Right Question is Already Half the Answer: Fully Unsupervised LLM Reasoning Incentivization

Qingyang Zhang [1 2] Haitao Wu [1] Changqing Zhang [1] Peilin Zhao [2] Yatao Bian [2 3 †]

Abstract

While large language models (LLMs) demonstrate exceptional reasoning capabilities, existing methods predominantly rely on supervised fine-tuning (SFT) followed by reinforcement learning (RL) on reasoning-specific data. These approaches critically depend on external supervisions–such as labeled reasoning traces, verified golden answers, or pre-trained reward models–which limits scalability and practical applicability. In this work, we propose Entropy Minimized Policy Optimization (`EMPO`), which makes an early attempt at fully unsupervised LLM reasoning incentivization. `EMPO` does not require any supervised information. By continuously minimizing the predictive entropy of LLMs on unlabeled user queries in a latent semantic space, `EMPO` enables purely self-supervised evolution of reasoning capabilities with strong flexibility and practicality.

1. Introduction

Large language models (LLMs) have demonstrated exceptional potential in challenging tasks such as mathematical reasoning (Guan et al., 2025) and code generation (Daya Guo, 2024). A prevailing paradigm for training reasoning LLMs involves firstly performing supervised fine-tuning (SFT) and then reinforcement learning (RL), or iterative combinations of both, applied to reasoning-specific datasets after pretraining (Yang et al., 2024). Unfortunately, these methods typically depend on large-scale reasoning datasets with various forms of supervised information, such as human-labeled reasoning traces, verified golden answers, or an additional pre-trained reward model.

Recent advancements, such as the pioneering work PFPO (Jiao et al., 2024) leverage self-consistency to generate pseudo label. Despite the promising results, the proposed method still necessitates supervision from instruction fine-tuning data and supervision signals from the frontier LLMs to initialize the RL process.

Recent advanced DeepSeek-R1-Zero directly initiating RL from the base model (Guo et al., 2025) and autonomously evolves sophisticated reasoning behaviors such as reflection and self-critic by exploring the reward signals provided by rule-based rewards, i.e., verified golden answers or an additional pre-trained reward model. Our motivation is to devise a fully unsupervised approach for powerful reasoning capability. Specifically, we propose a novel reinforcement learning algorithm termed as Entropy Minimized Policy Optimization (`EMPO`), which incentivizes the reasoning capability of LLMs in a fully unsupervised manner by minimizing their predictive entropy in a latent semantic space. This method optimizes the model to favor reasoning traces yielding consistent answers, enhancing output reliability. The semantic entropy objective we propose to minimize is a well-established measurement of LLMs' uncertainty, which extends beyond mathematical reasoning to free-form question-answering tasks. We further introduce entropy thresholding to filter unreliable reasoning traces, stabilizing the unsupervised training process.

2. Related Work

Self-Supervised and Semi-Supervised Reasoning. To address the dependency on labeled data, several self-supervised and unsupervised methods have emerged. (Huang et al., 2022) propose a self-improvement framework where LLMs generate high-confidence answers using Chain-of-Thought (CoT) prompting and self-consistency, subsequently fine-tuning on these pseudo-labels. However, the performance gains are often limited, and there is a risk of model collapse, as noted in (Shumailov et al., 2024). These methods, while reducing reliance on external labels, still involve supervised fine-tuning steps, contrasting with `EMPO`'s fully unsupervised RL approach.

Entropy Minimization and Semantic Consistency. Entropy minimization is a well-established technique in semi-

[†]Project Leader [1]College of Computer Scinence, Tianjin University [2]Tencent AI Lab [3]National University of Singapore. Correspondence to: Changqing Zhang <zhangchangqing@tju.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

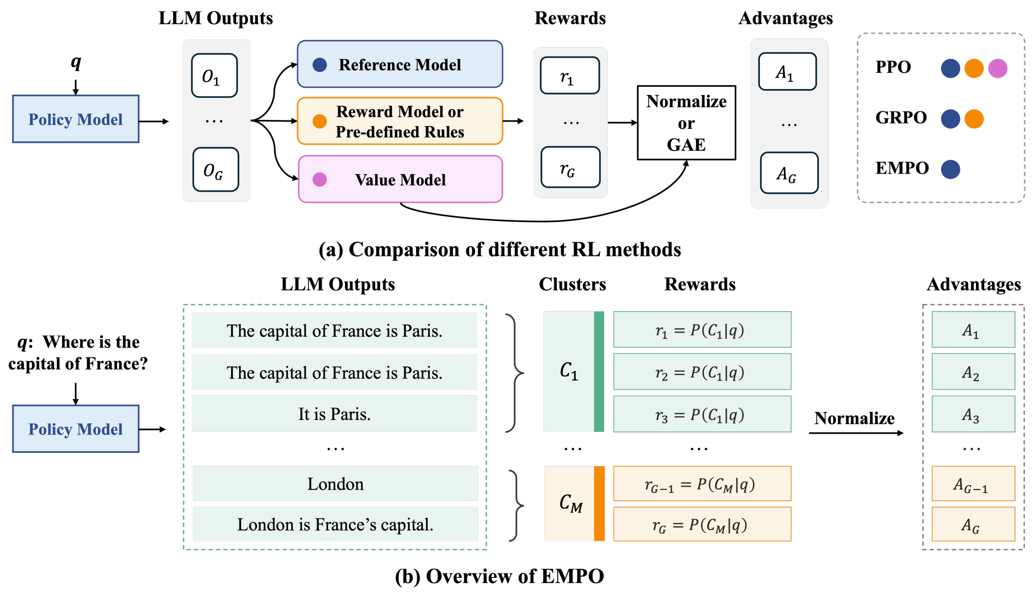

Figure 1. Overview of the proposed method. (a) Previous methods like PPO (Schulman et al., 2017) or GRPO (Shao et al., 2024) typically rely on external supervised signals, e.g., a pretrained reward model or golden answers. (b) The proposed Entropy Minimized Policy Optimization (EMPO) samples a set of responses from the current policy model and then builds semantic clusters according to their equivalence. By continuously minimizing the entropy at a meaning level, our method achieves competitive benchmark performance without any external supervision, i.e., rule-based reward, pre-defined test cases, or a pretrained reward model.

supervised and unsupervised learning, with roots in traditional machine learning. (Grandvalet & Bengio, 2004) demonstrate that minimizing entropy on unlabeled data can improve classification accuracy by encouraging model confidence. Test-time adaptation methods like Tent (Wang et al., 2020; Zhang et al., 2024) adapt models to new domains by minimizing entropy on test data, filling domain gaps without additional labels. These approaches highlight the potential of entropy minimization as an unsupervised objective, which EMPO leverages for LLM reasoning by extending it to semantic entropy (Kuhn et al., 2023) in a latent space. (Farquhar et al., 2024) further validate semantic entropy's utility in detecting hallucinations, reinforcing its relevance.

3. Method

We propose an RL-based method to minimize the entropy of LLM generations in a latent semantic space for incentivizing its reasoning capability. We term our method Entropy-Minimized Policy Optimization (EMPO), which is devised in a fully unsupervised manner without any forms of external supervised information.

3.1. Semantic Entropy Minimization Objective

In this work, we choose semantic entropy (Kuhn et al., 2023) as our unsupervised optimization objective, which is a natural extension of classical Shannon entropy specified for large language models. Specifically, we first samples a group of outputs $\{o_1, \cdots, o_G\}$ and then clusters the output sequences according to their meaning. That is, if two outputs share the same meaning, they should be merged into one same cluster in the semantic space. This can be done without notable computational cost by predefined rules such as N-gram, regular expressions or an additional small language model. Once built such a set of meaning clusters $\{c\}$ in semantic space, we then approximate the probability over the meanings as the proportion of sampled answers as

$$p(c_j|x) \approx |c_j|/G, \tag{1}$$

where $c_j \in \{c\}$ is the j-th meaning cluster. $|c_j|$ denotes the numbers of outputs that belong to c_j. Finally, given question q, the semantic entropy (denoted as H) over the model's output meanings distribution can be estimated as follows:

$$H = -\sum_{c_j \in \{c\}} p(c_j|q) \log p(c_j|q). \tag{2}$$

As proven by previous work, semantic entropy has a strong negative relationship with model accuracy, which can be used as an efficient measurement to detect unreliable LLM generations such as confabulation and hallucination (Kuhn et al., 2023; Farquhar et al., 2024).

3.2. Entropy-Minimized Policy Optimization

We propose Entropy-Minimized Policy Optimization (EMPO), an RL-based method that optimizes the pre-trained large language model π_θ to favor low semantic entropy responses given unlabeled user questions $\{q_i\}_{i=1}^{n}$. Given input questions, EMPO incentivizes the outputs that belong to higher probability meaning cluster, and thus minimizes the semantic entropy over the meaning distribution. Specifically, given a question q, our EMPO first samples a group of output $\{o_1, \dots, o_G\}$ from the current model π_θ and then merges them into a set of M meaning clusters $\{c_1, \dots c_M\}$. As we mentioned before, this can be done without notable computational cost (please refer to the quantitative results in Appendix F) by predefined rules such as N-gram, regular expressions or an additional small language model (SLM). Once built such a meaning set, EMPO approximately minimizes the semantic entropy H by maximizing the following objective

$$\begin{aligned}\mathcal{J}_{\texttt{EMPO}} &= \mathbb{E}_{[\{q\}\sim P(Q), \{o_i\}_{i=1}^{G}\sim\pi_\theta(O|q)]} \\ &\left[\frac{1}{|G|}\sum_{i=1}^{|G|}(\min(A_i, \text{clip}(1, 1-\epsilon, 1+\epsilon)A_i)\right], \quad (3) \\ &\text{s.t.}\ \ \delta_{low} < H < \delta_{high}\end{aligned}$$

where H is the semantic entropy defined in Eq. 2. The questions results in highly unreliable answers with entropy greater than δ_{high} are filtered out. Besides, we also filter out low-entropy answers to maintain the diversity of model outputs and further avoid potential reward hacking. Following previous work (Yu et al., 2025), we remove the KL constraint for better performance. ϵ clips extremely high or low advantages for stability similar to common practice.

4. Experiments

4.1. Experimental Settings

We conduct experiments on multiple datasets including both closed-form math reasoning tasks and free-form natural reasoning tasks. Our EMPO shows competitive performance by purely RL in a fully unsupervised manner compared to supervised finetuning and RL methods.

Prompt Collection and Data Engineering. For mathematical reasoning, following the common practice (Face, 2025; Jiao et al., 2024; Zhang et al., 2025), we adopt 20,000 prompts randomly selected from NuminaMath-CoT dataset (LI et al., 2024) for training without additional data engineering. For free-form natural reasoning tasks, we adopt the prompts from Natural Reasoning, a large-scale dataset consisting of diverse reasoning questions from multiple domains (e.g., Physics, Computer Science, Economics, Social Sciences and more) and select a subset consisted of 18,000 questions. Details about data filtering can be found in Appendix G.

Evaluation. ○ For mathematical reasoning, the performance is evaluated on a diverse suite of benchmarks including Minerva Math, MATH, AMC23, OlympaidBench and AIME24. ○ For free-form natural reasoning, we evaluate the models on MMLU-Pro (Wang et al., 2024) and GPQA (Rein et al., 2024) benchmarks, which consist of challenging reasoning-focused problems across various subjects, e.g., biology, business, chemistry, computer science and so on.

Model training. ○ For mathematical reasoning tasks, we train Qwen2.5-Math-1.5B and 7B Base models with our EMPO. The baselines we consider include supervised finetuning (SFT) and the representative GRPO. We also compared with Qwen2.5-Math Instruction models for a more comprehensive comparison, where the instruction model is trained by iteratively supervised finetuning and RL on private data. ○ For free-form natural reasoning tasks, we initialize from Qwen2.5-3B, 7B and 14B Base models. We consider the corresponding Instruct model, the original Base model with or without few-shot CoT prompt as baselines. Besides, we also compare with SFT where the Base model is tuned to fit the response of Llama3.3-70B-Instruct.

4.2. Main Results

Performance on Mathematical and Natural Reasoning Tasks. We conduct experiments on mathematical tasks to evaluate our method. The main results are shown in Table 1. EMPO has successfully incentivized the Qwen2.5-Math Base model with reasoning capability without dependency on any external supervision. We observe a substantial improvement in the average performance on commonly used mathematical reasoning benchmarks from 28.1% to 42.1% and 30.7% to 48.1% on 1.5B and 7B models, respectively. On the MMLU-Pro benchmark, our EMPO improves the accuracy from 32.1% to 50.1% and 32.7% to 58.8% on Qwen2.5-7B and 14B Base model respectively. Besides, on more challenging GPQA benchmark, EMPO results in increasing accuracy from 15.9% to 28.8% on 7B model, 30.6% to 35.3% on 14B model. Notably, through fully unsupervised RL training, the 1.5B and 7B model has both achieved competitive performance (42.1% and 48.1%) near to Qwen2.5-Math-Instruct (40.5% and 49.4%), where the latter depends on private dataset and multi-stage iteratively supervised fine-tuning and reinforcement learning.

Training Dynamics We further conduct experiments to investigate the reliability of our unsupervised reward signals. As shown in Figure 2, the unsupervised reward signals of EMPO have a strongly negative correlation with the true rewards based on golden answers. Thus by continuously minimizing the semantic entropy objective, the model can boost its accuracy in a fully unsupervised manner.

Table 1. Accuracy on mathematical reasoning benchmarks. We report the pass@1 accuracy tested with greedy decoding. Here q, r, a denote the dependency on questions, human-verified reasoning traces and golden answers respectively.

	Supervision	MATH	Minerva	Olympiad	AIME24	AMC23	Avg.
frontier model							
1.5B model							
Qwen2.5-Math	None	52.2	10.7	25.2	10.0	42.5	28.1
Qwen2.5-Math-Instruct	$\{q, r, a\}$	73.8	30.9	38.7	6.7	52.5	40.5
Qwen2.5-Math w/SFT	$\{q, r, a\}$	61.8	26.1	27.1	3.3	37.5	31.2
Qwen2.5-Math w/GRPO	$\{q, a\}$	75.2	32.0	33.6	16.7	52.5	42.0
Qwen2.5-Math w/EMPO	$\{q\}$	73.0	32.4	36.6	13.3	55.0	42.1
7B model							
Qwen2.5-Math	None	64.8	15.1	26.7	6.7	40.0	30.7
Qwen2.5-Math Instruct	$\{q, r, a\}$	82.8	43.8	41.2	16.7	62.5	49.4
Qwen2.5-Math w/SFT	$\{q, r, a\}$	72.2	34.6	33.2	10.0	45.0	39.0
Qwen2.5-Math w/ODPO	$\{q, a\}$	76.8	30.9	37.9	26.7	62.5	47.0
Qwen2.5-Math w/GRPO	$\{q, a\}$	77.8	39.7	39.1	20.0	57.5	46.8
Qwen2.5-Math w/EMPO	$\{q\}$	78.0	40.4	37.3	20.0	65.0	48.1

Table 2. Accuracy results on free-form natural reasoning benchmarks. We report pass@1 accuracy tested with greedy decoding. Here $\{q, r, a\}$ denote the dependency on questions, human-verfied reasoning traces and verifiable golden answers respectively.

	Supervision	MMLU Pro					GPQA
		STEM	Humanities	Social	Other	Avg.	
7B model							
Qwen2.5-Base	-	30.1	23.8	45.9	34.3	32.1	15.9
Qwen2.5-Base 5-shot	$\{q, r, a\}$	45.7	36.3	59.1	49.4	46.8	23.5
Qwen2.5-Instruct	$\{q, r, a\}$	56.9	38.1	64.1	58.6	55.2	35.3
Qwen2.5-Base w/SFT	$\{q, r, a\}$	32.6	7.1	15.8	30.1	25.6	22.4
Qwen2.5-Base w/GRPO	$\{q, a\}$	57.1	36.2	64.4	56.6	54.5	33.8
Qwen2.5-Base w/EMPO	$\{q\}$	52.4	34.6	59.0	50.9	50.1	28.8
14B model							
Qwen2.5-Base	-	30.8	28.0	44.4	33.0	32.7	30.6
Qwen2.5-Base 5-shot	$\{q, r, a\}$	51.9	35.8	63.4	54.4	51.4	33.2
Qwen2.5-Instruct	$\{q, r, a\}$	63.6	47.1	73.8	66.7	62.9	42.9
Qwen2.5-Base w/SFT	$\{q, r, a\}$	37.0	27.8	40.2	38.0	36.1	28.5
Qwen2.5-Base w/GRPO	$\{q, a\}$	62.9	42.1	68.6	59.8	59.6	35.6
Qwen2.5-Base w/EMPO	$\{q\}$	61.4	41.6	68.3	60.0	58.8	35.3

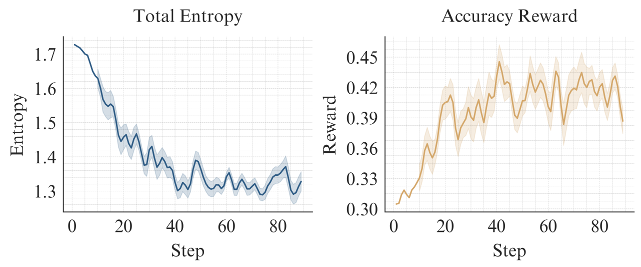

Figure 2. We visualize the training dynamics when tune Qwen2.5-Math-7B with EMPO. The left illustrates the running average of semantic entropy (Eq. 2). Along the unsupervised RL-based training trajectory, EMPO establishes a stable learning process with consistently decreased semantic entropy and improved accuracy.

EMPO's success highlights that intrinsic reward signals, derived purely from the model's objective to minimize semantic entropy and thus achieve greater consistency in its outputs, can be surprisingly potent for this elicitation process. In a well-pre-trained model, outputs that are semantically consistent are more likely to align with correct and coherent reasoning. EMPO leverages this by incentivizing the model to favor such consistent outputs, effectively guiding it to refine its selection from its collection of existing reasoning strategies without requiring external validation of correctness.

5. Conclusions and Future Work

In this work, we make an early attempt at incentivizing the reasoning capability of LLMs by fully unsupervised RL-based techniques. EMPO offers a particularly scalable, cost-effective, and practical approach to unlocking and refining the vast reasoning potential embedded within pre-trained LLMs, especially in domains where curated supervisory data is scarce or prohibitively expensive to obtain. Exploring semi-supervised learning or online learning scenarios would be an interesting future research direction.

References

Daya Guo, Qihao Zhu, D. Y. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

Face, H. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Farquhar, S., Kossen, J., Kuhn, L., and Gal, Y. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

Guan, X., Zhang, L. L., Liu, Y., Shang, N., Sun, Y., Zhu, Y., Yang, F., and Yang, M. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Jiao, F., Guo, G., Zhang, X., Chen, N. F., Joty, S., and Wei, F. Preference optimization for reasoning with pseudo feedback. *arXiv preprint arXiv:2411.16345*, 2024.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

LI, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S. C., Rasul, K., Yu, L., Jiang, A., Shen, Z., Qin, Z., Dong, B., Zhou, L., Fleureau, Y., Lample, G., and Polu, S. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R., and Gal, Y. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zhang, H., Yao, J., Ye, C., Xiong, W., and Zhang, T. Online-dpo-r1: Unlocking effective reasoning without the ppo overhead, 2025. Notion Blog.

Zhang, Q., Bian, Y., Kong, X., Zhao, P., and Zhang, C. Come: Test-time adaption by conservatively minimizing entropy. *arXiv preprint arXiv:2410.10894*, 2024.

Appendices

A. Prompt Templates

We provide the prompt templates used for training and evaluation.

For mathematical reasoning tasks, we adopt the following reasoning prompt template similar with Online-DPO-R1 project (Zhang et al., 2025) for both training and testing.

Mathematical Reasoning Training and Evaluation Template

< |im_start| >system
Please reason step by step, and output your final answer within \boxed{}.
< |im_end| >
< |im_start| >user
{Question} Let's think step by step and output the final answer within \boxed{}.
< |im_end| >
< |im_start| >assistant

To train models with our `EMPO` for free-form natural reasoning tasks, we adopt the following reasoning prompt template similar that we used in mathematical tasks for training.

Free-form Natural Reasoning Training Template

< |im_start| >system
Reason step by step, and output your final answer within \boxed{}.
< |im_end| >
< |im_start| >user
{Question} Reason step by step and output the final answer within \boxed{}.
< |im_end| >
< |im_start| >assistant

Since the MMLU-Pro and GPQA are both close-formed multi-choice benchmark. To evaluate the natural reasoning capability of the models, we use the following prompt template during testing.

MMLU-Pro Test Template

< |im_start| >system
Reason step by step, and output your final answer (the correct letter choice from A-P) within \boxed{}.
< |im_end| >
< |im_start| >user
{Question} Reason step by step and output the final answer (the correct letter choice from A-P) within \boxed{}.
< |im_end| >
< |im_start| >assistant

GPQA Test Template

< |im_start| >system
Reason step by step, and output your final answer (A, B, C, or D) within \boxed{}.
< |im_end| >
< |im_start| >user
{Question} Reason step by step and output the final answer (A, B, C, or D) within \boxed{}.
< |im_end| >
< |im_start| >assistant

B. Case Study

Qwen2.5-Math-7B's Answer to the 18-th Question in AMC23

Question: Janet rolls a standard 6-sided die 4 times and keeps a running total of the numbers she rolls. What is the probability that at some point, her running total will equal 3? The final answer can be written in the form $\frac{m}{n}$, where m and n are relatively prime positive integers. What is $m+n$? Response: To solve this problem, we need to determine the probability that at some point during 4 rolls of a standard 6-sided die, the running total will equal 3. We can use dynamic programming to keep track of the probabilities of each possible running total after each roll. Here's the plan:
1. Initialize a list to keep track of the probability of each possible running total after each roll.
2. Start with the initial condition that the probability of a running total of 0 after 0 rolls is 1.
3. For each roll, update the probabilities of each possible running total by considering the outcomes of rolling a 1, 2, 3, 4, 5, or 6.
4. After each roll, check if the probability of a running total of 3 is non-zero.
5. The final answer will be the probability of a running total of 3 after 4 rolls. Let's implement this in Python:
...
It looks like there was an issue with the implementation. The probability of the running total being exactly 3 after 4 rolls should not be zero, so let's correct the approach and re-implement the solution. Here's the corrected plan:
1. Initialize a list to keep track of the probability of each possible running total after each roll.
2. Start with the initial condition that the probability of a running total of 0 after 0 rolls is 1.
3. For each roll, update the probabilities of each possible running total by considering the outcomes of rolling a 1, 2, 3, 4, 5, or 6.
4. After each roll, check if the probability of a running total of 3 is non-zero.
5. The final answer will be the probability of a running total of 3 after 4 rolls.
Let's re-implement this in Python:
...
It seems there is still an issue with the implementation. Let's manually check the logic and correct it step by step.
1. Initialize the probability list with the initial condition that the probability of a running total of 0 after 0 rolls is 1.
2. For each roll, update the probabilities of each possible running total by considering the outcomes of rolling a 1, 2, 3, 4, 5, or 6.
3. After each roll, check if the probability of a running total of 3 is non-zero.
4. The final answer will be the probability of a running total of 3 after 4 rolls. Let's manually simulate the process: - After 0 rolls: Probability of 0 is 1...The probability of a running total of 3 after 4 rolls is 4/1296, which simplifies to 1/324. Therefore, the final answer in the form of m/n is 1/324, and the sum m + n is 1 + 324 = 325. The final answer is $\boxed{325}$.

C. Implementation details about semantic clustering

We detail the implementation of semantic clustering in our EMPO. The semantic cluster process is shown as Algorithm 1. We also provide the simplified python code to show how we determine whether two model outputs are equivalent or not in mathematical reasoning (Algorithm 2) and free-form natural reasoning tasks (Algorithm 3).

D. Additional Results on Llama3 model series

We conduct additional experiments to validate the efficacy of our EMPO on other model series beyond Qwen2.5. The results are shown in Table 3. Consistent with other concurrent practice, we are unable to implement R1-Zero-like training on the Llama series, i.e., directly initializing RL process from the Base model without SFT). Thus, we instead consider a semi-supervised learning approach by initializing from instruct-tuned model and enhance the reasoning capability with our EMPO.

Algorithm 1 Semantic Clustering

```
Require: question $q$, model responses $\{o_2, \ldots, o_G\}$, verifier $\mathcal{V}$
  $C \leftarrow \{o_1\}$
  for $i = 2$ to $G$ do
    for all $c \in C$ do
      $o_c \leftarrow \text{first}(c)$ {Randomly select one element}
      if $\mathcal{V}(q, o_c, o_i)$ then
        $c \leftarrow c \cup \{o_i\}$ {Add to existing cluster}
        break
      end if
    end for
    $C \leftarrow C \cup \{\{o_i\}\}$ {Create new cluster}
  end for
return $C$
```

Algorithm 2 Implementation of verifier for mathematical reasoning tasks.

```
from math_verify import parse, verify

def are_equivalent(model_output_1, model_output_2)
    prediction_1 = parse(model_output_1)
    prediction_2 = parse(model_output_2)
    return verify(prediction_1, prediction_2)
```

E. Additional training details

F. Computational Cost of Semantic Clustering

To evaluate the additional computational overhead introduced by semantic clustering in EMPO, we conducted comparative analyses of EMPO and GRPO in terms of total training duration and GPU memory utilization. The results of mathematical reasoning and natural reasoning are shown in Table and Table 5, respectively. It is worthy to note that the 14B model experiments requires slightly less computational time than the 7B model. This is because, in our 14B experiments, we reduced the batch size and maximum response length from 2 and 1024 to 1 and 768, respectively, compared to the 3B and 7B configurations. This adjustment was made to fit the limited GPU memory of one single 8×A100 machine.

G. Details of Prompt Collection

For mathematical reasoning, we directly use 20,000 prompts randomly selected from Numina-Math-CoT. For free-form natural reasoning tasks, we adopt the prompts from Natural Reasoning[1] by filtering out the questions with over-long prompt, reference answer. Besides, we use the response length of Llama3.3-70B-Instruct as a difficulty estimation metric, and filter out overly difficult samples with response lengths exceeding 4096 tokens. The data collection python code is demonstrated as follow:

[1]https://huggingface.co/datasets/facebook/natural_reasoning

Algorithm 3 Implementation of verifier for natural reasoning tasks.

```
verifier = AutoModelForCausalLM.from_pretrained(...)
tokenizer = AutoTokenizer.from_pretrained(...)

def are_equivalent(model_output_1, model_output_2, question, verifier)
    prediction_1 = parse(model_output_1)
    prediction_2 = parse(model_output_2)
    prompt = (
            f"User: ### Question: {question}\n\n"
            f"### Ground Truth Answer: {prediction_1}\n\n"
            f"### Student Answer: {prediction_2}\n\n"
            "For the above question, please verify if the student's
               answer is equivalent to the ground truth answer.\n"
            "Do not solve the question by yourself; just check if the
               student's answer is equivalent to the ground truth
               answer.\n"
            "If correct, output \"Final Decision: Yes\". If incorrect,
               output \"Final Decision: No\".\n"
            "Assistant: Final Decision: "
        )
    inputs = self.tokenizer(modified_prompt,
                  return_tensors="pt").to(self.model.device)
    input_ids = inputs.input_ids

    # inference for output logits
    with torch.inference_mode():
        outputs = self.model.forward(input_ids)
    logits = outputs.logits

    # get next output logits
    next_token_logits = logits[0, input_ids.shape[1] - 1, :]

    # get the token ID of "Yes" and "No"
    decision_tokens = self.tokenizer("Yes", "No")
    yes_id = decision_tokens.input_ids[0]
    no_id = decision_tokens.input_ids[1]

    # calculate probability
    probs = torch.softmax(next_token_logits, dim=0)
    yes_prob = probs[yes_id].item()
    no_prob = probs[no_id].item()


    return yes_prob > no_prob
```

Table 3. Accuracy on mathematical reasoning benchmarks.

	Supervision	MATH	Minerva Math	Olympiad Bench	AIME24	AMC23	Avg.
frontier model							
Llama-3.1-70B-Instruct	$\{q, r, a\}$	64.6	35.3	31.9	16.7	30.1	35.7
Eurus-2-7B-PRIME	$\{q, r, a\}$	79.2	38.6	42.1	26.7	57.8	48.9
1B model							
Llama3.2-Instruct	None	27.2	5.1	5.6	0.0	10.0	9.6
Llama3.2-Instruct w/GRPO	$\{q, a\}$	29.8	3.7	6.4	0.0	12.5	10.5
Llama3.2-Instruct w/EMPO	$\{q\}$	31.0	5.1	7.9	3.3	7.5	11.0
3B model							
Llama3.2-Instruct	None	46.2	19.1	15.3	3.3	20.0	20.8
Llama3.2-Instruct w/GRPO	$\{q, a\}$	49.2	22.4	17.6	13.3	32.5	27.0
Llama3.2-Instruct w/EMPO	$\{q\}$	49.8	20.2	18.4	13.3	30.0	26.3

Table 4. Comparison of total runtime (measured as 8× A100 GPU hours) and storage cost (measured by max total GPU memory utilization) between GRPO and EMPO. The GPU Memory semantic cluster process requires minimal computation and storage.

	Qwen2.5-1.5B-Math		Qwen2.5-7B-Math	
	GPU Hours	GPU Memory	GPU Hours	GPU Memory (GiB)
GRPO	11.2	240.4	8.5	501.3
EMPO	11.7	208.2	8.7	532.7

Algorithm 4 Python code of data filtering in a huggingface-like style.

```
from datasets import load_dataset

dataset=load_dataset("facebook/Natural-Reasoning")

filtered_dataset = dataset.filter(
    lambda x: (
        # no answer
        len(x["reference_answer"]) > 0
        # over-long answer
        and len(x["reference_answer"]) < 129
        # overly difficult questions
        and len(x["llama_responses"]) < 4096
        # over-long prompt
        and len(x["question"]) < 512
        # proof-oriented
        and ("prove" not in x["question"].lower())
        and ("proof" not in x["question"].lower())
    )
)
```

H. Additional Result about Pass@k

We provide additional visualization pass@k results of models trained with EMPO. The results are shown as follow.

I. The Influence of Clustering Quality on the Performance of EMPO

In our mathematical reasoning experiments, semantic clustering is achieved solely through regular expression matching without introducing additional models. Due to the naturally structured response formats in mathematical tasks, regular expression could accurately determine answer equivalence, resulting in relatively high clustering quality.

However, in more general free-form natural reasoning tasks where model responses are free-form much more diverse (e.g., matrix, numbers, a few lines of sentences/codes...), the clustering quality can impact EMPO's effectiveness. For instance, in

Table 5. Comparison of total runtime (measured as 8× A100 GPU hours) and storage cost (measured by total GPU memory utilization) between GRPO and EMPO. The GPU Memory semantic cluster process requires minimal computation and storage.

	Qwen2.5-3B		**Qwen2.5-7B**		**Qwen2.5-14B**	
	GPU Hours	GPU Memory	GPU Hours	GPU Memory (GiB)	GPU Hours	GPU Memory
GRPO	9.5	274.8	12.4	508.6	11.0	588.2
EMPO	11.1	286.9	14.6	532.7	11.5	541.1

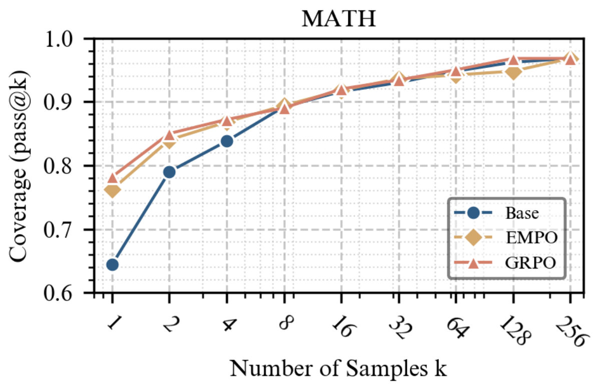

Figure 3. Math

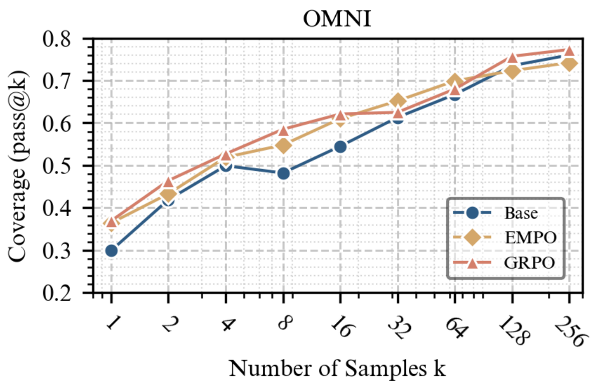

Figure 4. OMNI

our more early practice, we tried DeBERTa (a bert-like model with 300M parameters trained by microsoft) for semantic clustering. Due to the poor quality of semantic clustering, our EMPO straggled to scale up and suffered from frequent reward hacking. Subsequently, by leveraging the general-verifier released by Tiger-Lab (a fine-tuned Qwen2.5-1.5B-Math model) for clustering, we successfully generalized EMPO to more general free-form reasoning tasks. Noted that even though this small language model undergoes supervised finetuning, it serves within our fully unsupervised framework as a fixed utility function for semantic comparison, rather than serving as a external supervisor for task-specific feedback. There are several fundamental difference between cluster model and the reward model used in supervised RL:

- The cluster model does not evaluate output correctness relative to input queries. It just provides pairwise comparisons between the model's own outputs. That is, it only provides binary answer about "whether these two answer is the same?" rather than "which answer is better?".
- The cluster model does not provide any guidance, such as gradient information or hints on how to refine the reasoning traces.
- Compared to reward model or human-verifier golden answers, it can be much easier to implement such a cluster model. For example, in mathematical reasoning tasks, only regular expressions are enough for clustering. In natural reasoning tasks, a finetuned Qwen2.5-1B model can provide high quality semantic cluster results.