Trustworthiness and Self-awareness in Large Language Models: An Exploration through the Think-Solve-Verify Framework

Zhendong Liu, Changhong Xia, Wei He, Chongjun Wang
Department of Computer Science and Technology, Nanjing University
Xianlin Road No.163, Qixia District, Nanjing, Jiangsu Province, China
{dz20330019,chxia,522022330018}@smail.nju.edu.cn, chjwang@nju.edu.cn

Abstract

As Large Language Models (LLMs) become increasingly influential in reasoning tasks, ensuring their trustworthiness and introspective self-awareness is critical. This research introduces the Think-Solve-Verify (TSV) framework, an innovative strategy tailored to explore LLMs' trustworthiness, introspective self-awareness, and collaborative reasoning. This method accentuates a model's capability to construct introspective reasoning processes from answers and ensure their trustworthiness. The reasoning with TSV consistently performs at or near the top across the majority of datasets with a single interaction with LLM. Moreover, we refine the voting process of self-consistency within the Chain-of-Thought (CoT) approach, leading to notable accuracy enhancements. In our evaluations, this approach improved performance from 67.3% to 72.8% on the AQuA dataset. Furthermore, we delve into the model's ability to explain the given answers, highlighting the significance of discerning genuine comprehension from mere guesswork.

Keywords: chain of thought, trustworthiness, self-consistency, large language models

1. Introduction

In the dynamic world of Natural Language Processing (NLP), the evolution of large language models (LLMs) has been both swift and transformative. Since 2022, spurred by the introduction and extensive application of prompt engineering, these LLMs (OpenAI, 2023; Zeng et al., 2022; Du et al., 2022; Touvron et al., 2023; Anil et al., 2023) have exhibited prodigious advancements. It's undeniable that their prowess in tasks like writing, translation, and information extraction has redefined the bounds of what machines can achieve. However, for all their merits, these models reveal certain shortcomings when subjected to tasks that demand intricate reasoning. To illustrate, flagship models, such as the acclaimed GPT-4 (OpenAI, 2023), clock in at a modest accuracy rate of around 42% when tested on the MATH dataset (Hendrycks et al., 2021). To address this problem, method like Chain of Thoughts (CoT) (Shi et al., 2022; Wei et al., 2022; Zhang et al., 2022) represent promising endeavors to navigate around these challenges. By embracing a sequential reasoning approach, these models can deliver answers with a higher degree of accuracy compared to their direct-answer counterparts.

Yet, the quest for perfecting reasoning in LLMs is multi-dimensional. While a plethora of techniques such as Complex CoT (Fu et al., 2022), Tab CoT (Ziqi and Lu, 2023), Auto CoT (Zhang et al., 2022), and PHP (Zheng et al., 2023) strive to enhance reasoning precision, there's a conspicuous absence of focus on two paramount aspects: trustworthiness and model self-awareness. Should humans trust the content generated by LLMs, and do LLMs have self-awareness comparable to humans?

Several solutions have been put forth by researchers, encompassing methods like stepwise verification (Ling et al., 2023), aligning outputs to human standards using RLHF (Ouyang et al., 2022; OpenAI, 2023), and enhancing model fidelity by integrating with external systems (Lyu et al., 2023). Some scholars have explored the fine-tuning of LLMs to verify both final solutions and intermediary steps (Creswell and Shanahan, 2022; Paul et al., 2023; Cobbe et al., 2021). Nonetheless, a significant portion of these studies prioritize performance indicators, with a predominant focus on predictive accuracy as the primary benchmark for LLM evaluation. This approach has resulted in an evident gap concerning trustworthy AI, especially when it comes to the application of prompt engineering within LLMs. Furthermore, to the best of our knowledge, a comprehensive investigation into the self-awareness of these models is conspicuously absent. The challenge of reliably deriving correct solutions to mathematical reasoning questions amidst model output randomness remains unsolved. Moreover, insights into the collaborative potential of various models remain in their infancy.

Our paper makes three primary contributions:

- Introducing the TSV framework, as illustrated to simulate human-like reasoning. By segmenting the reasoning process into thinking, solving, and verifying stages, we allow LLMs to improve their reasoning capabilities in a structured manner.

LREC-COLING 2024, pages 16855–16866
20-25 May, 2024.

- Recognizing the paramount importance of trustworthiness and introspective self-awareness in LLMs, we delve deep into assessing their self-cognition capabilities. Through our research, we identify inherent limitations and introduce mechanisms to bolster their trustworthiness, ensuring they are more dependable in reasoning tasks.

- Leveraging the TSV framework, we embark on a comprehensive assessment of LLMs' ability to generate coherent explanations **when they have known answers**. Our emphasis is not just on the end result but also on the entire reasoning trajectory, ensuring that the model's answer stems from genuine comprehension and not guessing or overfitting.

2. Related Work

2.1. Chain of Thoughts

To enhance a model's reasoning performance, various strategies have been explored, such as Complex CoT (Fu et al., 2022), Tab CoT (Ziqi and Lu, 2023), Auto CoT (Zhang et al., 2022), Faithful CoT (Lyu et al., 2023), Least-to-Most prompting (Zhou et al., 2022). Through the design and use of prompts, these methods aim to enhance the capabilities of a single model, especially on mathematical reasoning tasks. Our work goes beyond mere predictive accuracy and integrates trustworthiness, problem-solving process elucidation, and model collaboration under the TSV umbrella.

2.2. Large Language Models' Trustworthiness

Although LLMs perform excellently in various tasks, they still produce illusory and untrustworthy outputs. Some researchers have investigated feedback and verification for LLMs (Chen et al., 2023; Madaan et al., 2023; Weng et al., 2022; Shinn et al., 2023). Some other researchers have proposed that models should be aware of their unknowns and uncertainties (Yin et al., 2023). The ReAct model (Yao et al., 2022), for instance, addresses issues via feedback loops. However, there's a discernible void in exploring model cooperation and delving deeper into their self-awareness. Our work discusses the trustworthiness of LLMs in more depth and quantitatively evaluates them based on a new TSV framework and some new metrics.

2.3. Consistency in language models

Previous studies have highlighted the potential inconsistency issues in language models (Adiwardana et al., 2020; Camburu et al., 2018). However, Nye et al. (Nye et al., 2021) address this by enhancing the logical consistency of samples from a System 1 model, integrating a logical reasoning module inspired by System 2. Furthermore, Wang et al. (Wang et al., 2022) introduce a novel decoding strategy termed "self-consistency", aiming to supplant the conventional greedy decoding in chain-of-thought prompting. This innovation notably augments the reasoning capabilities of language models within the Chain-of-Thought (CoT) framework. Building upon this, we further refine the self-consistency voting mechanism to elevate reasoning performance.

3. Method

3.1. Problem Formulation

A reasoning-based problem-solving task can be formally defined as a tuple (Q, C, T, S, V, A). In this tuple:

- Q denotes the target question,
- C provides the necessary context or background for addressing Q,
- T, S, and V represent the thinking, solving, and verifying stages of our proposed framework, respectively,
- A denotes the answer.

Drawing inspiration from (Ling et al., 2023), each stage can be conceptualized as a sequence of tokens or steps, represented as $S = (s_1, s_2, \ldots, s_m)$. An LLM undertakes this sequence to navigate through the problem-solving process. As shown in Fig. 1, we use red initial highlighting to align the formalization of the problem.

3.2. Motivation

The initial motivation for this paper came from observing phenomena like:

Case 1

Question: [Reasoning Problem]
LLM Response: [Response a]
User Feedback: [incorrect, please check.]
LLM: [Response b]
However, we found that often the LLM's output in checking is still incorrect, lacking the ability to recognize errors, which is the "self-awareness" deficit mentioned in the paper.

Case 2

Question: [Reasoning Problem]
LLM Response: [Incorrect Response]
However, if we proceed in two steps:
Question: [Reasoning Problem] + [Prompt to

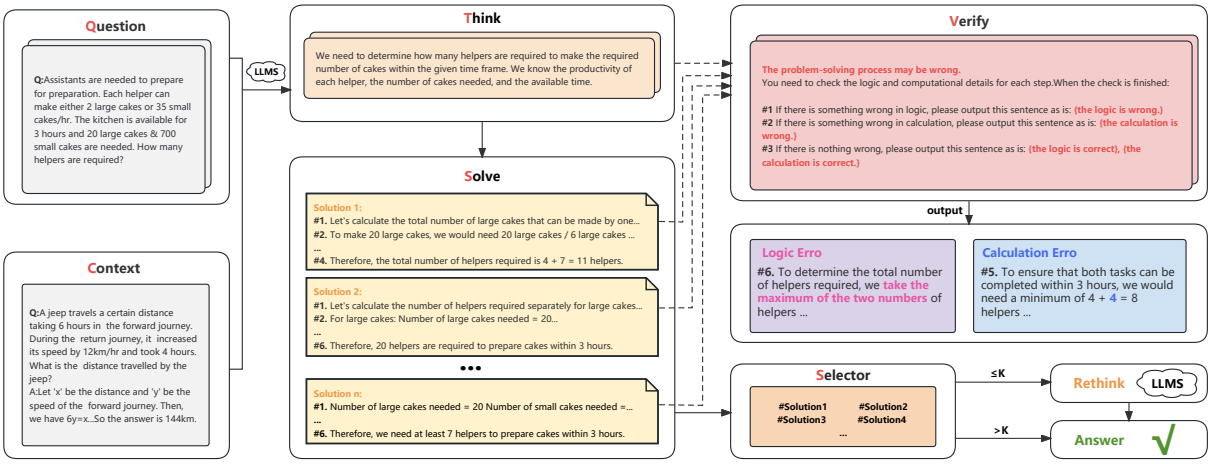

Figure 1: The comprehensive pipeline of our proposed TSV (**Think-Solve-Verify**) method. Based on the imitation of humans solving mathematical reasoning problems, we first use the thinking engine to **think** about the problem and generate simple problem-solving ideas instead of answers. Based on a generative thought process, we use multiple LLMs as solvers to **solve** the problem. After getting the answer, we **verify** the answer and use the selection algorithm we designed to improve the trustworthiness of the answer.

think, not solve]
LLM: [Thought Process]
Question: Please solve the problem based on this thought process.
LLM: [Correct Response]

Based on such cases, we designed the TSV framework. The existing work based on the Chain of Thought (CoT) primarily aims to generate reasoning processes and answers for a given problem using a singular model. However, within our defined framework, this is merely a process of producing T, S, and A given Q and C. It becomes evident that most current methods lack the verification stage, represented by V in our framework. This omission of verification underscores the inherent lack of self-awareness in these methods, hampering the trustworthiness of model outputs.

Beyond this, our framework reveals that generating T, S, and A from Q and C is just one of several tasks. Given practical considerations for the application of large language models, the evaluation of the following abilities also emerges as crucial:

1. **Model Self-awareness and Trustworthiness**: As the model provides an answer, it should also assess the trustworthiness of that response, delineating its capability boundaries. In essence, just like most humans, the model should be cognizant of the problems it can and cannot solve.

2. **Collaboration among Models**: The efficacy of hybrid expert models in enhancing performance has been well-documented (Si et al., 2023). With different models having varying parameters and computational costs, exploring how they can collaboratively address problems is of prime importance. For instance, a sophisticated model can be employed for the thinking process, while simpler models execute the solving process.

3. **Generating Solution Processes for Known Answers**: In many scenarios, especially in education or proving mathematical conjectures, the answer is already known. Here, the focus shifts to elucidating the reasoning and computation processes that lead to the known answer. This essentially becomes an exercise in explanation—where the emphasis is not on the model's answer but on its process of arriving at that answer.

Although self-consistency methods based on voting can enhance model performance and give confidence in answers, our observations align with the proverb that 'truth often resides with the minority'. This occurrence is more prevalent in high-uncertainty problems. For example, in a high-confidence scenario, the model might produce $A : 90\%$ and $C : 10\%$, clearly indicating A as the answer. However, in uncertain situations with results like $A : 40\%$, $B : 30\%$, $C : 10\%$, and $D : 20\%$, the correct answer becomes ambiguous. Our analyses indicate that the most favored answers in such scenarios aren't invariably correct. Thus, human selection, albeit introducing new complexities, can help navigate such situations. Given the advanced long-context processing capabilities of ex-

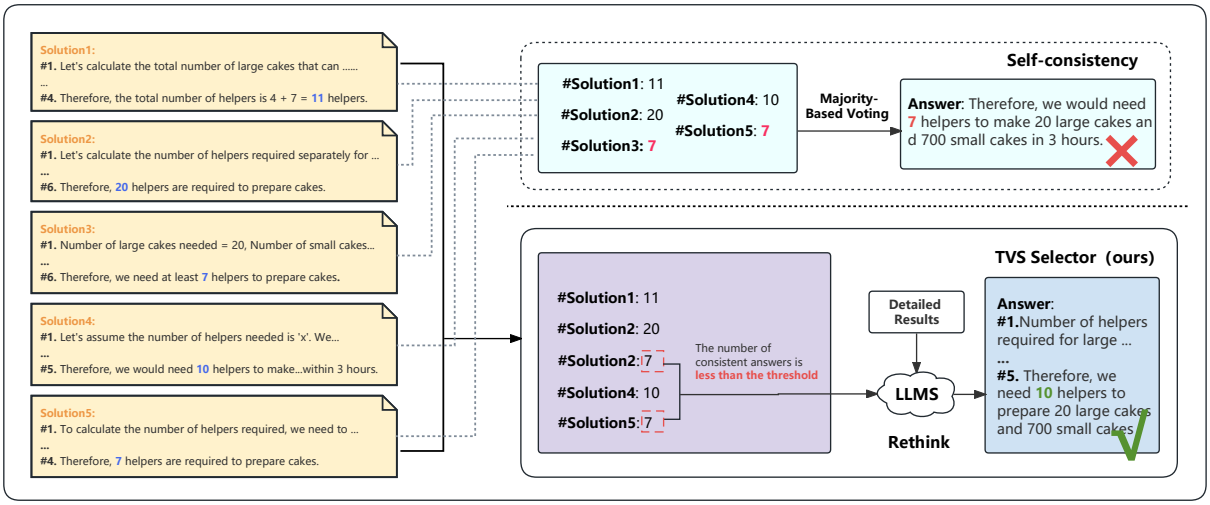

Figure 2: Illustration of our Enhanced Answer-Trustworthiness Calibration. Diverging from self-consistency methods, our approach employs multiple solvers and a selection mechanism. The algorithm ensures the selection of the answer by considering both the frequency of answers and the integrity of solution processes.

isting LLMs, we propose leveraging them to supplant this human decision-making step, as illustrated in Fig. 2.

3.3. Methodology

3.3.1. Model Self-awareness and Trustworthiness

Given the need for model trustworthiness and the challenges in fine-tuning large language models, we employ a prompt-based method to verify outputs. As depicted in Fig. 1, we use both direct verification and step-by-step verification. During this verification, both thinking and solving results are individually examined, resulting in two boolean variables V_T and V_S. If both are true, the model's verification output V_o is true; otherwise, it's false. If the model's output matches the answer A, A_o is set to true, otherwise, it's false.

Drawing parallels with confusion matrices in machine learning, we define a verification matrix using V_o and A_o:

	True($\boldsymbol{V_o}$)	False($\boldsymbol{V_o}$)
True($\boldsymbol{A_o}$)	TT	TF
False($\boldsymbol{A_o}$)	FT	FF

Where the first letter represents A_o and the second V_o. For instance, TT means both A_o and V_o are true. Similar to recall and precision metrics, we define verification metrics:

1. $Accuracy$:

$$\text{Accuracy} = \frac{TT + TF}{TT + TF + FT + FF}$$

This represents the standard model accuracy, indicating the proportion of outputs where A_o is true among all outputs.

2. $Accuracy_v$:

$$Accuracy_v = \frac{TT + FF}{TT + TF + FT + FF}$$

This metric showcases the model's self-awareness performance, measuring the performance of the verification output V_o aligns with the predicted answer output A_o.

3. SA_w **(Self Awareness for Wrong Answers):**

$$SA_w = \frac{FF}{FT + FF}$$

This quantifies the model's capability to recognize its incorrect answers, assessing whether the model can identify when its own output is erroneous.

4. SA_c **(Self Awareness for True Answers):**

$$SA_c = \frac{TF}{TF + TT}$$

This gauges the model's proficiency in recognizing its correct answers, determining if the model mistakenly classifies right answers as wrong.

However, it's essential to note that large language models may have correct results but erroneous processes. Because when this happens, the verifier may indeed have found a process error, rather than the verifier itself being wrong. Only when this

situation doesn't occur are the $Accuracy_v$ and SA_c metrics unbiased; otherwise, they should be used judiciously. In this paper, on the premise that it has no influence on the conclusion, we assume that when the answer is correct, the process of solving the problem is also correct, ignoring the impact of the incorrect process of solving the problem.

To obtain answer trustworthiness and enhance the trustworthiness of large language models, we propose a calibration method based on the self-consistency approach. As shown in Fig. 2, we use N solvers to get the output. Unlike the traditional self-consistency method, we don't solely rely on voting to determine the answer; instead, we've refined this process. The algorithm is as follows:

Algorithm 1 Algorithm for Enhanced Answer Trustworthiness Calibration

Require: Answers obtained from N different solvers, $A_1, A_2, \ldots, A_N$
Require: Solution process from N solvers, $S_1, S_2, \ldots, S_N$
Require: A threshold value, `threshold`
`counter` = Count occurrences of each answer in $A_1, A_2, \ldots, A_N$
Sort `counter` in descending order based on answer frequencies.
Calculate the total count of answers, `answer_count`.
if `counter[0] / answer_count` $\geq$ `threshold` **then**
  Set A_s to the most frequent answer.
else
  Initialize the `Selector` as an LLM with a long context and $A_1, A_2, \ldots, A_N$, $S_1, S_2, \ldots, S_N$.
  Determine A_s using the `Selector` based on all answers and solution processes.
end if
return The selected answer, A_s

3.3.2. Collaboration of LLMs

As illustrated in Fig. 1, we decouple the thinking, solving, and verification processes, meaning different models can undertake each of these tasks. This separation leverages the unique strengths of each model. For instance, while the GPT-4 model is sophisticated and computationally expensive, it excels in reasoning. In contrast, the more cost-effective GPT-3.5 model has its limitations. We can employ the sophisticated model for the thinking phase, while simpler models handle the solving phase. This separation ensures a balance between computational cost and performance, and, based on the TSV framework, different models are tested for their collaborative efficacy.

3.3.3. Generating Solution Processes

While most current reasoning tasks have unknown answers, there are scenarios, especially in education or while proving mathematical conjectures, where the answer is already known. In such situations, based on the TSV framework and prompt engineering, we explore the capability of large language models to generate reasoning and computation processes for known answers. This approach emphasizes not the correctness of the model's answer, but the process it employs to arrive at a known solution.

4. Experiments

4.1. Experimental Settings

4.1.1. Datasets

We use multiple popular datasets to evaluate our model, including AddSub Hosseini et al. (2014), MultiArith Roy and Roth (2016), ASDiv Miao et al. (2020), SVAMP Patel et al. (2021), GSM8K Cobbe et al. (2021), AQuA Ling et al. (2017) and MATH Hendrycks et al. (2021).

Given OpenAI's API constraints for GPT-4, exhaustive tests on large datasets like GSM8K and MATH are inefficient. Without affecting the experimental results, we randomly sampled 100 questions, testing them in 5 runs and averaging the results. For smaller sets like AQuA, all data was used.

Additionally, the motivation for choosing the AQuA dataset for further evaluation includes: (1) As seen in Table 1, the AQuA dataset shows the poorest performance, making it easier to demonstrate model differences. Although the MATH dataset performs worse, most models have very low performance on it, offering limited reference value. (2) The AQuA dataset is not very large. Using a dataset with thousands or even tens of thousands of complex mathematical reasoning problems would make human assessment costs unacceptably high.

4.1.2. Models and Prompt

We use the OpenAI **GPT-3.5** and **GPT-4** model for experiments (OpenAI, 2023; Brown et al., 2020; Ouyang et al., 2022). For the stable reproduction of the experimental results, the models we use are the GPT-3.5-0613 and GPT-4-0613 models. All models are employed via the OpenAI API key. For more detailed prompts and model parameters, please refer to the Supplementary Material.

	AddSub	MultiArith	ASDiv	SVAMP	GSM8K	AQuA	Average
Standard	84.2	90.1	78.2	79.2	32.7	31.9	66.0
CoT (Wei et al., 2022)	89.1	**98.3**	81.2	82.2	75.2	59.8	81.0
Complex CoT (Fu et al., 2022)	85.5	97.5	81.2	81.0	82.8	57.4	80.9
PHP (Zheng et al., 2023)	85.3	98.0	82.6	83.1	**85.1**	60.6	82.5
Faithful CoT (Lyu et al., 2023)	88.4	95.3	81.7	83.0	75.8	53.5	79.6
TSV (Ours)	**92.0**	97.0	**83.2**	**84.0**	79.2	**63.8**	**83.2**

Table 1: Accuracy comparison of various models on mathematical reasoning datasets. Each column represents a specific dataset or task, with the 'Average' column showing the mean performance across all datasets.

	InterAlgebra	Precalculus	Geometry	NumTheory	Probability	PreAlgebra	Algebra	Average
Standard	10.9	13.9	10.9	19.8	24.8	38.6	27.7	20.9
CoT	13.9	13.9	18.8	25.7	31.7	56.4	35.6	28.0
Complex CoT	14.6	16.8	22.3	33.4	29.7	53.8	49.1	31.4
PHP	17.1	16.1	25.4	**35.1**	**33.7**	57.7	51.1	33.7
Faithful CoT	/	/	/	/	/	/	/	31.8
TSV (Ours)	**22.8**	**23.8**	**34.7**	31.7	32.1	**61.4**	**58.4**	**37.8**

Table 2: Extended accuracy comparison of various models across a diverse set of mathematical reasoning datasets. The table highlights the models' capabilities in specific mathematical topics.

4.2. Reasoning Task: Given Q and C

For reasoning tasks provided with Question (Q) and Context (C), The formalization of most existing work based on the CoT approach is: Given Q and C, deduce T, S, and A. In the reasoning tasks, the zero-shot approaches typically lack C, while popular few-shot methods include context C. In most existing works, the thinking and calculation processes aren't separated, and they don't involve a verification step. When Q and C are provided, as shown in Fig. 1, our framework can infer problems similarly to other works, that is, given Q and C, however, it resolves T, S, and A. We will use the TSV framework to organize our experiments results below.

4.2.1. Thinking and Solving

Based on our framework, we found that using a single forward pass (model inference) for predictions with few-shot methods surpasses most existing CoT approaches. The results are shown in Table 1 and 2.

The TSV method consistently performs at or near the top across the majority of datasets. This indicates the robustness and effectiveness of the approach. The PHP method, despite requiring multiple rounds of interaction with the model, offers competitive performance, especially when observing datasets in Table 1. However, the added complexity and the requirement for multiple interactions can be seen as a limitation, especially when rapid results are needed. The TSV method, which requires just a single interaction, achieves similar or even better performance, highlighting its efficiency. The baseline method exhibits varied performance across datasets. While they show moderate success in some datasets, their performance significantly drops in others, emphasizing the need for more advanced methods.

In conclusion, while multiple methods show promise in addressing the challenges, TSV stands out due to its efficiency (single interaction with LLM) and consistently high performance. The results underscore the potential of TSV as a potent tool for reasoning tasks, particularly when efficiency and accuracy are paramount.

4.2.2. Verification and Model Fusion

As shown in Fig. 3, we observed that individual models without our TSV, when examining their outputs, show deficiencies. Even when they are wrong, the models still believe their thinking and calculations are correct, revealing a lack of self-awareness. The GPT-3.5 model lacks the capability to check its outputs in most cases when generating wrong answers and being asked to check with a simple prompt. This suggests a flaw in the model's self-awareness. As shown in Fig. 3, we observed that without using TSV, the model often shows defects when checking its outputs. On most datasets, the self-cognition indices of baseline methods and CoT methods are lower than when using TSV. Surprisingly, the baseline method on the MATH dataset can to some extent detect errors. Even if the model outputs are wrong, the models still believe that their thinking and calculation are

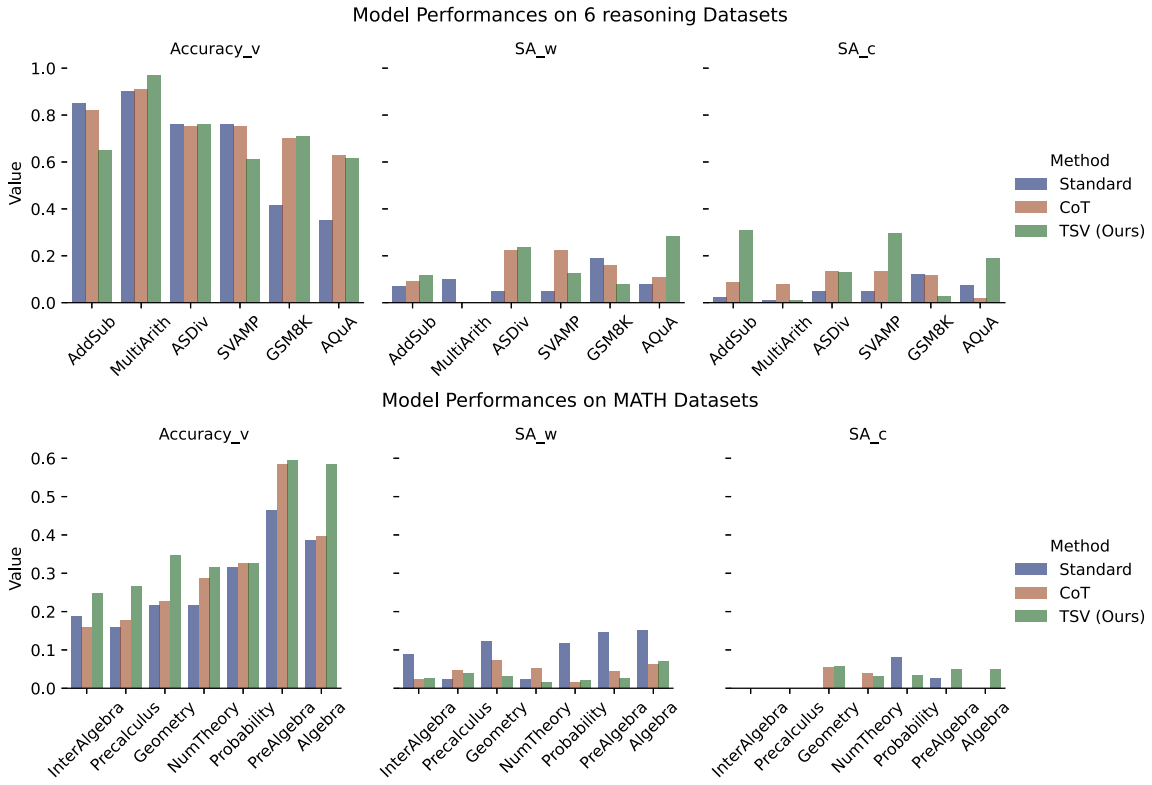

Figure 3: Comparative analysis of model performances across six reasoning datasets. The varying methodologies are color-coded to provide clear differentiation. This visualization underscores the differential strengths of each approach across datasets.

correct, indicating that they lack self-awareness. That is to say, after generating answers, the checking tasks based on prompt engineering are challenging for large language models.

The Table 3 differences on GPT-4 and GPT-3.5 (AQuA). We can see that when the "Break" is set to "Yes", the model is allowed to split the steps and verify each step. This is essential for understanding the self-awareness of the model. When a break is taken (i.e., processes are split and verified step-by-step), the model's self-awareness, as captured by the $Accuracy_v$ metric, is generally higher. This suggests that allowing the model to verify its steps improves its self-consistency. However, counter-intuitively, when no break is taken, there is an increase in the model's ability to recognize its mistakes, as indicated by the drop in SA_w when comparing rows where Break is 'No' versus 'Yes'.

The most evident observation is the discrepancy in the SA_w (Self Awareness for Wrong Answers) metric across different configurations. When we employ GPT-3.5 for both thinking and checking, the SA_w score is only 0.05, which suggests that the model rarely recognizes its mistakes. This is a significant shortcoming, as the model often believes its incorrect outputs are correct. On the other hand, when we use GPT-4 for thinking and GPT-3.5 for checking, the SA_w score increases to 0.74, showcasing the added advantage of having the GPT-4 model in the process. In addition to this, model fusion can effectively use the capabilities of different models. For example, after using GPT-4 for short thinking and generating prompts, the prediction accuracy of the GPT-3.5 model increased from 62% to 72%. Even the GPT-4 model is only used for thinking, it improves the performance a lot.

Break	Think	Check	Acc.	Acc_v	SA_w	SA_c
Yes	v3.5	v3.5	0.62	0.56	0.05	0.13
Yes	v4	v3.5	0.72	0.62	0.00	0.14
Yes	v3.5	v4	0.62	0.74	0.74	0.26
No	v3.5	v3.5	0.62	0.68	0.21	0.03
No	v4	v3.5	0.72	0.74	0.07	0.00
No	v3.5	v4	0.62	0.90	0.95	0.13

Table 3: Results from combining different versions of GPT for thinking and checking, under varying configurations with AQuA dataset. The table explores the interplay between breaking down the reasoning steps, model versions for thinking and checking, and the subsequent impact on accuracy and self-awareness metrics. Specifically, "Break" indicates if the reasoning steps are broken down for verification.

In conclusion, the results emphasize the importance of splitting steps, which allows for better verification and consequently improved model self-awareness. GPT-4's involvement in the thinking phase consistently boosts performance, underlining its enhanced capabilities compared to GPT-3.5. However, even with these improvements, the models' overconfidence in their outputs, especially when wrong, remains a significant challenge and highlights the importance of developing models with better self-awareness and introspection capabilities.

N	ACC	ACC_S	ACC_R	ACC_v	SA_w	SA_c
1	63.8	63.8	63.8	59.4	22.3	19.4
5	66.1	67.3	72.0	**61.0**	15.2	4.6
10	**67.3**	**72.8**	79.1	57.7	14.1	4.1
20	**67.3**	71.3	**79.9**	55.3	12.5	4.3

Table 4: Performance metrics of our approach on the AQuA dataset across different numbers of solvers. The column "N" represents the number of solvers used. Metrics such as the baseline accuracy (ACC), the accuracy after using our selection algorithm (ACC_S), the accuracy when dropping outputs with low confidence (ACC_R), and the introspective accuracy (ACC_v) are used to evaluate the model's performance and introspective capabilities.

4.3. Verification Task: Model Trustworthiness

After completing the inference task, we need to verify the results. However, the model's self-awareness flaw means it can't discover vulnerabilities in its direct reasoning process, presenting a challenge for the verification step. Thanks to the self-consistency method, we can provide confidence scores and enhance model performance. Existing self-consistency is based on majority voting, naturally yielding a confidence level. For example, if there are 8 answers out of 10 candidate answers that are 10, then the answer is 10, and the confidence is 80%. However, based on the idea that "truth is often in the hands of a few", we have improved this voting algorithm. As described in the steps of Algorithm 1, we use a threshold of 0.5 and experiment on the representative AQuA dataset. Table 5 offers a comprehensive view of the relationship between the number of CoT paths and accuracy on the AquA dataset. In the pursuit of enhancing model reliability, we conducted a systematic evaluation of our methodology across varying solver configurations, as delineated in Table 4. The overarching trend observed is the amplification of model accuracy with the incremental addition of solvers. Specifically, when the solver count escalates from 1 to 20, the overall accuracy demonstrates a subtle yet consistent augmentation, stabilizing at an optimal 67.3% for both 10 and 20 solvers. This saturation suggests a potential equilibrium, where the inclusion of additional solvers might not provide substantial gains in overall accuracy.

Delving deeper into the post-algorithmic performance ACC_S, our methodology distinctly outperforms the baseline, especially with an ensemble of 10 solvers, where it reaches a zenith of 72.8%. This accentuates the prowess of our algorithm in judiciously selecting outputs with heightened precision when furnished with an expansive solver pool. Furthermore, the efficacy of our algorithm becomes conspicuously evident when examining the ACC_R metric. As the solver ensemble expands, our algorithm exhibits an enhanced aptitude in effectively segregating less confident predictions, culminating in a peak accuracy of 79.9% with 20 solvers for the discarded outputs. However, a point of introspection emerges when assessing the model's introspective accuracy ACC_v. The marginal decline from 59.4% to 55.4% as solvers increase indicates potential areas for refining the model's introspective capabilities. From a self-awareness perspective, counterintuitively, the decline in SA_w from 22% to 12% as we transition from a singular solver to a group of 20 underscores our model's dropping proficiency in identifying its own errors. In stark contrast, the SA_c remains relatively invariant, hovering around 4% irrespective of the solver count, signifying consistent model behavior in instances where it doubts its correct answers.

In summary, our approach stands out by employing an ensemble of 10 solvers and achieving state-of-the-art results on the AQuA dataset using a single prompt with the GPT-3.5 model. The practical outcomes highlight both the performance of our method in optimizing output selection and its effectiveness in distinguishing dependable predictions. Nonetheless, additional scrutiny and enhancement are needed to delve into the introspective capabilities during the verification phase.

4.4. Explanation Task: Assessing the Model's Ability to Justify its Answers

When evaluating the reasoning performance, for multiple-choice questions or mathematical problems with unique solutions, answer verification can often be accomplished through straightforward string processing methods like regular expressions. However, evaluating T and S poses its set of challenges. A comprehensive assessment

Method	AQuA (%)		GSM8K (%)		SVAMP (%)	
	Logic	Calc.	Logic	Calc.	Logic	Calc.
Base (S)	92.2	86.2	99.8	92.2	94.6	92.0
Base (H)	66.4	94.0	86.2	98.1	82.7	98.9
Break (S)	76.3	72.1	58.8	54.9	74.3	68.2
Break (H)	66.1	96.2	84.3	99.4	88.6	99.4

Table 5: Comparative performances of GPT-3.5 and human evaluators in explanation task. In the 'Method' column, 'Base' and 'Break' represent no breaking (baseline) and breaking the solution into steps respectively. 'S' stands for self-checking, and 'H' represents human-checking.

of T and S given A demands an intricate understanding of semantic information.

The advent of Large Language Models (LLMs) has ushered in an era where automated evaluations based on semantic understanding are becoming feasible. To ensure the rigor and validity of our approach, we set human evaluations as our benchmark. We then juxtaposed the performance of ChatGPT in assessing the correctness of the reasoning and problem-solving process.

Table 5 provides an insightful comparison between ChatGPT and human evaluators regarding their ability to assess reasoning and problem-solving processes. In the baseline GPT3.5 model, without breaking the answers into steps, there is a pronounced disparity between the model's self-assessment and the human evaluators' assessment. For example, in the AQuA dataset, the model's self-estimated logical accuracy stands at 92.2%, whereas human evaluators rate it substantially lower at 66.4%. This pattern persists across the GSM8K and SVAMP datasets, indicating that the model might be overconfident about its logical reasoning capabilities.

Introducing the 'break' methodology, where solutions are evaluated step-by-step, reveals a significant shift in model self-assessment. While the accuracy of human evaluators remains relatively consistent or even slightly improved, especially in terms of logical reasoning, the model's self-assessed accuracy sees a significant decline, especially in the GSM8K dataset. The 'break' method indeed tempers the model's tendency to overestimate its capabilities. However, there's a catch. The process might inadvertently push the model towards underestimating its outputs, especially evident in the calculation accuracy metric. This shift from overestimation to potential underestimation reiterates the complexity of achieving a balanced evaluation.

5. Conclusion

In this study, we embarked on a comprehensive exploration of the capabilities and potential limitations of LLMs, specifically focusing on their application in mathematical reasoning. Our research was anchored in the dual objectives of enhancing model trustworthiness and evaluating the model's self-awareness.

Our findings underscored the inherent challenges faced by models in self-verification tasks. We observed that while models displayed considerable prowess in problem-solving, they often lacked introspective capabilities, sometimes expressing unwarranted confidence in incorrect answers. However, through systematic evaluations and methodological refinements, we introduced a calibration approach that significantly amplified model accuracy and trustworthiness. In the realm of explanation tasks, we highlighted the advancements made by LLMs, emphasizing their potential in automated evaluations grounded in semantic understanding. Though models like GPT-4 are impressive, achieving comprehensive understanding in AI is an ongoing journey. We aspire for our research to guide future endeavors towards more trustworthy and human-like AI models.

6. Acknowledgements

This paper is supported by the National Natural Science Foundation of China (Grant No. 62192783, 62376117), the Collaborative Innovation Center of Novel Software Technology and Industrialization at Nanjing University. Thanks for the National Natural Science Foundation of China for their support. In addition, we also thank the members of the IIP research group of Nanjing University for their support of this paper.

7. Bibliographical References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian

Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report.

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.

BSI. 1973a. *Natural Fibre Twines*, 3rd edition. British Standards Institution, London. BS 2570.

BSI. 1973b. Natural fibre twines. BS 2570, British Standards Institution, London. 3rd. edn.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations.

A. Castor and L. E. Pollux. 1992. The use of user modelling to guide inference and learning. *Applied Intelligence*, 2(1):37–53.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

J.L. Chercheur. 1994. *Case-Based Reasoning*, 2nd edition. Morgan Kaufman Publishers, San Mateo, CA.

N. Chomsky. 1973. Conditions on transformations. In *A festschrift for Morris Halle*, New York. Holt, Rinehart & Winston.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

Umberto Eco. 1990. *The Limits of Interpretation*. Indian University Press.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Paul Gerhard Hoel. 1971a. *Elementary Statistics*, 3rd edition. Wiley series in probability and mathematical statistics. Wiley, New York, Chichester. ISBN 0 471 40300.

Paul Gerhard Hoel. 1971b. *Elementary Statistics*, 3rd edition, Wiley series in probability and mathematical statistics, pages 19–33. Wiley, New York, Chichester. ISBN 0 471 40300.

Otto Jespersen. 1922. *Language: Its Nature, Development, and Origin*. Allen and Unwin.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *arXiv preprint arXiv:2306.03872*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. 2021. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. *Advances in Neural Information Processing Systems*, 34:25192–25204.

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.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Chenglei Si, Weijia Shi, Chen Zhao, Luke Zettlemoyer, and Jordan Boyd-Graber. 2023. Mixture of prompt experts for generalizable and interpretable question answering. *arXiv preprint arXiv:2305.14628*.

Charles Joseph Singer, E. J. Holmyard, and A. R. Hall, editors. 1954–58. *A history of technology*. Oxford University Press, London. 5 vol.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3746–3753, Istanbul, Turkey. European Language Resource Association (ELRA).

S. Superman, B. Batman, C. Catwoman, and S. Spiderman. 2000. *Superheroes experiences with books*, 20th edition. The Phantom Editors Associates, Gotham City.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

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, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

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 preprint arXiv:2210.03629*.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? *arXiv preprint arXiv:2305.18153*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Jin Ziqi and Wei Lu. 2023. Tab-cot: Zero-shot tabular chain of thought. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10259–10277.

8. Language Resource References

Cobbe, Karl and Kosaraju, Vineet and Bavarian, Mohammad and Chen, Mark and Jun, Heewoo and Kaiser, Lukasz and Plappert, Matthias and Tworek, Jerry and Hilton, Jacob and Nakano, Reiichiro and others. 2021. *Training verifiers to solve math word problems*. PID https://github.com/openai/grade-school-math.

Dan Hendrycks and Collin Burns and Saurav Kadavath and Akul Arora and Steven Basart and Eric Tang and Dawn Song and Jacob Steinhardt. 2021. *Measuring Mathematical Problem Solving With the MATH Dataset*. PID https://github.com/hendrycks/math.

Hosseini, Mohammad Javad and Hajishirzi, Hannaneh and Etzioni, Oren and Kushman, Nate. 2014. *Learning to solve arithmetic word problems with verb categorization*. PID https://www.cs.washington.edu/nlp/arithmetic.

Ling, Wang and Yogatama, Dani and Dyer, Chris and Blunsom, Phil. 2017. *Program induction by rationale generation: Learning to solve and explain algebraic word problems*. PID https://github.com/google-deepmind/AQuA.

Miao, Shen-yun and Liang, Chao-Chun and Su, Keh-Yih. 2020. *A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers*. PID https://github.com/chaochun/nlu-asdiv-dataset.

Patel, Arkil and Bhattamishra, Satwik and Goyal, Navin. 2021. *Are NLP models really able to solve simple math word problems?* PID https://github.com/arkilpatel/SVAMP.

Roy, Subhro and Roth, Dan. 2016. *Solving general arithmetic word problems*. PID https://arxiv.org/abs/1608.01413.