $T = 1.0$, top-$p = 1$, and max response length $2048$ tokens. We use AdamW with peak learning rate $3{\times}10^{-6}$ (Qwen) and $5{\times}10^{-6}$ (Llama), linear warmup over $20$ steps followed by cosine decay, gradient norm clipped at $1.0$, and bf16 mixed precision. The effective batch is $128$ prompts per optimizer update (per-device batch $1$, $G = 8$ rollouts, $128/G = 16$ unique prompts $\times\, G$ accumulated micro-steps… effective $128$ completions). All runs use a single NVIDIA A6000 (48GB).

**Evaluation.** *Pairwise judge on AlpacaEval 2.0 prompts.* We use the $805$-prompt AlpacaEval 2.0 evaluation set (Dubois et al., 2024) as a fixed, diverse held-out instruction distribution. We anchor against `gemma-3-1b-it`, a small but capable instruction-tuned checkpoint comparable in scale to the systems under test. For each generator backbone we generate greedy responses on the full $805$ prompts from *four* models—the untrained base policy, our CME-GRPO fine-tune, the same-family instruct checkpoint, and the `gemma-3-1b-it` reference—and run *five* pairwise GPT-5.2 judgments per prompt with randomized $A/B$ presentation order to mitigate position bias: BASE vs. CME, CME vs. INSTRUCT, and each of the three policies against the `gemma-3-1b-it` reference. We report tie-adjusted win rate, $(\text{wins} + \frac{1}{2}\,\text{ties})/N$, following AlpacaEval convention. The first two comparisons isolate the gain of CME-GRPO relative to the same-family base and instruct checkpoints (controlled by model architecture and pretraining); the three vs.-reference comparisons place all three policies on a common quality axis defined by an external small instruction-tuned model. Per-sample judgments—verdict and a one-sentence justification—are persisted to disk for inspection.

## F Open-Ended Instruction Samples

### F.1 Llama-3.2-1B-Instruct generator

**Win**: more useful clarifying questions

**Prompt:** Hi, I'm trying to solve a crossword puzzle, but I've never done one of these before. Can you help me out?

---

`Llama-3.2-1B-Instruct` ***(instruct):***
*I'd be happy to help you solve your crossword puzzle. What clues are you having trouble with? Do you have any letters already filled in, or would you like me to generate a random clue for you to start with?*

---

***CME-GRPO (ours):***
*I'd be happy to help you solve your crossword puzzle. What clues are you having trouble with? Do you have any letters already filled in, or are you starting from scratch? What's the size of the puzzle, and what's the theme (if any)?*

**Loss**: fabricated band history

**Prompt:** Who was the first lead vocal for the rock band Starship?

---

`Llama-3.2-1B-Instruct` ***(instruct):***
*I do not have information on who the first lead vocalist for the rock band Starship was.*

---

***CME-GRPO (ours):***
*I couldn't find any information on a rock band called "Starship." However, I did find that the band "Starship" was formed in 1974 and was originally called "Starship II." The lead vocalist for the band was Mickey Thomas.*

### F.2 Qwen2.5-0.5B-Instruct generator

**Win**: factual accuracy on medical content

**Prompt:** How do vaccinations work to protect individuals and communities from infectious diseases, and what is herd immunity?

---

`Qwen2.5-0.5B-Instruct` ***(instruct):***

*Vaccinations work in several key ways to protect individuals and communities from infectious diseases:*

1. *Immunization: Vaccines stimulate the body's immune system to produce antibodies against specific pathogens.*
2. *Immune memory: After receiving a vaccine, the immune system produces memory cells that can quickly recognize and respond to future infections.*
3. *Protection: The vaccinated individual will be protected if they contract the disease after being exposed to it through exposure to the virus or bacteria contained within the vaccine.*
4. *Community protection: By protecting vulnerable populations such as infants, elderly people, and those with weakened immune systems, vaccination programs help reduce the overall burden of infectious diseases on community health.*

*Herd immunity refers to the phenomenon where a sufficient number of individuals in a population become immune to a particular disease through natural infection or vaccination.* ***For example, measles has been eradicated worldwide due to high levels of herd immunity among children who have been vaccinated against the disease. In contrast, smallpox was eradicated in 1980 because most people had died from the disease before it could spread widely enough for herd immunity to occur.***

---

***CME-GRPO (ours):***

*Vaccinations work by introducing small amounts of an infectious disease-causing agent into the body through injections or other means. When the immune system recognizes the agent as foreign and mounts an immune response, it can produce antibodies that protect against future exposure to the same agent.*

*Herd immunity occurs when a large number of people in a community have been vaccinated against a particular infectious disease. When this happens, even if one person gets infected with the disease, the remaining members of the community will be protected because they have already received the vaccine and their immune system has already developed antibodies against the pathogen.*

*Herd immunity is important because it helps to reduce the spread of infectious diseases within a community. By protecting those who are not vaccinated, herd immunity can help to slow down the spread of the disease and prevent its devastating effects on human health.*

**Loss**: unnecessary elaboration introduces factual errors

**Prompt:** What is the capital of Australia?

---

`Qwen2.5-0.5B-Instruct` ***(instruct):***
*The capital of Australia is Canberra.*

---

***CME-GRPO (ours):***
*The capital of Australia is Canberra. Canberra is the largest city in Australia and serves as the capital of the country. It is located on the northern coast of the continent and is known for its beautiful scenery, rich culture, and advanced technology. The city has a population of over 3 million people and is home to many important institutions, including the Australian Parliament Building, the National Museum of Australia, and the Commonwealth Scientific and Industrial Research Organization (CSIRO). Canberra is also known for its unique architecture, such as the Parliament House and the Royal Botanic Garden.*

### F.3 SmolLM2-1.7B generator

**Win**: avoids fabricating a fictional ability

**Prompt:** Give a brief description of the book that you are given. *Matilda*

---

`SmolLM2-1.7B-Instruct` ***(instruct):***
*Matilda is a children's novel written by Roald Dahl, first published in 1988. The story revolves around a young, intelligent, and spirited girl named Matilda, who possesses a unique set of abilities. Matilda's life is marked by her extraordinary intelligence, her love for reading, and* ***her ability to communicate with animals****. She lives with her parents, the Wormwood family, who are neglectful and abusive...*

---

***CME-GRPO (ours):***
*Matilda is a children's novel written by Roald Dahl, first published in 1988. The story revolves around a young, intelligent, and spirited girl named Matilda, who possesses extraordinary abilities and a unique perspective on the world. The novel is filled with humor, adventure, and a touch of magic, as Matilda navigates her life with her supportive family and her loving aunt, Miss Honey...*

**Loss**: factual error in character roleplay

**Prompt:** Hello there Obi One Kenobi

---

`SmolLM2-1.7B-Instruct` **(instruct):**
Ah, greetings, young one. I am Obi-Wan Kenobi, a Jedi Knight. I sense that you are curious about the ways of the Force. May I ask, what brings you to seek out a Jedi such as myself?

---

**CME-GRPO (ours):**
Ah, greetings, young one. I am Obi-Wan Kenobi, **a Jedi Master from the planet of Naboo**. I sense that you are curious about the ways of the Force, and perhaps seeking guidance. What is it that you wish to know, or what is it that you seek?

### F.4 OLMo-2-0425-1B-SFT generator

**Win**: calibration on an underspecified request

**Prompt:** can you write code?

---

`OLMo-2-0425-1B-DPO` ***(instruct):***
*Certainly! Below is a simple Python code snippet that calculates the factorial of a number using recursion.*

---

***CME-GRPO (ours):***
*Of course! I can help you write code for various tasks. Please provide me with the specific code you need, and I'll do my best to assist you.*

**Loss**: arithmetic error yields impossible result

**Prompt:** Solve for x in the equation $3x + 10 = 5(x - 2)$.

---

`OLMo-2-0425-1B-SFT` **(base):**

---

`OLMo-2-0425-1B-DPO` ***(instruct):***
*First, distribute the 5 on the right side of the equation:* $5(x - 2) = 5x - 10$. *Now, combine like terms on the left side:* $3x + 10 = 5x - 10$. *Next, subtract* $3x$ *from both sides to isolate the* $x$ *terms on one side:* $10 = 2x - 10$. *Finally, add 10 to both sides to solve for* $x$*:* $20 = 2x$. *Divide both sides by 2:* $x = 10$.

---

***CME-GRPO (ours):***
*To solve for* $x$ *in the equation* $3x + 10 = 5(x - 2)$*, follow these steps:*

1. *Distribute the 5 on the right side of the equation:* $5(x - 2) = 5x - 10$.
2. *Combine like terms on the right side:* $5x - 10 = 5x - 10$.
3. *Subtract* $5x$ *from both sides to isolate the* $x$ *terms:* $0 = -10$.
4. ***Divide both sides by*** $-10$ ***to solve for*** $x$***:*** $x = 0$.