EEV: Enhancing Long-Term Reasoning of LLM with Effective and Efficient Verifier

Anonymous ACL submission

Abstract

Large language models have garnered significant attention due to their demonstrated strong logical reasoning capabilities. However, there is still ample room for improvement in handling complex long-term reasoning problems. Direct fine-tuning LLM on domain-specific data is an effective approach, but it requires substantial financial costs. Another research line aims to efficiently enhance performance by leveraging the model's inherent capabilities without tuning any parameters. They either utilize LLM's fact evaluation ability for self-verification during logical reasoning or employ voting methods to improve the consistency of the model's decisions. However, due to inherent limitations in specific domains, the benefits of self-verification approaches are typically limited. In this paper, we propose a compromised method, which involves training a small verification model to evaluate the reasoning process of large models. To overcome the error propagation problem of traditional verification model, we further propose a contrast-enhanced verification model training framework. The experimental results show that our proposed effective and efficient verifier (EEV) can achieve substantial performance gains on five datasets for multi-hop fact reasoning and long-term mathematical reasoning at a small cost.

1 Introduction

Large language models (LLM) have recently garnered significant attention due to their ability to address various tasks and exhibit strong reasoning capabilities. However, even state-of-the-art LLMs fall short in complex long-term reasoning tasks, such as multi-hop fact reasoning and intricate mathematical problems. To tackle the complex tasks, it is often necessary for large models to possess the capability to invoke tools, which has led to the widespread adoption of ReAct mechanism (Yao et al., 2022), inspired by the chain-of-thought (CoT) approach. This method dissects

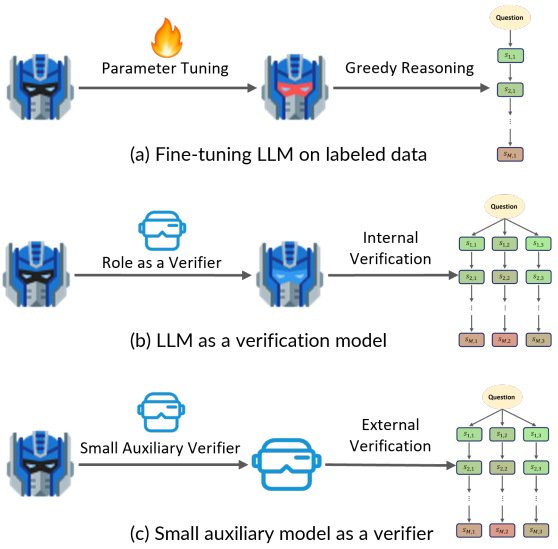

Figure 1: Three research directions to enhance the LLM's long-term reasoning ability: a) directly fine-tuning LLM on the annotated data; b) verifying the reasoning process by itself; c) Training an external verification model to guide the reasoning process.

complex problems into smaller components, iteratively obtaining the final solution through a three-step process of thinking, tool usage, and observation feedback. In contrast to the CoT (Wei et al., 2022) method, ReAct (Yao et al., 2022) involves a multi-step reasoning-feedback loop, with the reasoning context length noticeably increasing, posing a substantial challenge to the LLM's comprehension ability. Furthermore, as the reasoning process continues, error propagation becomes increasingly severe. These two critical factors restrict the performance of LLMs in long-term reasoning. To enhance the long-term reasoning capabilities of LLMs, three primary research directions are being pursued: 1) Directly fine-tuning LLMs using labeled data; 2) Improving the consistency of reasoning or optimizing the reasoning process through self-verification using the inherent capabilities of LLMs; 3) Utilizing small auxiliary models to evaluate the reasoning paths of LLMs and selecting the

optimal reasoning process.

Fine-tuning large language models, as shown in Figure 1(a), is a method that involves tuning the parameters of the model on annotated data, thereby enabling them to adapt to complex reasoning domains. However, this approach has several drawbacks. Firstly, it requires significant computational resources. Secondly, it can lead to the models forgetting knowledge from other domains, making the method unsuitable for scaling. Additionally, directly fine-tuning lacks transferability, as parameters trained on one LLM cannot be easily applied to another. Furthermore, for closed-source LLMs, direct access to the model parameters for training is often not achievable.

Increasing the long-term reasoning ability of LLM by its inherent capabilities, as shown in Figure 1(b), is also popular, which is convenient to use and just need designs some prompts. Two methods can be employed to achieve this: 1) Choosing the most voted option at each step, which aims to improve the self-consistency of LLM decisions and mitigate uncertainty errors. 2) Evaluating different reasoning trajectories internally to select the optimal one, which reinforces the model's understanding of the reasoning process through explicit verification. Reasoning and verification are two distinct perspectives of the same LLM regarding the reasoning process, operating independently. Compared with fine-tuning method, it is more efficient as they do not tuning any parameters. However, their performance improvement is normally limited due to the inherent constraints of LLMs in specific domains and their inability to discern better reasoning paths autonomously.

In this paper, we propose a method to enhance the long-term reasoning ability of LLMs by training a small auxiliary verification model to assist in evaluating each step of the reasoning process, as shown in Figure 1(c). The auxiliary verification model serves as a compromise between the two aforementioned methods. Compared to directly fine-tuning the LLM, we only need to fine-tune a smaller model with parameters comprising only 6.21% of a medium-sized 7B LLM's parameters, significantly reducing the computational cost required for parameter updates. Compared to utilizing the self-verification of the LLM, the auxiliary verifier can make full use of domain-annotated data to evaluate the quality of the reasoning process. It achieves a good balance between training efficiency and reasoning performance. However, collecting data for training the auxiliary verification model presents the biggest challenge. Inspired by the independent verification method proposed by Wang et al. (2023), we adopt the Monte Carlo Tree Search (MCTS) method to simulate the execution of multi-step reasoning processes, collecting positive and negative sample data by comparing them with real annotated results. The independent verification method is a result-supervised verifier that selects the highest-scoring reasoning trajectory among multiple randomly sampled trajectories generated by the LLM. However, the independent verification method suffers from the problem of error propagation due to the lack of timely intervention in the LLM's reasoning process. In this paper, we propose an interactive **E**ffective and **E**fficient **V**erifier (**EEV**) trained in a contrast-enhanced manner, which selects the current optimal reasoning process at each step, effectively alleviating the problem of error propagation.

In summary, our main contributions are as follows:

- We propose an effective and efficient verification model, which has excellent transferability and scalability. The EEV model is trained on the automatic process-supervised data constructed by MCTS, which can efficiently adapt to different LLMs.
- Our proposed interactive verification model trained with pairwise ranking loss can largely alleviate the error propagation problem in the traditional independent verification model.
- We evaluate our proposed EEV with adequate strong baselines on five different long-term reasoning tasks. The EEV model achieves significant performance improvements on all of these tasks.

2 Preliminaries

Task Definition A long-term reasoning trajectory of length M can be formulated as $T = (Q, s_1, s_2, ..., s_M)$ where Q is a given question, $\{s_i\}_{i=1}^{M}$ is a series of steps with the last one s_M presenting the final outcome.

ReAct is a long-term reasoning method that employs LLM-based agents as a generator to generate both reasoning traces and task-specific actions in an interleaved manner. Each step s_i of a ReAct trajectory consists of a thought t_i, an action

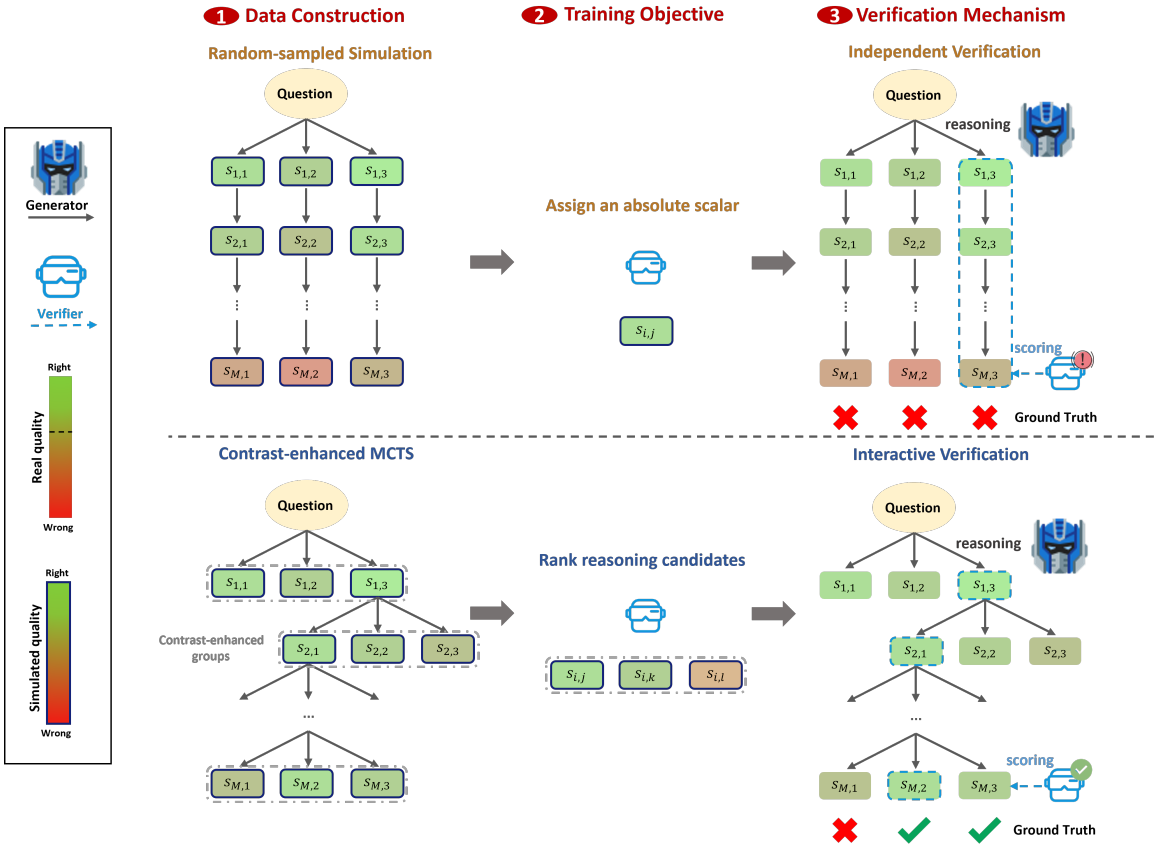

Figure 2: The three key processes of training and applying independent verifier and interactive verifier: 1) Training data construction: Traditional independent verification method constructs data through random-sampling simulation, while interactive verification method constructs data through contrast-enhanced MCTS; 2) Training objective design: Independent verification method updates the scoring model by making the scoring function approach the absolute value of the scoring of the sampled reasoning process, while interactive verification method learns by ranking different reasoning processes; 3) Reasoning verification mechanism: Independent verification method first lets the LLM randomly sample multiple full solutions and select the optimal one, while interactive verification method selects the current optimal reasoning process at each step.

a_i and an observation o_i so that $s_i = (t_i, a_i, o_i)$. At i-th step, LLM generates a natural language thought t_i and a task-specific action a_i, and receives an observation o_i from the environment, based on the its history trajectory $T_{i-1} = (Q, (t_1, a_1, o_1), (t_2, a_2, o_2), ..., (t_{i-1}, a_{i-1}, o_{i-1}))$ as input. Subsequently, (t_i, a_i, o_i) will be appended to the reasoning history, and the updated reasoning history T_i will serve as the input for LLM at the next step.

Monte Carlo Tree Search (MCTS) is a heuristic search algorithm for decision processes, aimed to evaluate each possible action by simulating random action sequences and updating the estimation based on the simulation results. The core idea of MCTS involves performing four basic steps iteratively: selection, expansion, simulation, and backpropagation. Starting from the root node, it selects child nodes based on a specific policy until reaching a leaf node. Then, it expands the tree by adding new nodes. Next, it performs random simulations on the expanded nodes to obtain evaluation values. Finally, it backpropagates the evaluation values to the root node, updating the value estimates for each action.

3 Methodology

In this section, we first propose our *interactive verification* mechanism. Subsequently, we present the corresponding data construction pipeline *contrast-enhanced MCTS* and training method based on the pairwise ranking loss. We elaborate on the shortcomings of existing methods and demonstrate how we have improved upon them.

3.1 Verification Mechanism

The external verification framework has a LLM-based generator and a verifier . Multiple can-

didate solutions (as well as reasoning trajectories) are first sampled from a generator, and then scored by a verifier. The solution with the highest score is selected as the final answer. The performance improvement achieved by verification hinges on the accuracy of the verifier, and is also upper bounded by the quality of the candidate solutions.

Independent verification (Wang et al., 2023) takes place after the entire generation is completed. The generator employs the full solution sampling that generates N independent inference trajectories $\mathcal{T} = \{T_j = (Q, s_{1,j}, s_{2,j}, ..., s_{M,j})\}_{j=1}^{N}$. The trajectory selection strategy of the independent verification is to select an optimal reasoning trajectory from all the generated candidates, which can be formulated as:

$$\textsc{IndV}(\mathcal{T}) = \underset{T_j}{\operatorname{argmax}}\, A(\{S(s_{i,j})\}_{i=1}^{M}),$$

where $S(\cdot)$ and $A(\cdot)$ represent scoring and aggregation function, respectively. The aggregation function can be average operation, which represents selecting the trajectory with the largest average score.

We propose an ***interactive verification*** mechanism, where verification process is performed immediately after each intermediate step is generated. At the i-th step, the generator generates N candidate steps $\{s_{i,j}\}_{j=1}^{N}$ based on the reasoning trajectory up to now $T_{i-1} = (Q, s_1, s_2, ..., s_{i-1})$, and then the verifier evaluates and selects the best step $s_{i,*}$ among them. The selected step $s_{i,*}$ will be appended to the reasoning trajectory and used for the next generation. The production of the optimal reasoning trajectory can be formulated as:

$$\textsc{IntV}(\mathcal{T}) = (Q, s_{1,*}, s_{2,*}, ..., s_{M,*}),$$
$$s_{i,*} = \underset{s_{i,j}}{\operatorname{argmax}}\, S(s_{i,j})$$

The advantages of our proposed interactive verification mechanism are as follows:

Higher performance upper bound Even if IndV is always capable of picking out the best, the quality expectation of the best solution selected, however, is relatively low due to the prevailing error propagation in full solution sampling. By integrating intermediate evaluation to generation step by step, our proposed interactive verification method effectively mitigates error propagation, resulting in an enhanced quality expectation of the best solution and thus a higher performance upper bound.

Lower inference cost The ReAct method requires waiting for external feedback before proceeding to the next step so that generator must generate each step sequentially. In contrast to the full solution sampling in independent verification, the LLM-based generator in the interactive verification mechanism generates intermediate candidate steps based on the same reasoning history, thus reducing the input tokens of generator to $1/N$. The number of tool calling is also $1/N$ since only the verifier-selected action are executed at each step.

3.2 Data Construction

To train an interactive verifier, we need process-supervised data with accurate step-wise annotations. However, most datasets lack step-wise annotations, and it is costly to resort to human annotators. Wang et al. (2023) defines the quality of each reasoning step as its probability of deducing the correct answer and employs a MCTS-simulation-based approach to annotate each step with completion success rate. Given a step s_i generated by a generator, a simulator is used to generate N follow-up completions $\{(s_{i+1,j}, ..., s_{M_j,j})\}_{j=1}^{N}$, and the percentage of simulated completions that eventually present the ground truth answer s* (i.e. the completion success rate) is annotated as the simulation score $\textsc{Sim}(s_i) = \frac{\sum_{j=1}^{N} \mathbb{1}(s_{M_j,j}=s*)}{N}$. Common pipeline for immediate-interactive data construction, which we refer to as *post annotation*, is similar to post verification where step-by-step solutions are first sampled by generators and then presented to human annotators or automatic simulators.

We adopted the simulation method from the independent verification mechanism (Wang et al., 2023). To fully leverage the contrastive information within the natural branching structure of search trees, we propose a *Contrast-enhanced MCTS* pipeline. The pipeline follows a step-by-step generation and simulation mechanism. For each question Q in the original outcome-supervised-only dataset serving as the root node, contrast-enhanced MCTS performs the following three steps iteratively:

Expension Starting from the root Q, choose an explorable node s_i. The path from Q to s_i is the reasoning trajectory up to now $T_i = (Q, s_1, s_2, ..., s_i)$, based on which a generator generates N_1 candidate steps $\{s_{i+1,j}\}_{j=1}^{N_1}$ for the next stage.

Simulation For each candidate $s_{i+1,j}$, if it is a intermediate step (i.e. internal node), a simulator generates N_2 completions $\{(s_{i+2,j}, ..., s_{L_j,j})\}_{j=1}^{N_2}$

and annotates it with the completion success rate. If $s_{i+1,j}$ already presents the final answer (i.e. leaf node), annotate it directly with 0-1 label according to the correctness of the answer. The simulated annotations are denoted as $\{\textsc{Sim}(s_{i+1,j})\}_{j=1}^{N_1}$

Selection The highest-annotating one among all annotated candidate steps $\{(s_{i+1,j}, \textsc{Sim}(s_{i+1,j}))\}_{j=1}^{N_1}$ will be chosen as the next node to explore, while the rest will be pruned and not further explored. As a result, the width of the search tree remains constant and quantities, where the constructed data contained positive samples and negative samples are balanced at each stage. If the chosen step happens to be a leaf node presenting the final answer, the iteration terminates.

In this case, a group of process-supervised data with shared reasoning history $\{(T_i, s_{i+1,j}, \textsc{Sim}(s_{i+1,j}))\}_{j=1}^{N_1}$ can be gained at each level of each search tree. Each group contains abundant contrastive information, which we refer to as *Constrast-enhanced Group*.

3.3 Training Objective

Traditional independent verification model are trained with a binary cross-entropy loss (Wang et al., 2023; Cobbe et al., 2021; Li et al., 2023) aiming to generate an absolute score between 0 and 1:

$$L_{BCE} = -(y_s \log r_s + (1 - y_s) \log(1 - r_s))$$

where s is a reasoning step, r_s is the score assigned by verification model and y_s is the ground truth label.

To effectively leverage contrast-enhanced groups, we propose a pairwise ranking loss with contrast-based weight and margin. During the training, a pair of positive and negative samples that differ in simulated scores $((s_p, y_p)), (s_n, y_n))$ (positive for a higher score) are selected from each contrast-enhanced group. The loss function works as follows:

$$d = y_p - y_n,$$
$$L_{Rank} = -W(d) * \log(\sigma(r_p - r_n - M(d))),$$

where $W(\cdot)$ and $M(\cdot)$ represent a weight function and a margin function, respectively. The loss function we designed is mainly focused on the following aspects:

Less biased within-group comparison Simulated annotations exhibit different degrees of bias. The simulated score for each reasoning step not only depends on its quality, but is also biased by the difficulty of the problem to solve and the capability of the simulator. For example, for an easy problem, the overall success rate of simulator's completions is high, resulting in both correct and incorrect steps receiving relatively high scores. In contrast, the simulated scores for both correct and incorrect steps are much lower for a challenging problem. Sometimes a correct step could receive a lower score than an incorrect step, thus leading to a inconsistency between the quality of the reasoning step and the corresponding annotation. As a consequence, using BCE loss to train the verifier to output an absolute score could confuse the verifier. However, data within a contrast-enhanced group shares the same reasoning history, thus their simulated annotation share the same degree of bias. While the absolute value of within-group annotations are biased, their relative ranking are precise. Therefore, We perform a within-group comparison as a circumvention, where the verifier learns to pick out the best of a contrast-enhanced group rather than to assign an absolute scalar.

More accurate contrastive information Following Touvron et al. (2023), a function $M(\cdot)$ is introduced to set the margin based on the annotation difference $(y_p - y_n)$. Our verifier can be guided to assign more differential scores to reasoning steps with more differences, thus providing more accurate scoring.

Mitigating simulation errors Simulation errors exist since a limited number of samplings are employed to simulate the real completion success rate. So smaller annotation differences are more likely to be noised by simulation errors. Therefore, we introduce an annotation difference-based weight function $W(\cdot)$ to assign more confidence to a more differential pair.

4 Experiments

4.1 Experimental Setup

Datasets Our experiments on two major types of long-term reasoning datasets. 1) Knowledge-intensive reasoning: two multi-hop question answering datasets HotpotQA, TriviaQA and a fact verification dataset FEVER; 2) Mathematical reasoning: two math word problem datasets GSM8K, MathQA. For datasets with private test sets, we perform testing on the validation sets.

Baselines 1) Vanilla ReAct with greedy reasoning: LLM-based generator generates a single

Models	Method	HotpotQA		TriviaQA	FEVER	GSM8K	MathQA
		EM	F1	EM	EM	EM	EM
LLaMA2-7B-Chat	ReAct	22.2	30.2	40.6	40.8	5.0	5.2
	Self-Verification	21.0	28.1	39.8	41.0	4.0	4.8
	Self-Consistency	24.8	33.4	42.8	41.8	10.6	8.2
	IndV	24.8	33.5	46.6	51.8	13.2	6.2
	EEV	26.4 ▲1.6	35.7 ▲2.2	50.8 ▲4.2	**58.4** ▲6.6	16.2 ▲3.0	7.2 ▼1.0
	EEV (+SC)	**29.0** ▲4.2	**39.7** ▲6.2	**52.4** ▲5.8	52.2 ▲0.4	**18.8** ▲5.6	**8.8** ▲0.6
Mistral-7B-Instruct-v0.2	ReAct	25.4	34.7	42.6	52.6	36.0	14.0
	Self-Verification	24.4	33.0	43.2	52.2	33.8	13.0
	Self-Consistency	30.4	40.6	53.8	54.2	42.8	19.4
	IndV	27.6	39.7	55.2	60.0	34.6	17.8
	EEV	27.4 ▼3.0	38.0 ▼2.6	**59.4** ▲5.6	**64.6** ▲4.6	43.6 ▲0.8	18.2 ▼1.2
	EEV (+SC)	**34.2** ▲3.8	**44.3** ▲3.7	58.6 ▲4.8	60.2 ▲0.2	**52.0** ▲9.8	**21.8** ▲2.4

Table 1: Evaluation long-term reasoning ability of LLMs on five tasks. There are two LLMs as the generator: LLaMA2-7B-Chat and Mistral-7B-Instruct-v0.2. EEV and EEV (+SC) are our proposed method. Self-Verification and Self-Consistency leverage the inherent ability to enhance the long-term reasoning performance, which do not need tuning any parameters. IndV is independent verification model, which is trained with BCE loss.

reasoning trajectory by greedy decoding; 2) Self-Verification: LLM as a verifier selects its preferred reasoning steps; 3) Self-Consistency (SC): majority voting based on multiple reasoning trajectories that LLM-based generator randomly sampled; 4) IndV: a process-supervised verifier with the same architecture as EEV, trained by binary cross-entropy loss on data constructed by random-sampled simulation, performs independent verification. Noting that due to computational resource limitations, we did not implement a direct fine-tuning experiment of the large language model. The proposed method is a LLM-parameter-agnostic approach, so we compared it directly with all other methods of the same type that do not require LLM's parameter fine-tuning.

Implementations All generator perform reasoning with tool calling in a ReAct manner. For retrieval augmented knowledge-intensive reasoning, generators are equipped with a Wikipedia API following Yao et al. (2022), by which they are able to retrieve related information to help answer questions. For program aided mathematical reasoning, python compiler-based calculator and equation solver are introduced to help numerical and algebraic operations. For knowledge-extensive reasoning, we employ Llama2-7b-chat as the generator and simulator for data construction. For mathematical reasoning, we employ Mistral-7b-instruct as the generator and simulator for data construction. Random-sampled simulation in the IndV baseline are using the same LLMs. Our EEV model and IndV verifier uses Longformer-large (Beltagy et al., 2020) as its backbone. We test our proposed verifier with Llama2-7-chat, Mistral-7b-instruct and GPT-3.5-turbo API.

4.2 Main Results

In this subsection, we compare our proposed EEV method based on interactive verification with four other strong baselines that do not require fine-tuning the LLM parameters on five tasks. The five different types of tasks include two knowledge-intensive multi-hop fact reasoning tasks and three complex mathematical reasoning tasks. Except for the HotpotQA task, which used F1 as the metric, all the other tasks only used EM (exact match accuracy) as the metric, where only the final inference result of the LLM matches the annotated result to be considered correct.

To verify the stability of our proposed method, we select two 7B-scale instruction-tuned LLMs as generators in the experiment: LLaMA2-7B and Mistral-7B-V2. The EEV method selects the best inference process at each step during the inference process of LLM-based generators, combined with the influence of self-consistency. It first selects the actions of calling tools according to different inference processes for voting, and then scores the inference processes corresponding to different voting actions, thus proposing the EEV (+SC) method.

Compared with the vanilla ReAct method, the self-verification method uses the large language model itself as the scorer to select the inference process it prefers from multiple inference processes. The experimental results in Figure 1 show that the

Model	Method	HotpotQA		GSM8K
		EM	F1	EM
GPT-3.5-turbo	ReAct	21.8	28.0	63.4
	Self-Consistency	22.6	29.0	67.4
	EEV	**31.0**	**39.8**	68.8
	EEV (+SC)	28.0	36.7	**74.4**

Table 2: Evaluate that closed-source model (GPT3.5-turbo) as generator and EEV as verifier on one multi-hop fact reasoning task (HotpotQA) and one mathematical reasoning task (GSM8K).

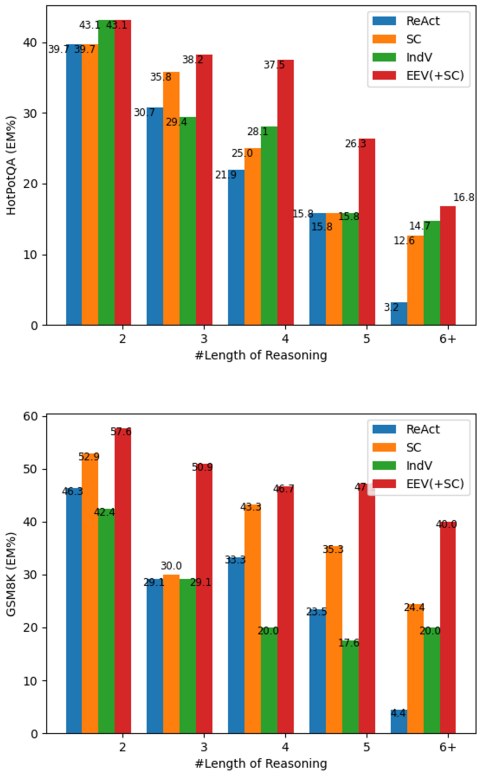

Figure 3: Three research directions to enhance the LLM's long-term reasoning ability: a) directly fine-tuning LLM on the annotated data; b) verifying the reasoning process by itself; c) Training an external verification model to guide the reasoning process.

self-verification method cannot improve the long-term reasoning performance of the large language model itself.

The self-consistency method is another method that utilizes the capabilities of LLM to improve the reasoning effect. It selects the inference process with the most votes among multiple sampling reasoning processes according to the selected inference action. The experimental results in Figure 1 show that compared with the vanilla ReAct method, the self-consistency method can consistently improve the performance of the long-term reasoning process. The main reason for this is that it references different inference processes and maintains the stability of the LLM's inference process through voting.

Independent verification is the method closest to our proposed interactive verification. Independent verification trains the scoring model through directly learning the score values, while our proposed interactive verification trains through sorting different inference processes, and the training data is collected through our improved contrast-enhanced MCTS process. The experimental results in Figure 1 show that our model can perform better on these long-term reasoning tasks. The scoring model trained through contrastive learning enables our proposed method to effectively distinguish between inference processes with subtle differences.

Compared to the EEV method, the EEV (+SC) approach demonstrated substantial performance gains in four out of five tasks utilizing LLaMA as the generator, with a notable decrease in performance observed on the FEVER dataset. When Mistral was employed as the generator, three tasks exhibited significant improvements, while slight decreases were observed on TriviaQA and FEVER. Our findings indicate that the EEV (+SC) method consistently enhanced performance for complex mathematical reasoning tasks. Notably, compared to knowledge-intensive reasoning tasks, mathematical reasoning demands a higher level of detail in the reasoning process, suggesting that EEV (+SC) performs better in scenarios with stringent requirements for reasoning details.

4.3 Analysis

In this subsection, we first analyze the performance of our proposed EEV model as verifier when used with a closed-source model as a long-term reasoning generator. Then, we examine the performance of different methods at various reasoning lengths.

4.3.1 Closed-source Model as Generator

We employ the closed-source model GPT3.5-turbo as the generator for the inference process. To minimize computational expenses, we conduct tests on a knowledge-intensive fact reasoning task and a mathematical reasoning task, respectively. Additionally, we did not evaluate the independent verification method that closely aligns with our proposed approach, as it requires substantial computational resources due to the absence of shared inference history. We compare the vanilla ReAct

method and the Self-Consistency method with our proposed methods. The experimental results in Figure 2 demonstrate that our proposed EEV models seamlessly integrates into the long-path inference process of the closed-source model, yielding significant performance improvements in both tasks.

4.3.2 Effect of Reasoning Length

In this subsection, we compare the variations in inference accuracy of different methods across different lengths of reasoning. We conducte statistical analysis on the HotpotQA and the GSM8K to examine the results. The findings as shown in Figure 3 demonstrate a clear performance advantage of our proposed EEV model as the length of reasoning increases, particularly in mathematical reasoning tasks. This indicates that our method effectively addresses the performance degradation issue caused by the error propagation, providing a promising solution.

5 Related Works

Long-Term Reasoning Long-term reasoning entails multi-step reasoning until reaching a final outcome. The notorious issues of hallucination and error propagation make long-term reasoning hard even for the state-of-the-art LLMs. Knowledge-intensive reasoning such as HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), FEVER (Thorne et al., 2018), CommmonsenseQA (Talmor et al., 2018), StrategyQA (Geva et al., 2021), and mathematical reasoning such as GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), MathQA (Amini et al., 2019), SVAMP (Patel et al., 2021), AsDiv (Miao et al., 2021), are two mainstream long-term reasoning tasks. Among various reasoning methods, Chain-of-Thought (CoT) (Wei et al., 2022) has demonstrated LLMs' capability to solve problems in a step-by-step manner, and much of the follow-up methods can be seen as its modifications and variants. Zhou et al. (2022), Kojima et al. (2022) and Diao et al. (2023) explore various design and selection methods of CoT prompts. Yao et al. (2023) and Besta et al. (2023) refine the structure of CoT to adapt to specific scenarios. In view of the hallucination during reasoning, it is practical to introduce external tools (Schick et al., 2023), eg. a simulation engine (Liu et al., 2022), a database (Yu et al., 2022), Wikipedia (Yao et al., 2022), a code interpreter (Gao et al., 2023; Chen et al., 2022), a calculator and a answer engine (Xu et al., 2023). Among the array of approaches for integrating LLM reasoning and tool invocation (Yao et al., 2022; Xu et al., 2023; Ruan et al., 2023; Kong et al., 2023), ReAct stands out by interleaving reasoning and tool calling, and incorporating external feedback into the reasoning process. This characteristic renders it a simple and effective multi-step reasoning technique, notable for its interpretability and applicability across various domains.

Verification The Verification mechanism aims to enhance model performances by selecting the optimal response among multiple generated candidates. This mechanism can be categorized based on the verifier: 1) *Internal Verification*: The model verifies its own generated responses. Wang et al. (2022) employs a majority voting mechanism. Zhao et al. (2023) allows models to choose from CoT and Program-Aided Language Models (PAL) results. Weng et al. (2023) integrates forward reasoning and backward verification to verify each reasoning step. 2) *External Verification*: An independent verifier is trained to evaluate model responses (Cobbe et al., 2021; Shen et al., 2021; Nichols et al., 2020; Wang et al., 2023; Liu et al., 2023; Li et al., 2023). Furthermore, Verification can also be divided into two groups based on its granularity: 1) *Outcome supervision*: Only verifying the final result. 2) *Process supervision*: Verifying each step of the process. While Uesato et al. (2022) illustrates similar performance between ORM and PRM on GSM8K, Lightman et al. (2023) advocates for the superior performance of PRM on MATH.

6 Conclusion

In this paper, we propose an efficient and effective verification (EEV) model to enhance the long-term reasoning ability of large language models. This method can significantly improve the long-term reasoning performance of large language models without any parameter tuning. To address the serious error propagation problem in traditional independent verification methods, we propose an interactive verification mechanism. In the data collection stage, we obtain a large amount of contrastive data based on the MCTS method, and then train the verification model using pairwise ranking method, which shows significant advantages compared to directly learning the score scalars in independent verification methods. Finally, we achieve consistent performance improvements on five long-term reasoning tasks, including multi-hop fact reasoning tasks and complex data reasoning tasks.

Limitations

We proposed an efficient and effective verifier, which largely enhances the LLM performance on the long-term reasoning tasks. Due to the limitation of budget and computation resource, the experiments lack the performance on larger open-source large language models and more powerful closed-source models.

Ethical Considerations

As our EEV methods are validated on the existing datasets, we follow the original copyright statements of all the datasets. All claims in this paper are based on the experimental results. No demographic or identity characteristics information is used in this paper.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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*.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

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*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, et al. 2023. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333.

Hunter Lightman, Vineet Kosaraju, Yura 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*.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023. Tinygsm: achieving> 80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. 2022. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*.

Eric Nichols, Leo Gao, and Randy Gomez. 2020. Collaborative storytelling with large-scale neural language models. In *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–10.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

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. *arXiv preprint arXiv:2109.03034*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. 2023. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *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*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv e-prints*, page arXiv:2210.03629.

Wenhao Yu, Chenguang Zhu, Zhihan Zhang, Shuohang Wang, Zhuosheng Zhang, Yuwei Fang, and Meng Jiang. 2022. Retrieval augmentation for commonsense reasoning: A unified approach. *arXiv preprint arXiv:2210.12887*.

Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Qizhe Xie. 2023. Automatic model selection with large language models for reasoning. *arXiv preprint arXiv:2305.14333*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.