Do LLMs know when they are wrong?

Bartosz Piotrowski[1], Witold Drzewakowski[12]
, Konrad Staniszewski[2], and Piotr Miłoś[12]

[1] IDEAS NCBR
[2] University of Warsaw

Abstract

Large language models (LLMs) seem to be biased to rather provide an unreliable answer than to admit that they are unsure how to correctly respond to a question. This is especially problematic in the context of reasoning-intensive problems. Therefore, in this work we study "*approximate verifiers*" – auxiliary statistical models estimating the correctness of outputs generated by LLMs. Such "verifiers" has been already studied, but typically they were constructed as LLMs themselves, often as large (or larger) than the base model they support. In this work, we introduce a novel lightweight approach, LiLaVe, which reliably extracts correctness signals from the hidden states of the base LLM, and operate with only a small fraction of the computational budget of LLM-based approximate verifiers.

1 Introduction

LLMs have shown unprecedented performance in a plethora of tasks related to processing natural language and knowledge retrieval. Recently, there has been interest in enhancing *reasoning capabilities* of LLMs. This effort includes applying LLMs to solve math problems, writing code, or predicting proof steps in proof assistants.

Efforts to improve LLM performance on reasoning-intensive tasks have followed two primary directions. First, there is a body of work focusing on *pre-training* or *fine-tuning* models targeting reasoning-intensive tasks. Second, there is ongoing research into designing *inference-time techniques* to enhance the performance of LLMs on reasoning-focused tasks, where an LLM is already pre-trained and fixed. Examples of such simple yet effective techniques are chain-of-thought prompting [8], and self-consistency decoding [7], also known as *majority voting*. More advanced inference-time approaches often combine the decoding process from the base LLM with a *verifier*, trained to assess the correctness of individual reasoning steps – or entire reasoning trajectories – in order to enhance the base model's performance [2, 4]. Typically, such verifiers are LLMs themselves, often as large – or larger – than the base model they support, making them computationally expensive.

To overcome this limitation, our work aims to develop computationally efficient verifiers, which can be used to enhance the performance of the base LLMs in reasoning-intensive tasks. To this end, we develop LiLaVe – Lightweight Latent Verifier, which is a simple and practical method for extracting the correctness signal from the *hidden states* of the base LLM.

2 Method and experiments

Given a question q, an LLM generates an answer sequentially as $y = y_1 y_2 \cdots y_m$, where y_is are individual tokens. During the decoding, we extract hidden states $h_t^l \in \mathbb{R}^n$ representing the activations from the l-th transformer's layer at the generation of the t-th token, where n is the hidden dimension of the model. Rather than using all layer-token pairs (l, t), we restrict extraction to subsets of indices which we determine experimentally.

While the answer y contains the chain-of-thought style reasoning, we determine its correctness solely by looking at the final answer. To evaluate correctness, we compare the generated

Table 1: Performance (AUC) of five methods for predicting the correctness of the LLM's answers.

benchmark	LiLaVe	self-reflect	logprobs	ORM-Mistral	ORM-Deepseek
GSM8K (test)	0.86	0.68	0.78	0.81	**0.88**
GSM-Symbolic	0.84	0.70	0.78	0.85	**0.90**
GSM-Symbolic-p2	**0.78**	0.60	0.63	0.73	0.75
algebra_linear_1d	**0.93**	0.61	0.81	0.90	0.90
MATH500	0.88	0.79	0.67	0.79	**0.90**

final answer to the ground truth, which results in a binary correctness label c. Finally, a dataset $\mathcal{D}$ for training LiLaVe consists of data points in the form of quadruples (h_t^l, l, t, c).

Having collected $\mathcal{D}$, we train an XGBoost [1] classifier M to predict the binary label c given the hidden state h_t^l and its location given by the indices l, t. The output score $M(h_t^l, l, t) \in [0, 1]$ is to be interpreted as the probability of the response y to be correct.

During inference, the base language model generates a response y along with a set of associated hidden states H_y, which are indexed by their locations (l, t). We then apply the trained XGBoost model M to predict a score s_h for each hidden state $h \in H_y$. Finally, these scores are aggregated, which results in the final correctness estimate, *i.e.*, the LiLaVe score:

$$\text{LiLaVe}(y) = \text{aggregate}(\{s_h\}_{h \in H_y}) \in [0, 1].$$

We compare LiLaVe with two natural baseline methods for estimating the probability of the correctness of the language model's answer: *logprob-based estimator* and *self-reflection prompting*, as well as two LLM-based verifiers.

Logprob-based estimator: Assume that for a question q, a language model generates a response $y = y_1, y_2, \ldots, y_n$, where each decoded token y_i is given probability p_i. For each question, we compute the sum of log-probabilities over a k-suffix: $\sum_{i=0}^{k-1} \log p_{n-k}$. We treat this sum as an (uncalibrated) estimation of the output correctness. For each dataset, we choose the suffix length k, for which this estimator achieves the highest AUC score.

Self-reflection prompting: We prompt the same LLM that generated the answer to rate its confidence in the answer's correctness on a 1–10 scale.

LLM-based verifiers: We also benchmarked two LLM-based verifiers (aka outcome reward models, or ORMs), trained on over 250k synthetic examples generated from Mistral 7B and DeepSeekMath-Instruct 7B, as implemented in [9]. Both of them are based of Llama 3.1 8B, and fine-tuned to return a real-valued score.

We evaluate LiLaVe and LiLaVe-based meta-generation strategies on four mathematical natural language Q&A datasets: GSM8K [2], GSM-Symbolic [5], MATH [3], and algebra_linear_1d [6]. For each of them we select 1000 training examples to train a dataset-specific LiLaVe. We test on sets of 500–1319 examples, depending on the dataset.

In Table 1 it can be seen the LiLaVe outperforms self-reflect, logprobs on all benchmarks, outperforms ORM-Mistral on all but one (GSM-Symbolic), and outperforms ORM-Deepseek on two benchmarks (GSM-Symbolic-p2, algebra_linear_1d), even though both ORMs are a few orders of magnitude bigger. Given these results, we conclude that LiLaVe excels at extracting useful signal estimating model's correctness.

This means that our method can be used in *meta-generation* strategies, such as *weighted majority voting*, where LiLaVe scores are weights of different LLMs answers to a math question, or *conditional self-correction*, where LLM is prompted to check and reconsider its answrer whenever the LiLaVe score of the initial answer is low

References

[1] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM, 2016.

[2] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[3] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[4] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *CoRR*, abs/2305.20050, 2023.

[5] I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024.

[6] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models, 2019.

[7] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[9] W. Xiong, H. Zhang, N. Jiang, and T. Zhang. An implementation of generative prm. https://github.com/RLHFlow/RLHF-Reward-Modeling, 2024.