LLM2: Let Large Language Models Harness System 2 Reasoning

Cheng Yang[1,2*] **Chufan Shi**[2*] **Siheng Li**[1*] **Bo Shui**[2] **Yujiu Yang**[2] **Wai Lam**[1]
[1]The Chinese University of Hong Kong [2]Tsinghua University
yangc21@mails.tsinghua.edu.cn
Correspondence: sihengli24@gmail.com yang.yujiu@sz.tsinghua.edu.cn

Abstract

Large language models (LLMs) have exhibited impressive capabilities across a myriad of tasks, yet they occasionally yield undesirable outputs. We posit that these limitations are rooted in the foundational autoregressive architecture of LLMs, which inherently lacks mechanisms for differentiating between desirable and undesirable results. Drawing inspiration from the dual-process theory of human cognition, we introduce LLM2, a novel framework that combines an LLM (System 1) with a process-based verifier (System 2). Within LLM2, the LLM is responsible for generating plausible candidates, while the verifier provides timely process-based feedback to distinguish desirable and undesirable outputs. The verifier is trained with a pairwise comparison loss on synthetic process-supervision data generated through our token quality exploration strategy. Empirical results on mathematical reasoning benchmarks substantiate the efficacy of LLM2, exemplified by an accuracy enhancement from 50.3 to 57.8 (+7.5) for Llama3-1B on GSM8K. Furthermore, when combined with self-consistency, LLM2 achieves additional improvements, boosting major@20 accuracy from 56.2 to 70.2 (+14.0)[1].

1 Introduction

Large language models (Brown et al., 2020; Chowdhery et al., 2023; OpenAI, 2023) have exhibited remarkable abilities across various tasks that span general assistance (OpenAI, 2022), coding (Chen et al., 2021), vision (Alayrac et al., 2022) and more. However, they still occasionally produce undesirable outputs in many scenarios, e.g., reasoning and planning (Mialon et al., 2023; Hu and Shu, 2023), factual consistency (Min et al., 2023), and human value alignment (Bai et al., 2022), etc. We hypothesize these deficiencies stem from the fundamental design of LLMs. Specifically, the next-token prediction objective optimizes LLMs to maximize the probability of human-generated strings empirically, with no explicit mechanism to distinguish between desirable and undesirable outputs. During the inference stage, LLMs autoregressively generate outputs token-by-token in a single pass, with no awareness of their errors. This procedure is reminiscent of System 1 in the dual-process theory, which postulates that thinking and reasoning are underpinned by two distinct cognitive systems (Stanovich and West, 2000; Evans, 2003; Kahneman, 2011). System 1 operates automatically and subconsciously, guided by instinct and experience. In contrast, System 2, thought to be unique to humans, is more controlled and rational, enabling deliberate thinking for difficult tasks, especially when System 1 may make mistakes (Sloman, 1996).

In this paper, we introduce LLM2, which aims to empower LLMs with System 2 reasoning. As shown in Figure 1, LLM2 integrates an LLM (System 1) with a process-based verifier (System 2). During inference, the LLM generates multiple candidates at each time step, and the verifier provides timely feedback on each candidate. By efficiently exploring the generation space based on the verifier's feedback, LLM2 ultimately identifies more effective outputs. During the training stage, the process-based verifier is optimized with a pairwise comparison loss to distinguish between desirable and undesirable tokens. To obtain informative token pairs data for process-supervision, we propose a token quality exploration strategy that generate synthetic data based on the potential impact of tokens on the generated text.

We evaluate LLM2 on two representative mathematical reasoning datasets: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). With the integration of System 2 reasoning, LLM2 achieves substantial performance improvement

[*]Equal Contribution. This work was completed during Cheng Yang's time at Tsinghua University.
[1]Code is available at https://github.com/yc1999/LLM2.

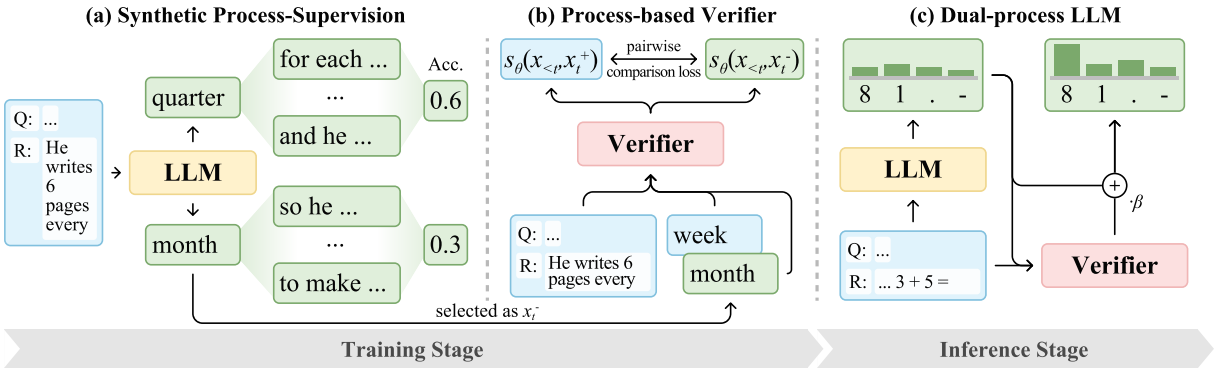

Figure 1: An illustration of the training and inference stages of LLM2. The training stage includes (a) synthetic process-supervision data collection and (b) the optimization of a process-based verifier. The inference stage involves (c) a dual-process LLM for generation.

across Llama3 models ranging from 1B to 8B parameters. For instance, compared to the vanilla Llama3-1B, LLM2 significantly improves accuracy from 50.3 to 57.8 (+7.5) on GSM8K, and from 24.2 to 28.8 (+4.6) on MATH. Combining LLM2 with self-consistency further boosts the model's performance, enhancing major@20 accuracy from 56.2 to 70.2 (+14.0) on GSM8K. Further analysis on the utilization of self-generated answers underscores the effectiveness and promising potential of synthetic process-supervision data.

2 Method

2.1 Dual-process LLM

We aim to build a dual-process LLM (i.e., LLM2), where an LLM serves as System 1 for giving plausible proposals and a verifier functions as System 2 for deliberate thinking, to refine and prevent mistakes introduced by System 1. Specifically, we formalize this procedure as:

$$\log \pi^*(x_t|x_{<t}) \propto \log \pi(x_t|x_{<t}) + \beta s(x_{<t}, x_t), \quad (1)$$

where π and π^* represent the policies of the LLM and dual-process LLM, respectively. The verifier steers π during decoding based on the process score $s(x_{<t}, x_t)$, with β controlling the strength. For computational efficiency, we focus verification on the most probable tokens at each time step. Therefore, we filter out low probability tokens using an adaptive plausibility constraint (Li et al., 2022):

$$\mathcal{V}_t = \{v \in \mathcal{V} : \mathbf{z}_t[v] \geq \log \alpha + \max_w \mathbf{z}_t[w]\}, \quad (2)$$

where $\mathbf{z}_t$ represents the logits of π, $\mathcal{V}$ is the vocabulary and $\mathcal{V}_t \subset \mathcal{V}$ denotes the token set filtered with the hyperparameter $\alpha \in [0, 1]$ at time step t.

Therefore, the logits of π^* at time step t, denoted as $\mathbf{z}_t^*$, are computed as:

$$\mathbf{z}_t^*[v] = \begin{cases} \mathbf{z}_t[v] + \beta s(x_{<t}, v) & \text{if } v \in \mathcal{V}_t, \\ -\infty & \text{otherwise.} \end{cases} \quad (3)$$

The probability distribution $\pi^*(x_t|x_{<t}) = \text{softmax}(\mathbf{z}_t^*)$. This formulation allows π^* to integrates seamlessly with various decoding strategies, depending on the use case.

2.2 Process-based Verifier

We initialize the verifier from an LLM, replacing the unembedding head with a linear head to produce scalar scores. Given a dataset $\mathcal{D} = \{x^i\}_{i=1}^N$, we synthesize process-supervision $\mathcal{D}_p(x) = \{x_{<t}, x_t^+, x_t^-\}_{t=1}^T$ for each instance x, where x_t^+ is more appropriate than x_t^-. Accordingly, the training dataset for the verifier is $\mathcal{D}_s = \{x^i, \mathcal{D}_p(x^i)\}_{i=1}^N$. We train the verifier with a pairwise comparison loss (Ouyang et al., 2022):

$$\mathcal{L}(s_\theta, \mathcal{D}_s) = -\mathbb{E}_{(x, \mathcal{D}_p(x)) \sim D_s} \sum_{t=1}^{T} \left[\log \sigma\left(s_\theta(x_{<t}, x_t^+) - s_\theta(x_{<t}, x_t^-)\right)\right]. \quad (4)$$

2.3 Synthetic Process-supervision

We aim to create $\mathcal{D}_p(x) = \{x_{<t}, x_t^+, x_t^-\}_{t=1}^T$ for each instance x. In particular, we use the ground-truth token x_t as x_t^+, which is desirable to be correct. Regarding x_t^-, our goal is to select tokens that express the undesirable failure modes of LLMs, e.g., reasoning errors, hallucinations and misalignment with human values. Then, through learning to distinguish between x_t^+ and x_t^-, the verifier can discern desirable and undesirable behaviors.

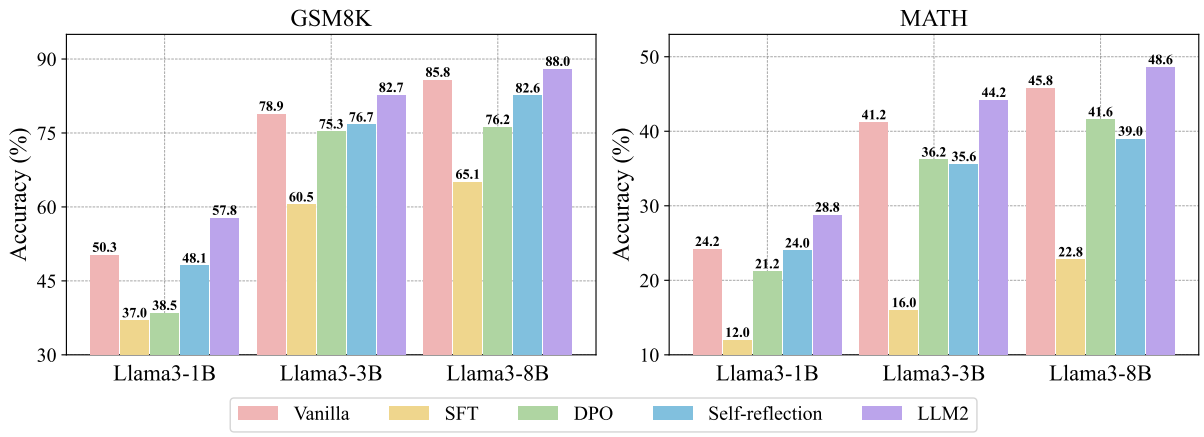

Figure 2: Results of LLM2 and other baselines' performance on GSM8K and MATH with Llama3 series.

To create x_t^-, one can sample tokens from the distributions predicted by LLMs. However, LLMs may assign high probability to alternative correct tokens, which leads to false x_t^- and confuses the training of the verifier. To alleviate this issue, we introduce a token quality exploration strategy for sampling x_t^-. Specifically, the token quality exploration strategy evaluates the quality of individual tokens based on their potential impact on the generated text. This strategy involves three key steps:

Continuation Generation For each candidate token $v \in \mathcal{V} \setminus \{x_t^+\}$ at time step t, we use the LLM to generate N continuations $\{c_j\}_{j=1}^N$, each starting with $x_{<t}$ concatenated with v.

Quality Assessment We evaluate the quality of each continuation based on the correctness of all decoded answers.

$$q(v) = \frac{1}{N} \sum_{j=1}^{N} \text{quality}(c_j), \tag{5}$$

where $\text{quality}(c_j)$ is a function that returns the quality score for each continuation. In this work, we use accuracy as the quality measure.

Negative Sampling We sample x_t^- from tokens with low quality scores:

$$x_t^- \sim \{v : q(v) < \tau, v \in \mathcal{V}_t \setminus \{x_t\}\}, \tag{6}$$

where τ is a threshold hyperparameter.

The token quality exploration strategy enables the identification of tokens likely to lead to low-quality outputs, providing informative negative examples for training the verifier. In this work, we consider the top-k most probable tokens according to the LLM's distribution as candidate set, which reduces the computational cost while still captures the most relevant candidates for x_t^-.

3 Experiments

3.1 Experimental Setup

Our experiments are based on the Llama3 model series, specifically using 1B, 3B and 8B instruct versions (Dubey et al., 2024). We leverage these LLMs as the System 1, and utilize them to initialize corresponding verifiers. We use the GSM8K training set as $\mathcal{D}$, and employ the LLMs to generate corresponding synthetic datasets $\mathcal{D}_s$ for training verifiers. For evaluation, we utilize two benchmarks: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). Further details regarding our experimental setup can be found in Appendix A.

3.2 Results

We present a comprehensive comparison of LLM2 against standard vanilla models and various pivotal baselines, including Self-reflection prompting (Madaan et al., 2024), Supervised Fine-tuning (SFT), and Direct Preference Optimization (DPO) (Rafailov et al., 2024). Further elaborations on these baselines are available in Appendix B. As depicted in Figure 2, implementing self-reflection prompting to engage the model in System 2 reasoning does not yield performance enhancements, suggesting a prevailing limitation in self-reflective capabilities for Llama3 models across different scales (1B, 3B, and 8B). Given that Llama3 has undergone extensive post-training with meticulously curated mathematical reasoning data (Dubey et al., 2024), applying GSM8K for either SFT or DPO

Task	Vanilla	LLM2	
		w/ Ground Truth	*w/* SA
GSM8K	50.3	57.8 (+7.5)	**59.7** (+9.4)
MATH	24.2	28.8 (+4.6)	**30.2** (+6.0)

Table 1: Results of using ground truth or self-generated answers (SA) for LLM2's synthetic process-supervision on GSM8K and MATH using Llama3-1B.

training results in performance degradation across both GSM8K and MATH benchmarks. Conversely, LLM2 emerges as an effective approach to enhance Llama3's performance across different model size. Llama3-1B exhibits an increase from 50.3 to 57.8 (+7.5) on GSM8K, while Llama3-8B progresses from 85.8 to 88.0 (+2.2). Moreover, LLM2 demonstrates robust generalization capabilities, with improvements on MATH despite the process-based verifier's training on GSM8K. Specifically, Llama3-1B rises from 24.2 to 28.8 (+4.6) on MATH, and Llama3-8B advances from 45.8 to 48.6 (+2.6).

4 Analysis

4.1 Self-generated Answers for Synthetic Process-supervision

We further refine our methodology by utilizing the model's self-generated correct answers as $\mathcal{D}$, replacing traditional golden solutions to formulate $\mathcal{D}_s$ for training verifiers. Instances that remain uncorrect after multiple samplings are excluded. Our experiments with Llama3-1B, as illustrated in Table 1 indicate that crafting $\mathcal{D}$ from self-generated data enhances the efficacy of LLM2. On GSM8K, performance heightens from 57.8 to 59.7, marking an improvement of 9.4 over the vanilla model. On MATH, results improve from 28.8 to 30.2, signifying a 6.0 increase over the baseline.

4.2 Self-consistency

We investigate the potential of integrating LLM2 with self-consistency (Wang et al., 2022), with detailed setup provided in Appendix C. As demonstrated in Figure 3, experiments conducted on Llama3-1B unveil that LLM2, when amalgamated with self-consistency, notably enhances performance. LLM2 trained with self-generated data (i.e., LLM2-SA) elevates Major@20 accuracy on GSM8K from 56.2 to 72.2, and on MATH, the Major@20 accuracy improves from 32.8 to 37.0.

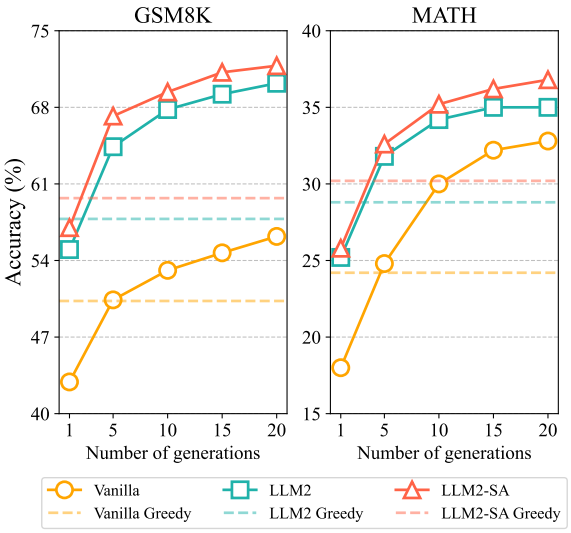

Figure 3: Results on combining LLM2 with self-consistency on GSM8K and MATH using Llama3-1B.

Method	Latency		
	1B	3B	8B
VANILLA	2.8 (× 1.00)	4.8 (× 1.00)	5.3 (× 1.00)
w/ LLM2	3.5 (× 1.25)	5.9 (× 1.23)	6.4 (× 1.21)

Table 2: Averaged per-instance decoding latency of LLM2 in seconds (s/example) on GSM8K.

4.3 Latency

We assess the impact of LLM2's decoding latency and compare with vanilla models on Llama3 model series. Specifically, as shown in Table 2, we report the averaged per-instance inference latency on GSM8K. Since the process-based verifier in LLM2 only performs inference when the LLM provides multiple candidate tokens after the adaptive plausibility constraint, LLM2 introduces an additional 1.21x to 1.25x latency. This latency tends to decrease as the modes's parameters increase.

4.4 Comparison with PRM Method

We compare LLM2 with Math-Shepherd (Wang et al., 2024), a representative Process Reward Model (PRM) baseline for Llama3-1B, with the results presented in Table 3. For a fair comparison, we use the GSM8K subset[2] to train a Llama3-1B PRM model as the baseline. The results show that Math-Shepherd's performance converges at Best-of-N (N=20), achieving 57.6 and 27.0 on GSM8K and MATH, respectively, while LLM2 achieves 59.7 and 30.2, demonstrating LLM2's ad-

[2] https://huggingface.co/datasets/peiyi9979/Math-Shepherd

Task	Math-Shepherd (Best-of-N)				LLM2
	5	10	15	20	
GSM8K	51.6	54.4	56.0	57.6	**59.7**
MATH	26.4	27.2	27.0	27.0	**30.2**

Table 3: Performance comparison between Math-Shepherd (Best-of-N) (Wang et al., 2024) and LLM2 on GSM8K and MATH using Llama3-1B.

Task	Vanilla	SFT	DPO	Self-reflection	LLM2
GSM8K	69.2	56.0	60.3	68.7	**73.5** (+4.3)
MATH	46.4	22.8	38.6	43.8	**49.0** (+2.6)

Table 4: Results of LLM2 and other baselines' performance on GSM8K and MATH with Qwen2.5-1.5B.

vantages. Additionally, using PRM's Best-of-N for inference potentially introduces an N-fold latency, whereas LLM2 only incurs approximately 1.2x latency. This demonstrates the advantage of LLM2's token-level supervision signals (Lin et al., 2024), which enable more efficient and precise optimization during the generation process.

4.5 Employ Qwen2.5

We further investigate the generalizability of LLM2 across diverse LLM families, conducting experiments on the Qwen2.5-1.5B model (Team, 2024). As illustrated in Table 4, LLM2 emerges as a robust approach to enhance the performance of Qwen2.5-1.5B on both the GSM8K and MATH benchmarks. Specifically, compared to the vanilla model, LLM2 achieves notable improvements in mathematical reasoning, with performance gains of 4.3 and 2.6 on GSM8K and MATH, respectively. In contrast, other methods fail to surpass the vanilla baseline, highlighting the unique efficacy of LLM2. This aligns with our observations on the Llama3 model series, where LLM2 consistently enhanced performance across different model sizes and tasks, reinforcing its potential as a universal enhancement framework for different LLM families.

5 Related Work

Verifier for LLMs. Training verifiers to explicitly distinguish between desirable and undesirable outputs has been a promising method to improve the capabilities of LLMs. Existing verifier modeling can be broadly classified into two categories: (1) Outcome-based modeling (Shen et al., 2021; Cobbe et al., 2021), which train verifiers to learn how to distinguish between correct and wrong outputs and selects more optimal ones from a number of candidates at inference time. (2) Process-based modeling (Uesato et al., 2022; Lightman et al., 2023), which supervises each reasoning step of the generation process. To alleviate the reliance on human-annotated process-supervision data, Wang et al. (2024) propose to automatically construct process-supervision data, where the correctness of a mathematical reasoning step is defined as its potential to reach the final answer correctly.

In LLM2, we propose a process-based verifier to emulate System 2 reasoning. It is trained on synthetic process-supervision data generated by our token quality exploration strategy. During inference, this verifier can intervene at any time step, providing immediate feedback without waiting for the completion of specific steps or the entire output.

System 2 for LLMs. Recent works explore the incorporation of System 2 into LLMs, primarily during the inference stage (Weston and Sukhbaatar, 2023; Deng et al., 2023; Saha et al., 2024). These approaches often leverage System 2 mechanisms, such as reflection and planning (Madaan et al., 2024), to generate explicit and verbalized reasoning content, which then guides subsequent token generation. Alternatively, some research focuses on transferring System 2 capabilities to System 1 during the training phase through methods such as distillation (Yu et al., 2024), thereby obviating the need for generating intermediate reasoning tokens during the inference stage.

LLM2 integrates System 2 during the inference-stage. Specifically, LLM2 leverages a process-based verifier as the System 2 to provide real-time feedback at each token generation step without generating auxiliary content.

6 Conclusion

In this work, we introduce LLM2, a framework that augments LLMs with a System 2-like reasoning process. By coupling an LLM with a process-based verifier, LLM2 proficiently differentiates between optimal and suboptimal outputs. The framework is empowered by synthetic process-supervision data generated via a novel token quality exploration strategy, which is instrumental in training the verifier. Our empirical results and analyses confirm the efficacy of LLM2 in enhancing LLM performance.

Limitations

While LLM2 demonstrates significant improvements in mathematical reasoning tasks, our exploration does not extend to other reasoning domains such as commonsense reasoning and code generation, due to computational resource constraints. We are optimistic about the potential of LLM2 to generalize well to these additional tasks. However, applying LLM2 to open-ended tasks, like creative writing, presents challenges due to the lack of definitive supervisory signals for synthetic process-supervision. Addressing these challenges offers a promising direction for future research.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736.

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

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

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

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jonathan St BT Evans. 2003. In two minds: dual-process accounts of reasoning. *Trends in cognitive sciences*, 7(10):454–459.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Zhiting Hu and Tianmin Shu. 2023. Language models, agent models, and world models: The law for machine reasoning and planning. *arXiv preprint arXiv:2312.05230*.

Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *CoRR*, abs/2305.20050.

Zicheng Lin, Tian Liang, Jiahao Xu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. 2024. Critical tokens matter: Token-level contrastive estimation enhence llm's reasoning capability. *arXiv preprint arXiv:2411.19943*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023.

Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 12076–12100. Association for Computational Linguistics.

Sean O'Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. 2024. Branch-solve-merge improves large language model evaluation and generation. 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 8345–8363.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2269–2279. Association for Computational Linguistics.

Chufan Shi, Cheng Yang, Xinyu Zhu, Jiahao Wang, Taiqiang Wu, Siheng Li, Deng Cai, Yujiu Yang, and Yu Meng. 2024a. Unchosen experts can contribute too: Unleashing moe models' power by self-contrast. *arXiv preprint arXiv:2405.14507*.

Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. 2024b. A thorough examination of decoding methods in the era of llms. *arXiv preprint arXiv:2402.06925*.

Steven A Sloman. 1996. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3.

Keith E Stanovich and Richard F West. 2000. 24. individual differences in reasoning: Implications for the rationality debate? *Behavioural and Brain Science*, 23(5):665–726.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

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*.

Jason Weston and Sainbayar Sukhbaatar. 2023. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

A Experimental Setup

Dataset. We leverage the training set of GSM8K (Cobbe et al., 2021) as $\mathcal{D}$ and use the test set of GSM8K as one of our evaluation set. Although we do not use the MATH (Hendrycks et al., 2021) train set to train the verifier, we utilize the MATH test set as an additional evaluation set to validate the effectiveness of the verifier in improving general mathematical reasoning. Due to computational resource constraints, we randomly sampled 500 examples from the original MATH test set for our evaluation.

Hyperparameter Setting. We generally set β to 0.25 in Equation 1, α to 0.1 in Equation 2 and τ to 0.5 in Equation 6. We set N to 20 in Equation 5. For top-k in Section 2.3, k is set to 5.

Model Details. We list the Llama3 and Qwen2.5 models used in our experiments along with their corresponding HuggingFace model names in Table 5.

Model	HuggingFace Model Name
Llama3-1B	`meta-llama/Llama-3.2-1B-Instruct`
Llama3-3B	`meta-llama/Llama-3.2-3B-Instruct`
Llama3-8B	`meta-llama/Llama-3.1-8B-Instruct`
Qwen2.5-1.5B	`Qwen/Qwen2.5-1.5B-Instruct`

Table 5: Llama 3 and Qwen2.5 models and their corresponding HuggingFace model names.

Details of Training Verifiers. We train our verifiers using 8 NVIDIA A100 80GB GPUs. The training process is conducted over 3 epochs with a batch size of 128. We employ a learning rate of 2e-5 and utilize a cosine learning rate scheduler.

B Baselines

We implement four representative baselines:

Vanilla utilizes the original Llama model directly for inference.

Supervised Fine-tuning (SFT) fine-tunes LLMs to maximize the log-likelihood of the training data, which in our case is the GSM8K training set. The training process is conducted over 3 epochs with a batch size of 128. We employ a learning rate of 2e-5 and utilize a cosine learning rate scheduler.

Direct Preference Optimization (DPO) (Rafailov et al., 2024) optimizes language models directly from desirable and undesirable outputs, eliminating the need for an explicit reward model. For desirable data, we use the GSM8K training set; for undesirable data, a randomly sampled incorrect output from the model serves as the undesirable example. The training process is conducted over 1 epoch with a batch size of 128. We set $\beta = 0.01$ and employ a learning rate of 5e-7 and utilize a cosine learning rate scheduler.

Self-reflection Prompting (Madaan et al., 2024) involves first generating an output, followed by prompting the model to assess whether its output is correct and whether to revise the output. This approach can be seen as introducing System 2 reasoning through prompting. The specific prompt is shown in Table 6.

```
Please review your answer. If you think it is
  correct, just repeat your answer. If you think
  it is incorrect, please generate the correct
  one.
```

Table 6: Prompt for Self-reflection prompting.

C Self-consistency Setup

For vanilla self-consistency, we use temperature sampling with temperature $\tau = 1.0$ for instruct models to reach the best baseline performance (Shi et al., 2024b). For combining LLM2 with self-consistency, we simply set β to 0.25 in Equation 1, α to 0.1 in Equation 2 and do temperature sampling with temperature $\tau = 1.0$.

D Comparison with Token-Level Decoding Methods

To further demonstrate the effectiveness of our process-based verifier, we compare LLM2 with token-level decoding methods. Specifically, we implement contrastive decoding (CD) (Li et al., 2022) and DoLa (Chuang et al., 2023), and evaluate their performance on the GSM8K and MATH datasets. The results are shown in Tables 7 and 8.

For CD, we follow the hyperparameter settings from Li et al. (2022); O'Brien and Lewis (2023); Shi et al. (2024a), using Llama3-1B as the amateur model. For DoLa, we follow the hyperparameter settings from Chuang et al. (2023); Shi et al. (2024b). The results reported for both CD and DoLa represent their best performance across their hyperparameter ranges. As shown, CD does not yield significant improvements, primarily because

CD requires an ideal amateur model (O'Brien and Lewis, 2023; Shi et al., 2024b) which may not always exist. As for DoLa, while it proves effective for factual knowledge tasks, it can have adverse effects on reasoning tasks (Chuang et al., 2023; Shi et al., 2024b).

Model	Vanilla	CD	DoLa	LLM2
Llama3-1B	50.3	-	47.2	**57.8**
Llama3-3B	78.9	79.8	76.1	**82.7**
Llama3-8B	85.8	86.4	83.0	**88.0**

Table 7: Results of token-level decoding methods on GSM8K with Llama3 series.

Model	Vanilla	CD	DoLa	LLM2
Llama3-1B	24.2	-	23.6	**28.8**
Llama3-3B	41.2	42.0	39.6	**44.2**
Llama3-8B	45.8	46.4	43.2	**48.6**

Table 8: Results of token-level decoding methods on MATH with Llama3 series.

E Accuracy of Process-based Verifier

We further analyze the accuracy of LLM2's process-based verifier in distinguishing between ground-truth and non-ground-truth tokens. Specifically, using the GSM8K test set, we pair each question q with its answer a. Then we leverage the vanilla models to perform next-token prediction tasks on $(q, a_{<t})$ and collect the non-ground-truth token with the highest probability as $\tilde{a}_t$. Subsequently, we input $(q, a_{<t}, a_t)$ and $(q, a_{<t}, \tilde{a}_t)$ into the corresponding verifier. A correct prediction is determined by whether the verifier assigns a higher score to $(q, a_{<t}, a_t)$. The results, presented in Table 9, demonstrate the verifier's effective token-level accuracy.

F Case Study

We present two representative cases from GSM8K using Llama3-1B to demonstrates how LLM2 improves mathematical reasoning in Table 10 and 11.

In Case 1, LLM2 demonstrates its ability to prevent computational errors. While the vanilla model made an arithmetic error in calculating weekly egg production ($252 \times 7 = 1754$), LLM2 correctly computed 1764 eggs per week, leading to the accurate final answer of 294.

Model	Llama3-1B	Llama3-3B	Llama3-8B
Accuracy	75.5	77.0	79.4

Table 9: Accuracy of LLM2 verifier (1B, 3B and 8B) on GSM8K for the corresponding Llama3 model series.

In Case 2, LLM2 shows how it prevents logical errors. The vanilla model overlooked Terry's daily consumption of 2 yogurts, while LLM2 correctly accounted for both the unit price (1.25) and total consumption (60 yogurts over 30 days), yielding the correct answer of 75.

These cases demonstrate how LLM2's verification mechanism helps maintain both computational and logical accuracy throughout the reasoning process.

Question	Lloyd has an egg farm. His chickens produce 252 eggs per day and he sells them for $2 per dozen. How much does Lloyd make on eggs per week?
Vanilla	## Step 1: Calculate the number of eggs produced per week Since there are 7 days in a week, Lloyd's chickens produce 252 eggs per day * 7 days = 1754 eggs per week. ## Step 2: Convert the number of eggs per week to dozens Since there are 12 eggs in a dozen, Lloyd produces 1754 eggs / 12 = 146.8333 dozens of eggs per week. ## Step 3: Calculate the total amount of money Lloyd makes per week Lloyd sells each dozen of eggs for $2, so he makes 146.8333 dozens * $2 = $292.6666 per week. ## Step 4: Round the total amount of money Lloyd makes per week to the nearest dollar Rounding $292.6666 to the nearest dollar gives $293. The final answer is: $\boxed{293}$
LLM2	## Step 1: Calculate the number of eggs produced per week To find out how many eggs Lloyd produces per week, we need to multiply the number of eggs produced per day by 7. eggs_per_day = 252, eggs_per_week = eggs_per_day * 7 = 252 * 7 = 1764. ## Step 2: Calculate the number of dozens of eggs produced per week Since there are 12 eggs in a dozen, we need to divide the total number of eggs produced per week by 12 to find out how many dozens of eggs Lloyd produces per week. dozens_per_week = eggs_per_week / 12 = 1764 / 12 = 147. ## Step 3: Calculate the total amount of money Lloyd makes per week To find out how much Lloyd makes per week, we need to multiply the number of dozens of eggs produced per week by the price per dozen. price_per_dozen = $2, total_money_per_week = dozens_per_week * price_per_dozen = 147 * 2 = 294. The final answer is: $\boxed{294}$

Table 10: A case study from GSM8K using Llama3-1B, where LLM2 corrects the vanilla model's arithmetic error.

Question	Terry eats 2 yogurts a day. They are currently on sale at 4 yogurts for $5.00. How much does he spend on yogurt over 30 days?
Vanilla	## Step 1: Calculate the cost per yogurt The cost per yogurt is $5.00 / 4 yogurts = $1.25 per yogurt. ## Step 2: Calculate the total cost for 30 days The total cost for 30 days is $1.25 per yogurt * 30 days = $37.50. The final answer is: $\boxed{37.5}$
LLM2	## Step 1: Calculate the cost of one yogurt To find out how much Terry spends on yogurt per day, we need to calculate the cost of one yogurt. The current sale price of 4 yogurts for $5.00 is $5.00 / 4 = $1.25 per yogurt. ## Step 2: Calculate the cost of yogurt over 30 days Now that we know the cost of one yogurt, we can calculate the total cost of yogurt over 30 days. Terry eats 2 yogurts a day, so over 30 days, he eats 2 * 30 = 60 yogurts. The total cost of yogurt over 30 days is 60 * $1.25 = $75.00. The final answer is: $\boxed{75}$

Table 11: A case study from GSM8K using Llama3-1B, where LLM2 corrects the vanilla model's logical error.