General Purpose Verification for Chain of Thought Prompting

Anonymous ACL submission

Abstract

Many of the recent capabilities demonstrated by Large Language Models (LLMs) arise primarily from their ability to exploit contextual information. In this paper, we explore ways to improve reasoning capabilities of LLMs through (1) exploration of different chains of thought and (2) validation of the individual steps of the reasoning process. We propose three general principles that a model should adhere to while reasoning: (i) Relevance, (ii) Mathematical Accuracy, and (iii) Logical Consistency. We apply these constraints to the reasoning steps generated by the LLM to improve the accuracy of the final generation. The constraints are applied in the form of verifiers: the model itself is asked to verify if the generated steps satisfy each constraint. To further steer the generations towards high-quality solutions, we use the perplexity of the reasoning steps as an additional verifier. We evaluate our method on 4 distinct types of reasoning tasks, spanning a total of 9 different datasets. Experiments show that our method is always better than vanilla generation, and, in 6 out of the 9 datasets, it is better than best-of N sampling which samples N reasoning chains and picks the lowest perplexity generation.

1 Introduction

Large Language Models (LLMs) have demonstrated impressive capabilities of performing a diverse range of tasks by framing them as text generation (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023; OpenAI, 2023; Bubeck et al., 2023, *inter alia*). Chain-of-Thought prompting (Nye et al., 2021; Wei et al., 2022; Chowdhery et al., 2022) further improved their performance on challenging reasoning tasks using a simple trick of generating intermediate steps before giving the final answer allowing the LLM to spread computation over more tokens (Goyal et al., 2023). However, this approach lacks a mechanism to rectify errors in reasoning. While LLMs may eventually reach the correct answer, they might do so via incorrect intermediate reasoning steps, or worse, never reach the correct answer due to earlier mistakes (Turpin et al., 2023). To illustrate this, we provide a concrete example in Figure 1, where the final answer is correct, but the intermediate steps are (i) irrelevant (Shi et al., 2023), (ii) contradicting previous steps (Mündler et al., 2023), and (iii) with mathematical errors (Patel et al., 2021). Recent work (Yao et al., 2023; Xie et al., 2023; Pan et al., 2023) has attempted to alleviate these problems by employing a search mechanism or a self-correction mechanism in the spirit of "System 2" thinking. Other directions include training a dataset-specific verifier to improve the performance when aggregating multiple reasoning chains (Li et al., 2023). However, all these approaches have dataset-specific adaptations and don't generalize out-of-the-box.

In this work, we explore if catching early mistakes in reasoning chains through problem-agnostic verification can improve reasoning in LLMs. We propose three general principles that a model should adhere to while reasoning: (i) Relevance, (ii) Mathematical Accuracy, and (iii) Logical Consistency and use models, called verifiers, to test for each principle. Each verifier operates on a step (Uesato et al., 2022) generated from the step-by-step manner of Chain-of-Thought prompting and assigns a score to that step. We design the verifiers to operate at this granularity so they can detect intermediate mistakes and discourage the LLM from committing to an erroneous reasoning chain. To further steer the generation towards better steps, we use the perplexity of the reasoning step as an additional verifier. We then explore various ways, including Self-Consistency (Wang et al., 2022), to aggregate verifier scores and report their downstream task performance.

We make the following contributions: (i) we propose a general framework for guiding reasoning in

Question: A concert ticket costs $40. Mr. Doe bought 10. Mr. Benson bought 12 tickets and received a 5% discount for every ticket bought that exceeds 10. How much did Mr. Benson pay in all?

Step by Step Solution:

1. Mr. Doe paid in total 10 * $40 = $400. (Irrelevant)
2. Without the discount, Mr. Benson paid in total 12 * 40 = $480.
3. Mr. Benson received a discount of 2 * ($40 - $38) = $4 ($2 per ticket)
4. So Mr. Benson received a discount of $0.02 per ticket. (Inconsistent)
5. With discount, Mr. Benson paid $480 - 2 * $2 = $478. (Mathematically inaccurate)
6. Therefore, Mr. Benson paid $476.

Figure 1: An illustrative example highlighting how the final answer can ultimately be correct (i.e. $12 * \$40 - (12 - 10) * \$40 * 0.05 = \$476$), but it is reached through steps that are (i) irrelevant (Step 1), (ii) contradicting previous steps (Step 4 contradicts Step 3), or (iii) with mathematical errors (Step 5).

LLMs using verifiers which offers the flexibility to use a problem-agnostic implementation across any reasoning task but also offers the adaptability to use task- and dataset-specific implementations, and (ii) we show how using our proposed verifiers can improve reasoning outcomes in LLMs and can also improve existing ensembling techniques like Self-Consistency. Importantly, our work is not intended to be an exploration on the best way to use a computational budget to achieve a desired performance, but an exploration of whether the LLM are capable (even if inefficiently) of detecting their own mistakes together with a simple recovering mechanism.

2 Related Work

We focus here only on LLM-based approaches, and divide previous related work according to (i) the generalizability of the prompts used, and (ii) how the final answer is generated.

Types of Prompts In prior work, the prompts used can be categorized based on their level of generality. Some approaches utilize a singular prompt, applying it uniformly across a wide spectrum of datasets and tasks. Wei et al. (2022) proposed chain-of-thought prompting with in-context examples. Kojima et al. (2022) then explored zero-shot prompts capable of exhibiting similar behaviors. Other recent works explore using LLMs to self-evaluate (Yin et al., 2023) and potentially improve upon their generation with the resulting feedback (Saunders et al., 2022; Chen et al., 2023; Pan et al., 2023; Shinn et al., 2023). Bai et al. (2022) use an LLM with in-context examples to detect and edit the responses of a chat model that are harmful or toxic. Madaan et al. (2023) proposes a framework to iteratively self-improve the generations of a LLM. Yao et al. (2023) tightly integrates an LLM with custom dataset-specific prompts to act as a guiding mechanism in the underlying search space. Hao et al. (2023) expands on this by using a Monte Carlo tree search strategy. Other recent work questioned the extent to which using an LLM to evaluate and improve its own generations is viable (Huang et al., 2023), a conclusion which we observed as well in our preliminary work and sidestepped by re-sampling instead of asking the LLM to refine.

Importantly, previous work explored self-evaluation through the lens of task-specific evaluation and prompts, a direction that inherently constrains the broader utility of Large Language Models (LLMs) as general-purpose reasoners. On the other hand, our approach follows a distinct trajectory: we deliberately eschew the use of prompts tailored to individual datasets or tasks.

How the Final Answer is Generated A second dimension is that of how the model arrives at the final solution, where we distinguish between methods that take a linear approach (Wei et al., 2022; Kojima et al., 2022; Goyal et al., 2023) from the methods that do not (Yao et al., 2022; Wang et al., 2023; Long, 2023). By linear approaches, we refer to those methods where the final answer is generated token-by-token in one go. On the other hand, non-linear approaches typically include a search mechanism (Xie et al., 2023; Yao et al., 2023; Besta et al., 2023) or a self-reflection process (Madaan et al., 2023; Pan et al., 2023).

For example, recent work explored tightly integrating the LLM to act as a guiding mechanism in the underlying search space (Xie et al., 2023; Yao et al., 2023). This involves one LLM generating candidate steps while another LLM assigns single float value as a value score. This value score is derived from an LLM with a dataset-specific prompt and in-context examples.

Within this dimension, our approach aligns with the non-linear paradigm. We leverage verifiers to evaluate each step in the solution-generation process, with the overarching aim of guiding the generation towards solutions that receive high scores, as determined by the verifiers. Differently from previous work on self-evaluation, we explore a setting of self-evaluation that is problem-agnostic.

3 Proposed Method

We propose a novel approach that seamlessly integrates with any given Large Language Model's (LLM) solution generation process, at both step generation and step evaluation (Wei et al., 2022; Wang et al., 2022; Kojima et al., 2022). Our approach consists of two components: a solution generator G and a set of verifiers $\mathcal{V}$, where each verifier specializes in a particular qualitative aspect of reasoning. The solution generator G is responsible for generating candidate steps, and each verifier $v \in \mathcal{V}$ is responsible for checking whether the candidate step is in compliance with the specific reasoning property. We explore properties that are generally applicable to a wide range of reasoning tasks. We provide an illustrative example in Figure 2.

This section is further organized as follows. We describe the notations and abstractions used in Section 3.1, the solution generation component in Section 3.2, the proposed verifiers in Section 3.3, the procedure to obtain fine-grained scores for reasoning chains in Section 3.4 and how we use verifiers in aggregate in Section 3.5.

3.1 Notation

In the following, we define the notation we adopt throughout the paper.

A token as $t \in \texttt{Vocab}$ where $\texttt{Vocab}$ represents the set of possible tokens defined by a given vocabulary. We use $[t_1, \ldots, t_n] \in T$ to represent a given text, with T denoting the set of all potential texts of varying lengths. Under this notation, we represent a problem as $q \in T$ and a reasoning step as $r \in T$, both presented in free-text form. A solution generator G in the form of a function that takes text as input and returns text as output: $G : T \rightarrow T$. We interpret the output text as a sequence of reasoning steps $R^q = [r_1^q, \ldots, r_n^q]$. For simplicity, we define a reasoning step as the sequence of tokens until a new line, similar to Uesato et al. (2022). A verifier $v \in \mathcal{V}$ implemented a function that takes text as input and returns an indicator: $v : T \rightarrow \{0,\ 1\}$. The returned value represents whether the reasoning step satisfies the verifier constraint (1) or not (0).

We will next describe the Solution Generator and Step Verification components.

3.2 Solution Generation

The solution generator (typically an LLM) operates over a prompt q and generates a sequence of tokens as output: $G : T \rightarrow T$. For our purpose, we concentrate on the correctness at the level of a reasoning step, instead of individual tokens. A reasoning step, as defined in this work, represents the sequence of tokens up to the occurrence of a new line (Uesato et al., 2022). The next reasoning step can then be generated by conditioning the generator G on both the question q and on the sequence of previously generated reasoning steps $R_{1:i}$, which is initially empty. This conditioning can be expressed as follows: $P(r_{i+1}^q | q, R_{1:i}^q)$.

For the solution generation process, we adopt the zero-shot prompt used in Kojima et al. (2022) which simply appends "Let's think step by step" to the problem question to elicit a chain-of-thought like behavior in the model's response without any annotated exemplars (Wei et al., 2022). Nevertheless, our proposed method is agnostic to the specific implementation of the solution generator.

3.3 Step Verification

Our research aims to investigate whether specifying a subset of conditions that an ideal reasoning chain should satisfy and then employing these conditions to score the corresponding reasoning chain can result in improved performance on downstream reasoning tasks.

To this end, we explore three general and necessary (but not sufficient) conditions that a given step should satisfy in order for the resulting solution to be sound from a reasoning perspective: (i) Relevance, (ii) Mathematical Accuracy (if applicable), and (iii) Logical Coherence. In this work, we use for our verifiers a set of LLMs provided with a detailed instruction of the task and constraints.[1] We then map the output of the LLM to $\{0, 1\}$ based on its content. If the relevance verifier generates "not relevant", for example, we interpret this as a score of 0. Importantly, to verify the generalizability of our proposed methodology, we keep the implementation of our verifiers fixed for all reasoning tasks and for all datasets. We provide an illustrative example of the verifiers we use in Figure 2. We provide in-context exemplars for the Mathematical Accuracy verifier due to challenges in the LLM's ability to generate a valid intermediate structured output, a requirement of this verifier's setup.[2] In addition to the scores from the aforementioned three verifiers, we use the perplexity score of a reasoning step as an additional verifier, in order to encourage

[1] Prompts available in Appendix C

[2] Same exemplars were used for all Math datasets.

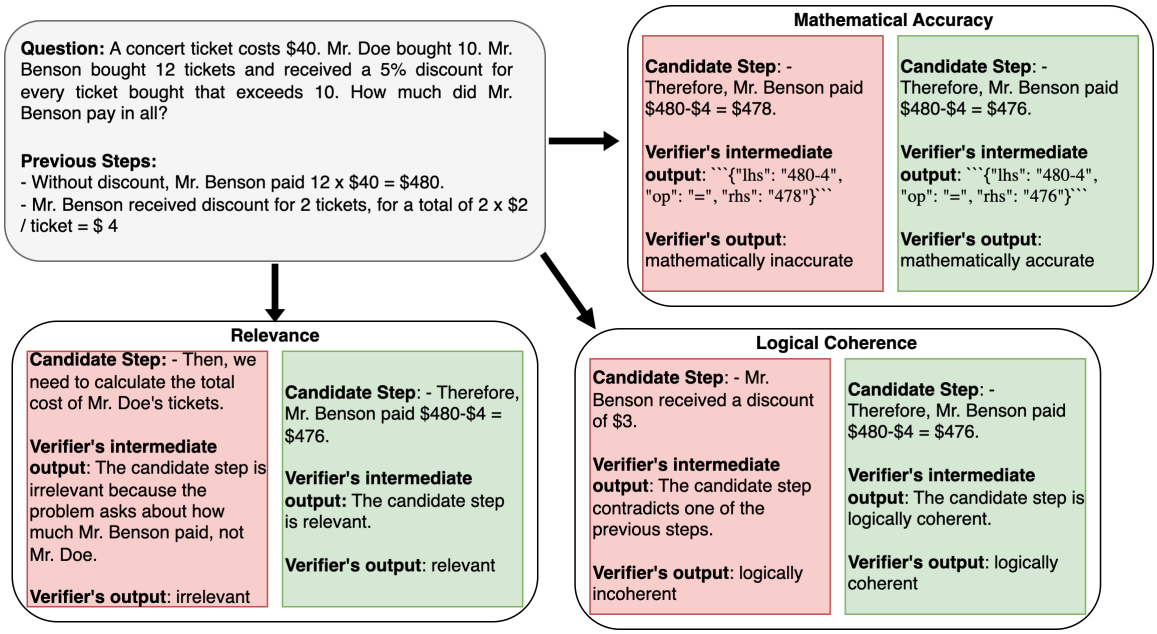


Figure 2: An example of each of our proposed verifiers applied to a given question and previous steps.

the final solution towards text deemed more likely by the LLM. We explain each verifier in greater detail below.

3.3.1 Relevance

The first verifier in our proposed framework is the Relevance verifier, where the goal is to constrain a reasoning step to contribute to the construction of a meaningful solution narrative. We provide a *relevant* and an *irrelevant* example in Figure 2. We begin with a question, a sequence of previous steps, and a candidate step. The Relevance verifier assesses the candidate reasoning steps for their relevance to the problem at hand. In the example provided, calculating how much Mr. Doe spent is irrelevant because the problem asks for how much Mr. Benson spent and there is no connection between them.

We acknowledge the inherent subjectivity and nuance associated with determining the relevance of a given reasoning step. However, there are instances where it becomes clear that a reasoning step is distinctly irrelevant, deviating from the coherent solution narrative. For instance, some reasoning steps may veer into speculative or unrelated content, which our Relevance verifier aims to identify.

3.3.2 Mathematical Accuracy

The Mathematical Accuracy constraint enforces the need for each reasoning step to contain correct mathematical calculations. We implement this in a similar manner to Tool-based approaches (Schick et al., 2023), working as follows. First, we extract the mathematical formulas (if present) from a sentence as structured output, as depicted in the *Verifier's intermediate output* field corresponding to the Mathematical Accuracy constraint in Figure 2. For each mathematical calculation present, we extract the left-hand side (*lhs*), the right-hand side (*rhs*), and the operator (*op*). Then, we programmatically execute the extracted formulas (if any) and compare them using the extracted operator.

3.3.3 Logical Consistency

A third condition for a logically sound argument we use is for the reasoning steps to not contradict each other (M. and Mckeon, 1941). To this end, we introduce Logical Consistency as our third verifier. This verifier operates over the previous steps and the current candidate step. For example, in Figure 2, the candidate step - *Mr. Benson received a discount of $3.* contradicts one of the previous steps, as one of the previous steps already established that Mr. Benson received a discount of $4.

3.3.4 Step-wise Perplexity

In addition to the scores resulting from our previously introduced constraint verifiers, we leverage step-wise perplexity as another source of signal, with the goal of favoring lower-perplexity solutions. For each reasoning step $r_i = [t_1, \ldots, t_n]$, we compute the perplexity over its token constituents. We hypothesize that lower-perplexity reasoning steps are more desirable, as a lower perplexity prompt is correlated with a higher final performance (Gonen et al., 2022). We can interpret a partial reasoning

chain $R_{1:i} = [r_1, \ldots, r_i]$ as (part of) a prompt that will be used to generate r_{i+1}, making the findings in Gonen et al. (2022) applicable for our purposes.

3.4 Constraint Satisfaction Score

Except for the Perplexity verifier, all our proposed verifiers output a binary value, representing whether a given reasoning step satisfies the given constraint or not. For example, if the Relevance verifier gives a score of 1 for a given reasoning step r, this means that the given step is deemed as relevant. In order to reduce the variance and get a more fine-grained score s, we use the expected value: $s = \mathbb{E}(\mathbb{1}_v(r))$, which we approximate using sampling. Since each verifier is implemented with an LLM, we can sample multiple generations, map each one to a binary value $\{0, 1\}$, and then average.

3.5 Using Verifiers in Aggregate

3.5.1 Scoring a Reasoning Chain R

Given a verifier $v \in V$, we extend the concept of a score for a given reasoning step r to the score for a given (partial or not) reasoning chain R by aggregating the scores over each of its constituent reasoning steps. Formally, we extend the verifier's scores to that of a reasoning chain $R = [r_1, \ldots, r_i]$, where we first obtain a score for each r_i, resulting in the following score vector: $[\mathbb{E}(\mathbb{1}_{v(r_1)}), \ldots, \mathbb{E}(\mathbb{1}_{v(r_i)})]$, and then aggregate. A low-scoring reasoning step does not necessarily render the entire reasoning chain wrong, but it does increase the likelihood of inaccuracies. To combine these scores, we employ the geometric mean as a milder alternative to the *min* operator in our aggregation process: $v(R) = GM([\mathbb{E}(\mathbb{1}_v(r_1)), \ldots, \mathbb{E}(\mathbb{1}_v(r_i))])$ We obtain a single score for a given reasoning chain R and a set of verifiers $\mathcal{V}$ by aggregating over the scores of each verifier $v \in \mathcal{V}$ on R. Our proposed framework allows for the customization of each verifier's contribution during aggregation. We use a weighted arithmetic mean, as defined below.

$$\mathcal{V}(R) = \frac{\sum_{i=1}^{|\mathcal{V}|} w_i \times v_i(R)}{\sum_{i=1}^{|\mathcal{V}|} w_i}$$

We set $w = 2$ for perplexity and $w = 1$ for all the others. We selected $w = 2$ for perplexity based on preliminary experiments on the train partition of GSM8k and CSQA 2.0. Importantly, we use the same weights for all our experiments.

3.5.2 Ensembling Methods using the Verifiers

Ensembling techniques work by aggregating the solution of multiple reasoning chains to obtain a final solution. For example, Self-Consistency (Wang et al., 2022) randomly samples a given number of reasoning chains, and then performs a majority vote on the final answer. Instead of resorting to a majority voting mechanism over randomly sampled reasoning chains, we propose to leverage the scores obtained from our proposed verifiers to do the selection and the weighting.

4 Experiments

4.1 Experimental Setting

Models We use *Falcon*[3] (Almazrouei et al., 2023) as our base LLM, as it was one of the largest and most capable open-source model family freely available at the time of the experiments.[4] We use the same model for both solution generation and solution verification. For solution generation, we use the zero-shot prompt from Kojima et al. (2022). For verification, we use different prompts for each verifier, available in Appendix C.

Datasets We perform experiments spanning 4 reasoning tasks: Math, Commonsense, Symbolic, and Other, and 9 datasets: BigBench Date Understanding (bench authors, 2023) (*Other*), CommonsenseQA (Talmor et al., 2019), CommonsenseQA 2.0 (Talmor et al., 2021) and Strategy (Geva et al., 2021) (*Commonsense*), Coinflip and Last Letter Concatenation (Wei et al., 2022) (*Symbolic*), GSM8k (Cobbe et al., 2021), SVAMP (Patel et al., 2021), and AddSub (Kojima et al., 2022) (*Math*). We show an example from each dataset in Appendix B.1.[5]

We use the standard evaluation metrics as previous work, which is Accuracy score computed between the gold answer and the predicted answer.

Proposed Method Setting All our proposed verifiers are dataset-agnostic and we use the same prompts for all our experiments. Due to computational constraints, we use the mathematical accuracy verifier only for the math datasets. In an attempt to minimize the impact of the underlying search strategy for the step-by-step solution, we adopt the following approach: we first sample 40

[3] Specifically, we use *Falcon-40B-Instruct*

[4] Open source according to https://opensource.org/

[5] For Last Letter Concatenation, we use only 2 words instead of 4.

	Other	Commonsense			Symbolic		Math		
	BigBench Date	CSQA	CSQA 2.0	Strategy	Coinflip	Last Letter (2)	GSM8k	SVAMP	AddSub
Random Chain	55.77±3.06	47.91±1.22	58.75±1.68	56.03±0.99	58.67±2.04	15.68±1.51	29.23±1.58	38.78±1.26	41.04±1.79
Low PPL Chain	63.69±0.00	48.81±0.00	59.10±0.00	**60.90±0.00**	**72.80±0.00**	**45.00±0.00**	40.50±0.00	53.50±0.00	58.48±0.00
Top Chain wrt Verifiers	**69.12±0.21**	**56.79±0.12**	**62.16±0.22**	57.21±0.17	64.02±0.21	41.64±0.44	**45.94±0.30**	**56.36±0.23**	**62.34±0.25**

Table 1: Comparison between two baselines: (1) Random Chains, and (2) Low PPL Chain, and our proposed method, (3) Top Chain wrt Verifiers. In this setting, we record the performance when selecting *one* reasoning chain, according to each method's selection criteria. We report Accuracy (↑).

reasoning chains for each problem and for each dataset, then use the scores resulting from our proposed verifiers to guide our selection process.

Baselines We analyze the verifiers' contributions by comparing the performance of the proposed method against the following baselines: (i) *Random Chains*, where we use the LLM to sample a solution, using the same prompt as Kojima et al. (2022), and (ii) *best-of N* sampling (Adiwardana et al., 2020; Wang et al., 2022), where we sample a total of 40 reasoning chains and select the one with the lowest perplexity. Our motivation for using these two baselines is two-fold. First, we want to allow both the baselines and our proposed method to operate over the same candidate reasoning chains. Secondly, it has been observed in Wang et al. (2022) that Best-of N sampling performs better than greedy decoding, especially for large N.

Experiments We conduct the following experiments: (i) Single chain analysis, where we use the resulting scores of each reasoning chain to select a single reasoning chain, (ii) Self-Consistency, where we aggregate multiple reasoning chains, and (iii) Single chain analysis with incomplete chain scoring, where we only score the reasoning chains based on the an initial % (or number) of the reasoning steps.

4.2 Single Chain

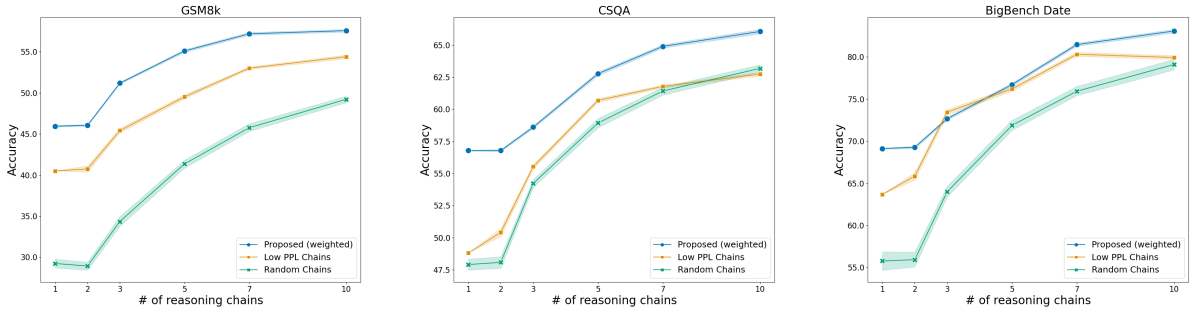
In this experiment, we assess how the scores generated by our proposed verifiers are correlated with the likelihood of a reasoning chain reaching the correct final answer. For this purpose, we conduct the following experiment: from the 40 sampled reasoning chains, we select the highest-scoring chain based on our proposed verifiers' scores. We present our results in Table 1, comparing with two baselines: *Random Chains* and *best-of N* sampling. We make the following remarks.

First, over all datasets, our proposed method performs better than selecting a reasoning chain at random, with improvements ranging from 1.18 points (Strategy) to 25.68 points (Last Letter), with an average improvement of 12.63.

Second, we remark that our proposed method outperforms the best reasoning chain according to perplexity (Low PPL Chains) in over 65% of the cases (6 out of 9 datasets). This means that our proposed verification procedure provides valuable information beyond what is captured by simply selecting the lowest perplexity chains. We note that this trend does not hold true for Symbolic Reasoning, where for both datasets investigated (*Coinflip* and *Last Letter*) the Low PPL Chain is better than the one selected according to our proposed verifier. An exploration over *Coinflip* revealed that steps where the coin has not been flipped received, on average, a lower relevance score, although this information is relevant. We leave the exploration of better verifiers for Symbolic Reasoning to future work. All in all, the average improvement of our proposed method over the reasoning chain with the lowest perplexity is 1.43 points.

4.3 Self-Consistency

In this experiment, we explore how well our proposed method leverages ensemble techniques, particularly Self-Consistency (Wang et al., 2022). We start with the same set of 40 reasoning paths and employ different selection strategies to evaluate their effectiveness: (i) we randomly sample from these paths (*Random Chains*), (ii) we select chains with the lowest perplexity from this set (*Low PPL Chains*), and (iii) we choose the reasoning chains with high scores, as determined by the verifiers, from these 40 paths (*Proposed (weighted)*). Unlike the original majority vote approach, we empirically found that weighting each reasoning chain by their verifier scores yields slightly better results. However, it is worth noting that this improvement does not hold when using only perplexity, as shown in (Wang et al., 2022). We show in Figure 3 the behavior of our proposed method. We make two observations: First, our proposed method is able to leverage ensembling techniques, showing consistent performance gains as the number of reasoning chains increases. Second, we remark that our



(a) GSM8k (b) CSQA (c) BigBench Date Understanding

Figure 3: Comparison between our proposed method and two baselines, when using Self-Consistency and between 1-10 reasoning chains. We report Accuracy (↑).

proposed method scales better, consistently outperforming the baselines.

We further investigate the performance impact of weighted voting, utilizing scores from our proposed verifiers, against the standard majority-voting approach. Specifically, we apply both voting methods to the *identical set* of reasoning chains, initially selected at random. We found that using the scores of our proposed verifiers to do a weighted voting improves over the majority voting in over 96% of the cases.[6] Due to space constraints, we include the resulting plots in Appendix E.

4.4 Verifying Incomplete Reasoning Chains

In our prior experiments, our proposed verifiers evaluated complete reasoning chains. Now, we explore their effectiveness when applied exclusively to the initial reasoning steps. This experiment provides insights into the potential utility of our proposed method in an "online" setting, where the reasoning step-level evaluation is employed to guide the search for good reasoning chains without fully generating multiple candidate solutions.

We assess the impact of using the verifiers for varying percentages of reasoning steps, denoted as X% along the X-axis of our line plot in Figure 4. We remark that the final performance increases with the % of steps verified and that verifying only the first 20% of the steps is sufficient to increase the final performance beyond that of random chains.

Since knowing beforehand the total number of reasoning steps is unrealistic, we also experiment with only verifying a given number of the initial reasoning steps. Due to space limitations we include these results in Appendix G. Additionally, we include in Table 5 the resulting performance when verifying between 0 and *All* reasoning steps

Verifiers				Math		
P	R	M	C	AddSub	GSM8k	SVAMP
✗	✗	✗	✗	41.04±1.79	29.23±1.58	38.78±1.26
✗	✗	✗	✓	48.84±0.85	33.51±0.69	45.41±0.56
✗	✗	✓	✗	45.76±1.99	35.59±1.60	41.68±1.65
✗	✓	✗	✗	49.04±0.44	33.38±0.52	46.98±0.42
✓	✗	✗	✗	59.09±0.59	41.53±0.49	53.85±0.45
✓	✓	✓	✓	**62.34±0.25**	**45.94±0.30**	**56.36±0.23**

(a) Ablation over Math datasets.

Verifiers			Other	Commonsense			Symbolic	
P	R	C	BigBench Date	CSQA2.0	CSQA	Strategy	Coinflip	Last Letter
✗	✗	✗	55.77±3.06	58.75±1.68	47.91±1.22	56.03±0.99	58.67±2.04	15.68±1.51
✗	✗	✓	62.70±0.68	56.77±0.55	53.00±0.59	54.31±0.70	49.17±0.76	20.63±0.78
✗	✓	✗	62.57±0.45	61.03±0.21	52.62±0.29	57.31±0.26	53.63±0.51	15.79±0.30
✓	✗	✗	64.06±0.60	60.27±0.25	51.35±0.36	**60.03±0.30**	**73.36±0.42**	**41.73±0.48**
✓	✓	✓	**69.12±0.21**	**62.16±0.22**	**56.79±0.12**	57.21±0.17	64.02±0.21	41.64±0.44

(b) Ablation over Non-Math datasets.

Table 2: Ablation study on the effect of each verifier on the downstream tasks when selecting a *single* reasoning chain. We differentiate between math and non-math datasets.

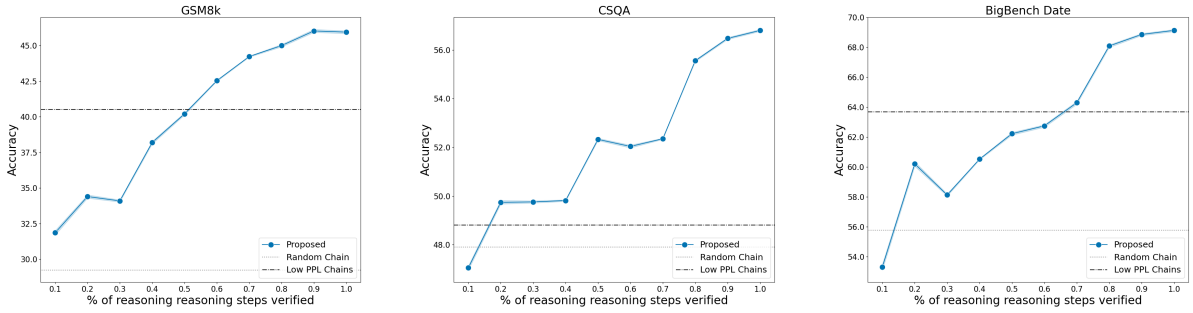
over all datasets. In 7/9 cases, the performance increases even when verifying only the first step. When verifying the first two steps, the final performance increases in all the cases.

4.5 Contributions of each Verifier

In this experiment, we assess the contribution to the final performance of each of our verifiers: (1) Low Step **P**erplexity, (2) **R**elevance, (3) **M**athematical Accuracy (if applicable), (4) Logical **C**onsistency. First, we observe that each individual verifier is meaningfully contributing towards the final solution. For example, for the math datasets (Table 2a), employing *any* verifier improves the final performance, with improvements ranging from 2.90% to 21.30%. All in all, using as little as a single verifier improves the final performance in over 89% of the cases.[7] Secondly, we remark that combining all the verifiers gives further improvements, beyond those obtained by using a single verifier,

[6] 78/81

[7] 35/39



(a) GSM8k (b) CSQA (c) BigBench Date Understanding

Figure 4: Verifying only the first X% steps of a given reasoning chain. (↑).

suggesting that each verifier is adding meaningful and non-overlapping information. We note that there is a notable exception to this trend, where for the Symbolic Reasoning tasks (and for the Strategy dataset), a distinct combination of verifiers (i.e. only Perplexity) attains a better score than using all the verifiers. We provide results covering a wider range of verifier combinations in Appendix F.

4.6 Human Evaluation

While the proposed verifiers meaningfully contribute to the final performance on the reasoning downstream tasks, we perform a human evaluation study to assess: (1) how well they correlate with human judgment and (2) how reliably concepts such as logical consistency or relevance can be evaluated by humans. We include instructions and inter-annotator agreement scores in Appendix H.

We compute pearson correlation scores (in range $[-1, 1]$) between human assessments and the scores proposed by our verifiers and, additionally, also between GPT-4o and humans to evaluate how the proposed method scales with a stronger underlying model. We plot average correlations up to step K in Figure 5. We note the following.

First, we note that each verifier exhibits an overall significant (p-value < 0.0001) and positive correlation between human judgement and performance. When the correlation is negative, it is better to not use it (e.g., Coinflip has scores 69 (without) vs 64 (with) the negatively correlated logical consistency verifier, see 4b). We also show in Appendix I how even small positive correlations statistically differentiate better outcomes on average. Second, we observe (and show in Appendix H) a large variance in the inter-annotator agreement score, which we hypothesize comes from different reasoning styles between humans and noise from a weak underlying model (Falcon). We also explore different agreement metrics, aiming to capture some of the Cohen's Kappa Limitations (e.g., Cohen's Kappa Paradox (Zec et al., 2017)). Lastly, we found that less than 2% of the errors marked by the annotators are not captured by one of the principles explored.

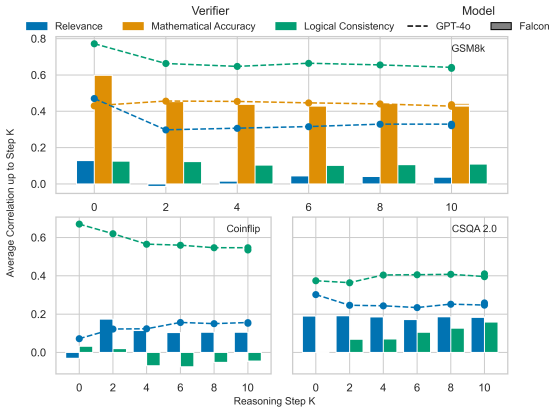


Figure 5: Correlation across datasets and steps

5 Conclusion

We explore a general-purpose verification procedure consisting of task- and dataset-agnostic verifiers at the reasoning step-level inspired by fundamental principles of sound reasoning: Relevance, Mathematical Accuracy and Logical consistency. On top of these reasoning principles, we leverage the perplexity of the reasoning step to steer the LLM towards high-quality solutions.

Across four distinct reasoning tasks, spanning nine datasets, we show that using the proposed verifiers to score the reasoning chains leads to notable performance improvements when compared to randomly sampled reasoning chains. Most notably, our proposed approach outperforms the lowest perplexity reasoning chain in over 6 out of the 9 datasets we tested. This indicates that the proposed verifiers provide additional valuable information beyond what is captured by the perplexity measure of the reasoning chain. We leave the exploration of better and more efficient verifiers to future work.

Limitations

While our proposed framework is flexible and admits different implementations for the verifiers, in this work we implemented each verifier with a prompt-based LLM approach. This type of implementation can increase the energy consumption of the deployed system, leading to a performance–energy-consumption trade-off. Secondly, employing step-by-step verifiers increases the computational time needed by the system to produce an output. This trade-off must be analyzed on a case-by-case basis and compared to other alternatives, such as self-consistency. Different from self-consistency, our proposed approach aims to improve the correctness of each step in a step-by-step solution.

While our evaluation spanned multiple reasoning tasks and datasets, it was limited to tasks in the English language only. We leave the evaluation on more challenging datasets (e.g., MATH (Hendrycks et al., 2021)) or datasets with contradictory information (Chen et al., 2022; Kazemi et al., 2023) to future work.

During our human evaluation study, we observed low, yet significant, positive correlations. To validate the observed performance improvements on downstream tasks, we conducted additional experiments on synthetic data, which confirmed that these improvements are indeed expected.

Lastly, we only used Falcon-40B-Instruct in our experiments, as it was one of the largest and most capable open-source[8] model at the time of the experiments. We utilized GPT-4o and human correlations to provide a glimpse into how verifiers implemented with a more powerful underlying model might perform in comparison. We leave the exploration of other models to future work.

Ethics Statement

Our proposed approach utilizes large language models, which are known to be biased and to hallucinate. In this work, we do not pre-train nor fine-tune any large-scale models. Instead, we use already pre-trained open-source models and prompting.

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra-Aimée Cojocaru, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *ArXiv*, abs/2311.16867.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional ai: Harmlessness from ai feedback.

BIG bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, John A. Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuan-Fang Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712.

Hung-Ting Chen, Michael J.Q. Zhang, and Eunsol Choi. 2022. Rich knowledge sources bring complex knowl-

[8] Open source according to https://opensource.org/

edge conflicts: Recalibrating models to reflect conflicting evidence. In *Conference on Empirical Methods in Natural Language Processing*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *ArXiv*, abs/2304.05128.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168.

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.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation. *ArXiv*, abs/2212.04037.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2023. Think before you speak: Training language models with pause tokens. *ArXiv*, abs/2310.02226.

Kilem L. Gwet. 2014. Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *ArXiv*, abs/2305.14992.

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. *NeurIPS*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *ArXiv*, abs/2310.01798.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. Boardgameqa: A dataset for natural language reasoning with contradictory information. *ArXiv*, abs/2306.07934.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916.

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, Toronto, Canada. Association for Computational Linguistics.

Jieyi Long. 2023. Large language model guided tree-of-thought. *ArXiv*, abs/2305.08291.

E. A. M. and Richard Mckeon. 1941. The basic works of aristotle.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin T. Vechev. 2023. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *ArXiv*, abs/2305.15852.

Maxwell Nye, Anders Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *ArXiv*, abs/2112.00114.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Liangming Pan, Michael Stephen Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *ArXiv*, abs/2308.03188.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Ouyang Long, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *ArXiv*, abs/2206.05802.

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*, abs/2302.04761.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Huai hsin Chi, Nathanael Scharli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alon Talmor, Ori Yoran, Ronan Le Bras, Chandrasekhar Bhagavatula, Yoav Goldberg, Yejin Choi, and Jonathan Berant. 2021. Commonsenseqa 2.0: Exposing the limits of ai through gamification. *ArXiv*, abs/2201.05320.

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. *ArXiv*, abs/2302.13971.

Miles Turpin, Julian Michael, Ethan Perez, and Sam Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *ArXiv*, abs/2305.04388.

Gladys Tyen, Hassan Mansoor, Peter Chen, Tony Mak, and Victor Carbune. 2023. Llms cannot find reasoning errors, but can correct them! *ArXiv*, abs/2311.08516.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *ArXiv*, abs/2302.01560.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, MingSung Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning.

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*, abs/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*, abs/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? In *Annual Meeting of the Association for Computational Linguistics*.

Slavica Zec, Nicola Soriani, Rosanna Irene Comoretto, and Ileana Baldi. 2017. High agreement and high prevalence: The paradox of cohen's kappa. *The Open Nursing Journal*, 11:211 – 218.

A Frequently Asked Questions

A.1 What performance improvements are expected given certain correlation levels?

We argue that the low correlations are not problematic and that a correlation of 1.0 is (currently) unrealistic, as it would imply a verifier as good as humans. To further support this assertion and to add details on the level of improvements that can be expected from correlations like 0.1, we ran experiments on artificially generated data ans show the results in Table 10. This experiment further demonstrates that the improvements we have seen with correlation scores of 0.1 are normal. Intuitively, if the scores of the verifiers are not correlated with humans for a given question, the worst it can do is to select a random chain. However, if the verifiers are correlated with humans for a given reasoning chain, then they will select those reasoning chains that humans agree are better.

We supplement the correlation values of Falcon with that of GPT-4o, a more powerful model, to highlight how the proposed method scales with a stronger underlying model. Overall, GPT-4o obtains better correlation scores with human judgments.

A.2 The variance in Annotator's agreement scores

We hypothesize that the large variance in the inter-annotator agreement score comes from different reasoning styles between different humans. Nevertheless, the overall agreements presented in Table 6 highlight between Moderate Agreement and Substantial Agreement. We show all the inter-annotator agreements over each of the four attributes in Appendix H. We also elaborate on how Cohen's Kappa might be too harsh, a phenomenon known as Cohen's Kappa Paradox (Zec et al., 2017). For example, the Cohen's Kappa scores between the following two annotations $a_1 = [1, 1, 1, 1, 1, 1]$, $a_2 = [1, 1, 1, 1, 1, 0]$ is 0, even though they only disagree on one instance.

To this end, we included additional agreement scores: Gwet's AC1 (Gwet, 2014) and naive agreement. Both showed that the annotators agree more than initially revealed by Cohen's Kappa.

A.3 Costs of the proposed method

Our focus in this work has been to explore the extent to which LLMs are capable of error detection and error correction. Especially since there is a divergence in findings across prior work regarding the LLMs ability to detect and correct its errors (Tyen et al., 2023; Huang et al., 2023; Madaan et al., 2023, *inter alia*) even if it meant temporarily sacrificing efficiency. We leave the exploration of more efficient methods to future work.

A.4 Subjectivity of the verifiers

The relevance and logical consistency verifiers may appear subjective, evidenced by the varying responses from human annotators. However, our focus has not been on addressing nuanced scenarios where human judgment itself might differ. Instead, our intention has been to capture and rectify the more straightforward errors, those instances where even human annotators unanimously agree.

B Experimental Settings

B.1 Datasets

We include in Table 3 an input/output example for each dataset used. We also experimented with the Object Tracking problem from BigBench, but we removed it because for all experimental settings, encompassing both baseline and proposed methods, the performance consistently fell below the chance-level threshold. For Last Letter Concatenation, we use only 2 words instead of 4, as we empirically observed that Falcon-40B is not able to tackle the problem when there are four words. We use a temperature of 0.7 for all our experiments.

B.2 Hardware

We ran the experiments on machines with 8 A10G 24GB GPUs (AWS `g5.48xlarge`). In total, we used approximately 10 weeks worth of `g5.48xlarge` time.

C Verifiers

We include the prompts we used for each verifier.

C.1 Relevance Prompt

> You are a helpful assistant that is good at evaluating reasoning chains in order to solve logic problems.
> You are given a logic problem and a draft solution with numbered steps that we need to complete. Evaluate the draft solution by determining whether it adds relevant information that helps to solve the problem. If it is relevant answer by 'yes, the solution is relevant', otherwise

Reasoning Task	Dataset Name	Input	Expected Output
Other	BigBench Date Understanding	Yesterday was April 30, 2021. What is the date today in MM/DD/YYYY? Choices: (A) 05/01/2021, (B) 02/23/2021, (C) 03/11/2021, (D) 05/09/2021, (E) 06/12/2021, (F) 04/29/2021	(A)
Commonsense	CSQA 2.0	a pupil can be either a student or part of an eye	yes
	Strategy	Is it common to see frost during some college commencements?	yes
Symbolic	Coinflip	A coin is heads up. Whitney flips the coin. Erika does not flip the coin. Tj does not flip the coin. Benito flips the coin. Is the coin still heads up? Note that "flip" here means "reverse"	yes
	Last Letter (2)	Take the last letters of each words in "Whitney Benito" and concatenate them.	yo
Math	GSM8k	Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?	72
	SVAMP	Bryan took a look at his books as well. If he has 34 books distributed equally in 2 bookshelves. How many books are there in each bookshelf?	17
	AddSub	Joan found 70 seashells on the beach. she gave Sam some of her seashells. She has 27 seashell. How many seashells did she give to Sam ?	43

Table 3: An example of input and expected output for each of the datasets we experiment with.

say 'no' and explain which steps failed.
Problem: {problem statement}
Draft solution: {previous steps}
Draft step: {current step}
Your evaluation of the draft step:

C.2 Mathematical Accuracy Prompt

For brevity, we have included a single example from the set of 20 in-context examples. The in-context examples are a mix of inputs with mathematical calculations and without. Furthermore, within the set of examples involving mathematical calculations, both incorrect and correct calculations are included for comprehensive coverage.

Instruction:
The task is to extract the mathematical calculations appearing below and return the result in JSON format. Please do not perform any additional calculations and do not introduce any number or numerical expression that does not appear in the original input text. If there is no explicit calculation performed, do not return anything.

Input:
Therefore, he has $87-$32=«87-32=40»$40 left

Output:
“‘json
{[{"lhs": "87-32", "op": "=", "rhs": "40"}]}
“‘
<..> Input:
{input}

Output:
“‘json

C.3 Logical Consistency Prompt

"""You are a smart, critical, and logical teacher assistant. You are critically reading a student's answer line by line and verifying each line for any contradictions in the student's argument. More information below.

Previous Steps:
{previous steps}

Last Step:
{current step}

Instruction:
Given the information present in the Last Step and in the Previous Steps, please check if the conclusion present in the Last Step is contradicting any information from the Previous Steps.

Feedback:
Based on the Last Step, which is "{current step}", and on the Previous Steps, we can conclude that the Last Step is"""

D Self-Consistency

We include here the plots for all the datasets for the Self-Consistency experiment performed in Section 4.3

E Self-Consistency: Weighted Voting vs Majority Voting

We include in Figure 7 the results over all datasets for the (1) Weighted Voting and (2) Majority Voting over the same reasoning chains.

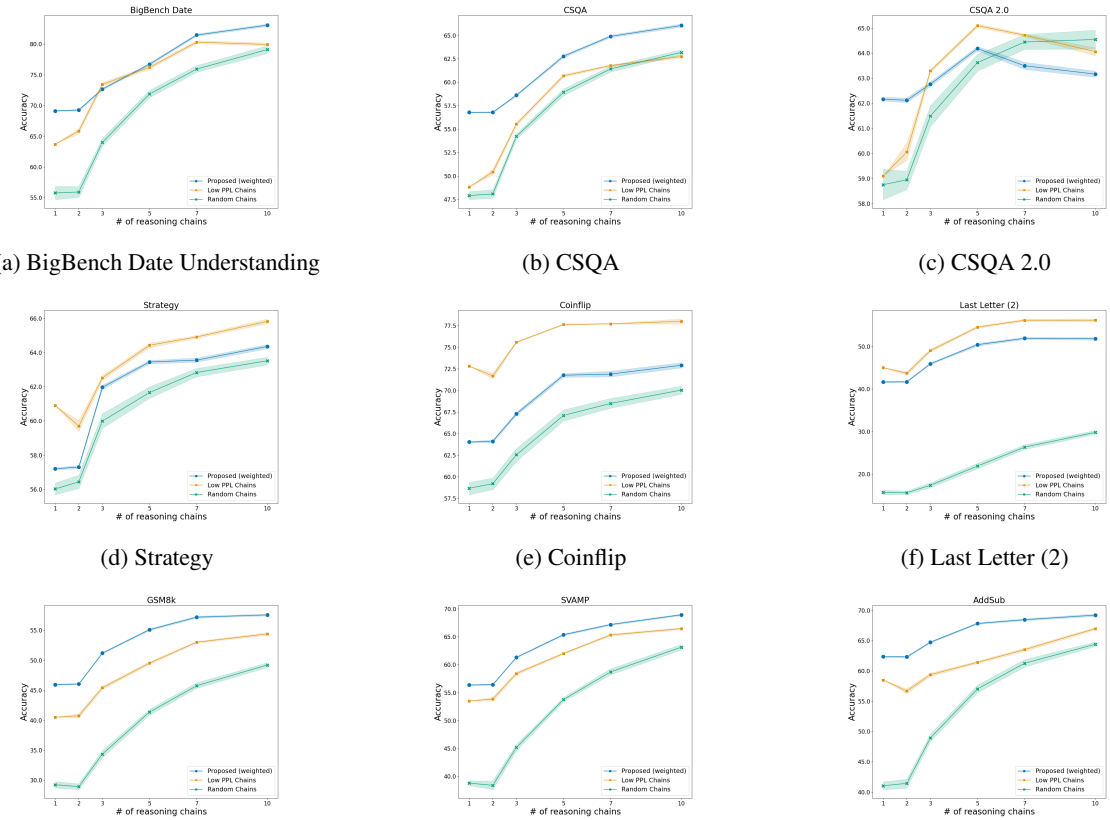
F Ablation

We include a more comprehensive ablation in Table 4, where we ablate over more combinations of the verifiers.

G Verifying Incomplete Reasoning Chains

To complement Figure 4, where we verified a *percentage* of the total number of reasoning steps for a given reasoning chain, we also include Figure 8, where we verify a fixed number of reasoning steps. The motivation behind this experiment is that the number of reasoning steps might not be known beforehand.

To supplement the analysis performed in Section 4.4, we include in Table 5 the performance obtained by our proposed method when verifying a varying number of steps, from 0 (no verification) to All (verify all reasoning steps).



(a) BigBench Date Understanding (b) CSQA (c) CSQA 2.0

(d) Strategy (e) Coinflip (f) Last Letter (2)

(g) GSM8k (h) SVAMP (i) AddSub

Figure 6: Self-consistency, sampling between 1-10 reasoning paths (↑)

H Human Evaluation

H.1 Inter-Annotator Agreement

We include the inter-annotator agreement across all 4 measured attributes in Figures 9, 10, 11, 12. We remark the large variance in the agreement, even for principles that are less subjective (e.g. *Mathematical Accuracy*)

We also include in Table 6 the overall agreement by attribute.

H.2 Other Agreement Scores

We further investigated the human annotation data and found many instances where the agreement score, as given by Cohen's Kappa, was too harsh. For example, the Cohen's Kappa scores between the following two annotations $a_1 = [1,1,1,1,1,1]$ and $a_2 = [1,1,1,1,1,1]$ is nan. Between $a_3 = [1,1,1,1,1,1]$, $a_4 = [1,1,1,1,1,0]$ is 0, even though they only disagree on one instance. Lastly, for $a_5 = [1,1,1,0,1,1,1,1,0,1]$ and $a_6 = [1,1,1,1,1,0,1,1,1,1]$ the Cohen's Kappa score is -0.154. We remark that in all three cases above, judging only from the annotations, the annotators tend to agree, but this is not reflected in the Cohen's Kappa score, a phenomenon called Cohen's Kappa Paradox (Zec et al., 2017). We note that Krippendorff's alpha only fixes the first example.

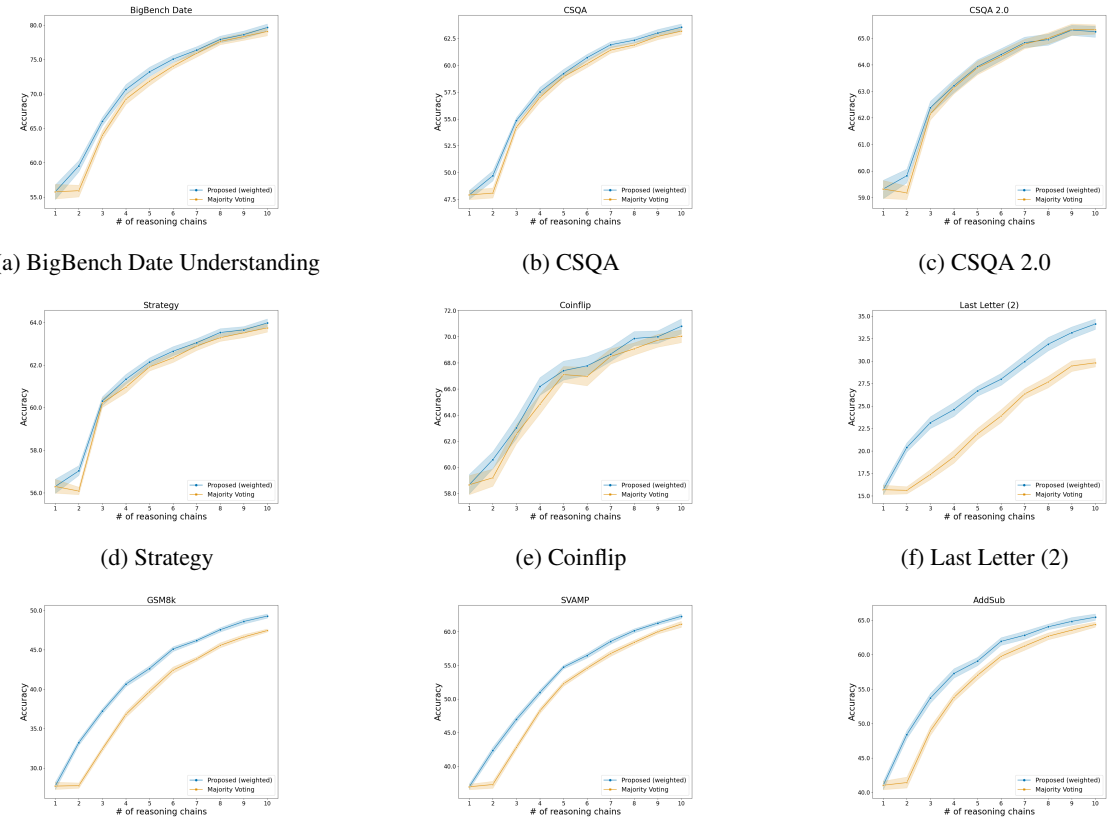
Therefore, we include in Table 7 the agreement score, as calculated using Gwet's AC1 (Gwet, 2014). We remark that the agreement scores are higher than initially revealed by Cohen's Kappa scores.

Lastly, we include in Table 8 the agreement computed as the percentage of time the annotators give the same label, without accounting for agreement by chance.

H.3 Correlations

We include the Pearson correlations between each of the 4 measured attributes (Relevance, Logical Consistency, Mathematical Accuracy, and Overall Correctness) in Figure 13. We remark that there is a large variance in the correlation scores.

We also include in Figure 14 the correlations between the verifiers and the human assessments. Additionally, we include the correlations between

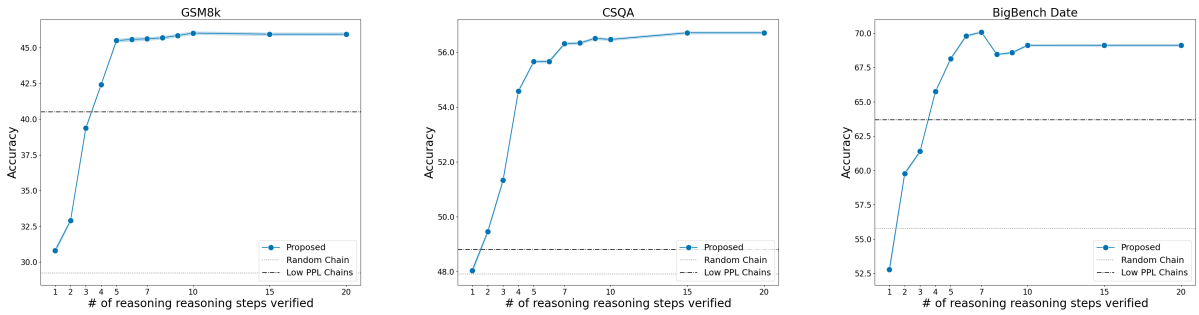


(a) BigBench Date Understanding (b) CSQA (c) CSQA 2.0

(d) Strategy (e) Coinflip (f) Last Letter (2)

(g) GSM8k (h) SVAMP (i) AddSub

Figure 7: Self-consistency on the *same* reasoning chains, comparing between weighting the final answer using the scores from our proposed verifiers or taking the majority voting (↑). Overall, using the scores of our proposed verifiers to perform weighted voting consistently improves the final performance.



(a) GSM8k (b) CSQA (c) BigBench Date Understanding

Figure 8: Verifying only the first X steps of a chain.

the human assessment of the overall correctness of a given reasoning step and the aggregated score, with and without perplexity.

H.4 Human Annotator Instructions

We provide an overview of the instructions given to the human annotators.

(1) Overall Correctness: If there is any reasoning issue with this step, answer n. If you cannot evaluate the step because you lack expertise or there are some other issues with the step, answer a. Please let us know about the reason in the notes column. If there is nothing wrong with the reasoning step, answer y.

(2) Mathematical Accuracy: Are all arithmetic calculations in this reasoning step correct? This question is strictly about arithmetic calculations and not about how the calculation is used to

Verifiers				Math		
Perplexity	Relevance	Math Accuracy	Consistency	AddSub	GSM8k	SVAMP
✗	✗	✗	✗	41.04±1.79	29.23±1.58	38.78±1.26
✗	✗	✗	✓	48.84±0.85	33.51±0.69	45.41±0.56
✗	✗	✓	✗	45.76±1.99	35.59±1.60	41.68±1.65
✗	✓	✗	✗	49.04±0.44	33.38±0.52	46.98±0.42
✗	✓	✓	✓	53.21±0.40	43.37±0.25	49.83±0.18
✓	✗	✗	✗	59.09±0.59	41.53±0.49	53.85±0.45
✓	✗	✓	✗	60.58±0.47	45.66±0.31	55.80±0.37
✓	✗	✓	✓	59.57±0.21	**48.02±0.24**	55.35±0.24
✓	✓	✗	✗	57.47±0.33	41.87±0.29	54.83±0.15
✓	✓	✓	✗	59.82±0.22	44.47±0.22	**56.44±0.24**
✓	✓	✓	✓	**62.34±0.25**	45.94±0.30	56.36±0.23

(a) Ablation Single Chain math results, Accuracy, Higher is better ↑

Verifiers			Other	Commonsense			Symbolic	
Perplexity	Relevance	Consistency	BigBench Date	CSQA 2.0	CSQA	Strategy	Coinflip	Last Letter (2)
✗	✗	✗	55.77±3.06	58.75±1.68	47.91±1.22	56.03±0.99	58.67±2.04	15.68±1.51
✗	✗	✓	62.70±0.68	56.77±0.55	53.00±0.59	54.31±0.70	49.17±0.76	20.63±0.78
✗	✓	✗	62.57±0.45	61.03±0.21	52.62±0.29	57.31±0.26	53.63±0.51	15.79±0.30
✗	✓	✓	63.50±0.26	60.47±0.12	54.74±0.25	55.98±0.24	52.68±0.37	16.50±0.24
✓	✗	✗	64.06±0.60	60.27±0.25	51.35±0.36	**60.03±0.30**	**73.36±0.42**	41.73±0.48
✓	✗	✓	64.72±0.31	58.74±0.12	52.99±0.24	58.54±0.13	64.89±0.38	**46.03±0.34**
✓	✓	✗	**71.04±0.22**	**62.21±0.19**	54.28±0.14	58.98±0.14	69.81±0.20	39.53±0.21
✓	✓	✓	69.12±0.21	62.16±0.22	**56.79±0.12**	57.21±0.17	64.02±0.21	41.64±0.44

(b) Ablation Single Chain non math results, Accuracy, Higher is better ↑

Table 4: Ablation over the types of verifiers used. Overall, all verifiers are meaningfully contributing towards the final solution.

progress toward the solution.

(3) Logical Consistency: Is this reasoning step logically consistent, in itself, and with previous steps? Answer y if the step is logically consistent within itself, and with all previous steps, including the prompt (problem statement). A step is logically consistent when it uses available information in a way that is logically correct. In most cases this means that the conclusions that are reached in this step follow logically from assumptions made. It can also mean that the step does not contradict information provided in previous steps (or the same step).

(4) Relevance: Does this reasoning step add information that is relevant for solving the problem? Information is relevant when it is useful for solving the problem (e.g. it states helping assumptions, or it reaches a conclusion that answers the problem or get you closer to an answer). Information can be added by re-stating information from the prompt, by reaching a conclusion, or by introducing completely new information – any of these can be relevant or irrelevant.

H.5 Ratios of Steps Annotated as Incorrect

We show in Table 9 the ratio of steps each annotator deemed as incorrect.

I Expected Performance Given Correlation Levels

We showed in Sections 4.2 and 4.3 that employing the proposed verifiers leads to performance improvements. Then, we analyzed in Section 4.6 the correlations between the proposed verifiers and human judgments, observing significantly positive (but low) correlations. In this section, we further analyze what improvements can we expect for a given level of correlations. To this end, we conducted additional experiments using artificially generated data, allowing us precise control over the correlation values between the verifiers and human judgments. By randomly sampling scores to simulate the verifiers' and human annotators' judgments, we manipulated the data to induce positive correlations

	Other	Commonsense			Symbolic		Math		
# of Steps Verified	BigBench Date	CSQA 2.0	CSQA	Strategy	Coinflip	Last Letter (2)	GSM8k	SVAMP	AddSub
0	55.77±3.06	58.75±1.68	47.91 ± 1.22	56.03±0.99	58.67±2.04	15.68±1.51	29.23±1.58	38.78±1.26	41.04±1.79
1	52.77±0.66	59.73±0.26	48.04 ± 0.26	**58.65±0.23**	62.03±0.61	40.22±0.64	30.80±0.53	38.12±0.59	46.29±0.71
2	59.77±0.35	61.05±0.21	49.45 ± 0.16	58.22±0.14	67.82±0.31	**51.25±0.39**	32.90±0.24	41.31±0.26	50.64±0.38
3	61.40±0.22	61.55±0.14	51.33 ± 0.17	58.04±0.14	**70.60±0.31**	46.89±0.26	39.38±0.29	46.27±0.22	51.14±0.25
4	65.75±0.18	**63.40±0.19**	54.58 ± 0.14	58.28±0.16	68.26±0.36	41.49±0.23	42.42±0.16	49.61±0.20	58.02±0.25
5	68.15±0.25	62.35±0.19	55.66 ± 0.10	56.96±0.17	66.72±0.24	33.13±0.35	45.49±0.26	53.38±0.23	60.67±0.29
All	**69.12±0.21**	62.16±0.22	**56.79 ± 0.12**	57.21±0.17	64.02±0.21	41.64±0.44	**45.94±0.30**	**56.36±0.23**	**62.34±0.25**

Table 5: Single Chain results, Accuracy, Higher is better ↑

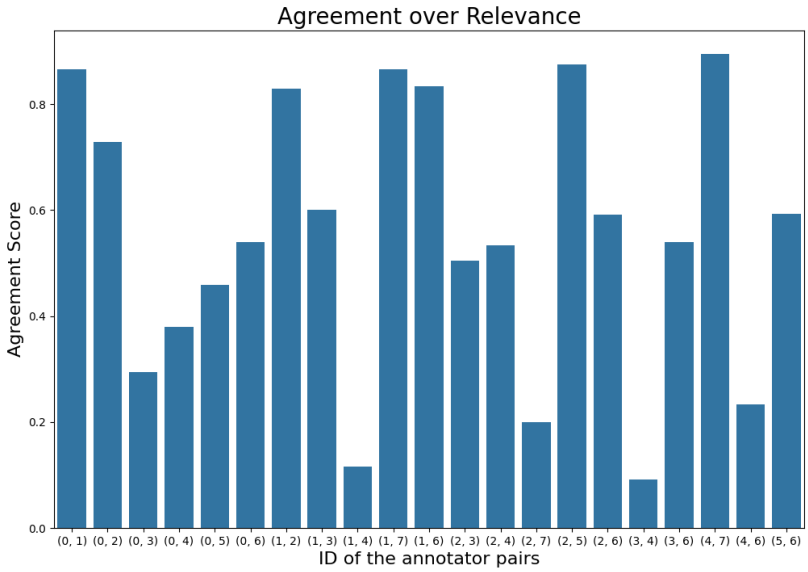


Figure 9: Annotator Agreement over the Relevance of a given reasoning step

Attribute	Overall Agreement
Relevance	0.55
Math Accuracy	0.81
Logical Consistency	0.53
Overall Correctness	0.66

Table 6: Overall Agreement by Attribute

Attribute	Overall Agreement
Relevance	0.71
Math Accuracy	0.84
Logical Consistency	0.75
Overall Correctness	0.81

Table 7: Overall Agreement by Attribute computed using Gwet's AC1

and recorded the resulting final scores.[9]

We summarize our results in Table 10. We remark that even for modest correlations, the performance increase is over 20% relative, in line with what we observed empirically with real data.

J Examples

We provide two qualitative examples in Figures 15, **??**, and 17, comparing the solutions chosen by the lowest perplexity method and by our proposed method, that of using the verifier scores.

[9]The correctness of reasoning chains in the artificially generated data is determined based on the sampled scores representing human judgments. A reasoning chain is considered correct if it consists of more than 75% correct reasoning steps.

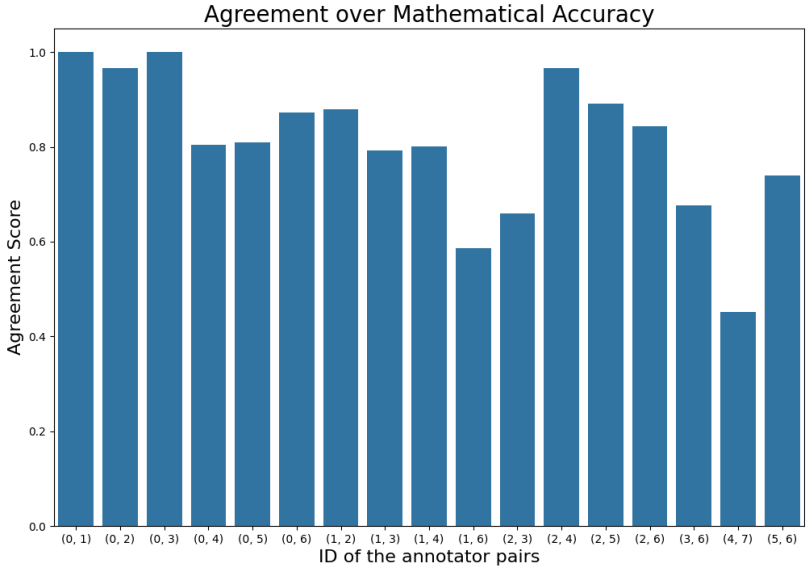


Figure 10: Annotator Agreement over the Mathematical Accuracy of a given reasoning step

Attribute	Overall Agreement
Relevance	0.83
Math Accuracy	0.89
Logical Consistency	0.86
Overall Correctness	0.89

Table 8: Overall Agreement by Attribute computed using the Naive approach

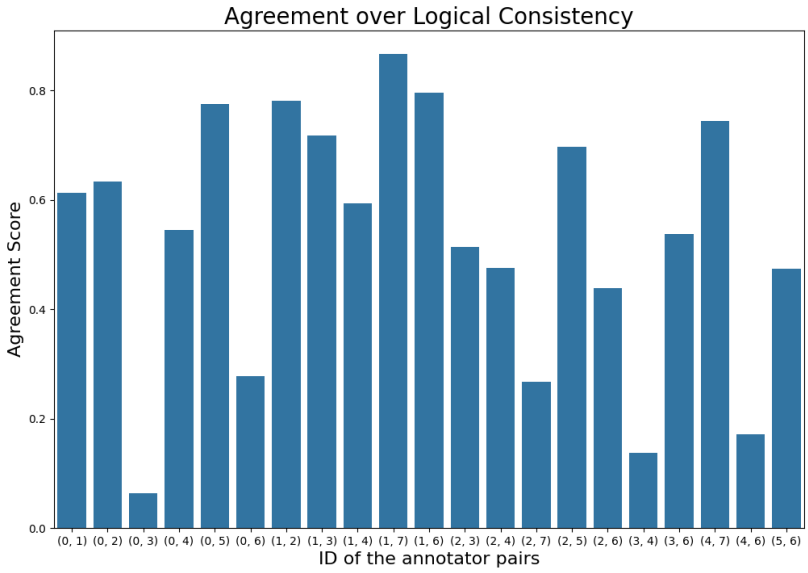


Figure 11: Annotator Agreement over the Logical Consistency of a given reasoning step

Metric\Annotator	A0	A1	A2	A3	A4	A5	A6	A7
Ratio of steps annotated as Irrelevant	0.41	0.37	0.24	0.33	0.17	0.22	0.36	0.35
Ratio of steps annotated as Mathematically incorrect	0.06	0.05	0.03	0.03	0.07	0.06	0.11	0.03
Ratio of steps annotated as Logically Inconsistent	0.4	0.35	0.25	0.31	0.28	0.23	0.28	0.34
Ratio of steps annotated as Overall Incorrect	0.43	0.38	0.31	0.37	0.31	0.24	0.34	0.36

Table 9: The ratio of steps each annotator annotated as Irrelevant, Mathematically incorrect, Logically Inconsistent, and Overall Incorrect.

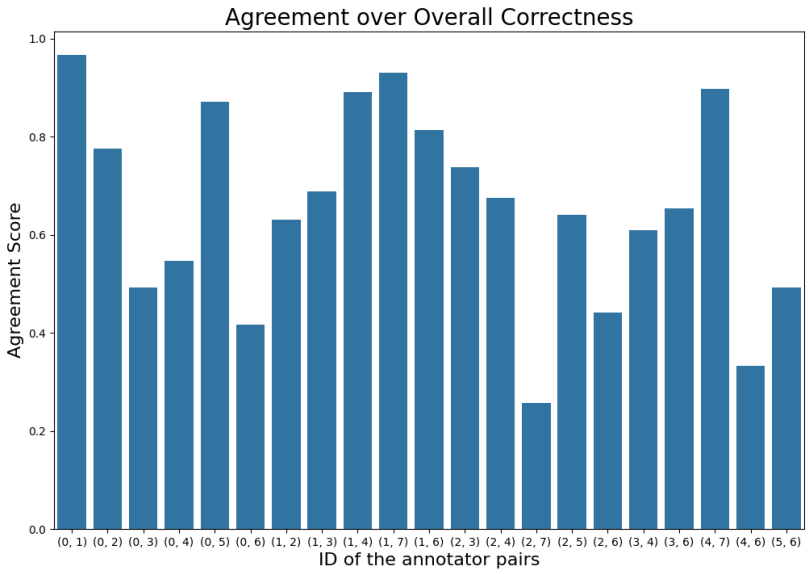
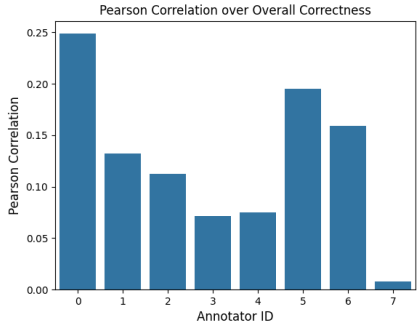
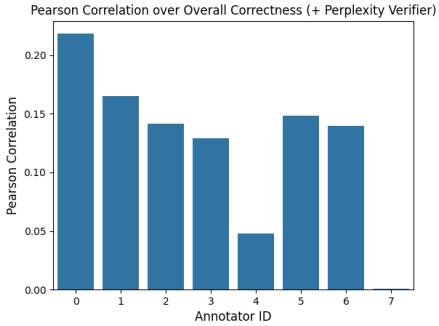


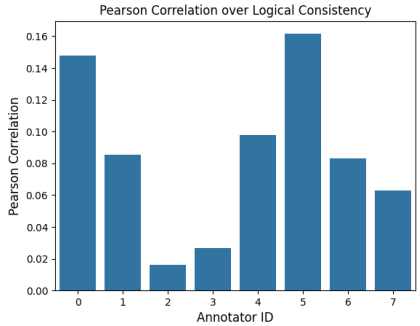
Figure 12: Annotator Agreement over the Overall Correctness of a given reasoning step



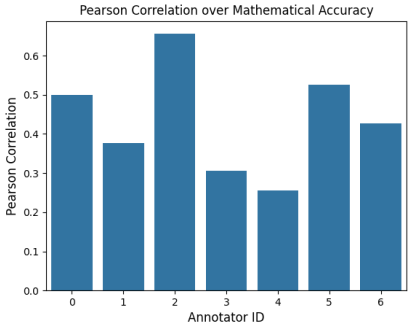
(a) Correlation score between our overall score (excluding the perplexity verifier) and the annotators assessment.



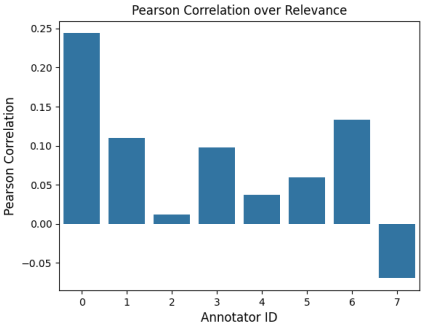
(b) Correlation score between our overall score (including the perplexity verifier) and the annotators assessment.



(c) Correlation score between our Logical Consistency Verifier and the annotators assessment.



(d) Correlation score between our Mathematical Accuracy Verifier and the annotators assessment.



(e) Correlation score between our Relevance Verifier and the annotators assessment.

Figure 13: Correlations between the scores of the verifiers and the human annotators' assessment. Additionally, we consider two scores for the *Overall Assessment* attribute. One was computed only with the three verifiers and the second one was computed with the three verifiers, together with the perplexity verifier.

Method	Score
Model with no verifier	0.19
Model with verifier with $correlation = 0.075$	0.22
Model with verifier with $correlation = 0.1$	0.23
Model with verifier with $correlation = 0.15$	0.26
Model with verifier with $correlation = 0.25$	0.31
Model with verifier with $correlation = 0.5$	0.48
Model with verifier with $correlation = 0.75$	0.72
Model with verifier with $correlation = 1.0$	0.98

Table 10: Scores of the proposed method given various level of correlations. Even for modest correlations, the performance increase is over 20% relative.

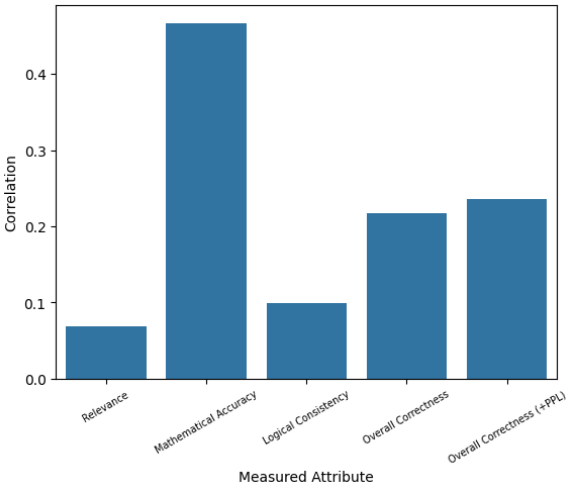


Figure 14: Correlation between the scores of our proposed verifiers and the human assessments. Additionally, we include the correlations between the human assessment of the overall correctness of a given reasoning step and the aggregated score (with and without perplexity).

Problem Statement
Antoine's french onion soup recipe calls for 2 pounds of onions. He likes to double that amount. His soup serves 6 people. The onions are currently on sale for $2.00 a pound. He also needs 2 boxes of beef stock, that are also on sale for $2.00 a box. What is the cost per serving? (Round to the nearest integer.)

Gold Answer
$2

Lowest perplexity solution
- The recipe calls for 2 pounds of onions, so that's $4.00 for the onions.
- Antoine doubles the amount of onions, so he uses 4 pounds for his recipe.
- That means he needs to buy 2 boxes of beef stock, which are on sale for $2.00 each.
- The total cost for onions and beef stock is $8.00.
- Therefore, the cost per serving is $1.33 (rounded to the nearest integer).

Solution according to verifiers
- Antoine likes to double the amount of onions, so he uses 4 pounds of onions.
- Each onion costs $2.00, so 4 pounds of onions cost 4 * $2.00 = $«4*2=8»8.00.
- He also needs 2 boxes of beef stock, which cost 2 * $2.00 = $«2*2=4»4.00.
- Therefore, the total cost for all ingredients is $8.00 + $4.00 = $«8+4=12»12.00.
- Since the soup serves 6 people, each person will eat $12.00 / 6 = $«12/6=2»2 worth of soup.
- Therefore, the cost per serving is $2.00.

Figure 15: Qualitative example of the types of solutions preferred by our proposed method.

Problem Statement
Pam and Fred went to a carnival. Pam rode the roller coaster 2 times while Fred rode it 4 times. After that, each of them decided to ride the luge 2 times. If each ride cost 6 tickets, how many tickets did they use that day?

Gold Answer
60

Lowest perplexity solution
- Pam rode the roller coaster 2 times, so that's 2 * 6 = 12 tickets.
- Fred rode the roller coaster 4 times, so that's 4 * 6 = 24 tickets.
- After that, they each rode the luge 2 times, so that's 2 * 2 * 6 = 24 tickets.
- Finally, Pam rode the roller coaster 2 times, so that's 2 * 6 = 12 tickets.
- Fred rode the roller coaster 4 times, so that's 4 * 6 = 24 tickets.
- After that, they each rode the luge 2 times, so that's 2 * 2 * 6 = 24 tickets.
- Finally, Pam rode the roller coaster 2 times,

Solution according to verifiers
- Pam rode the roller coaster 2 times, which means she used 2 * 6 = 12 tickets.
- Fred rode the roller coaster 4 times, which means he used 4 * 6 = 24 tickets.
- After riding the roller coaster, Pam and Fred rode the luge 2 times each, which means they used 2 * 2 * 6 = 24 tickets in total.
- So, they used a total of 12 + 24 + 24 = 60 tickets that day.

Figure 16: Qualitative example of the types of solutions preferred by our proposed method.

Problem Statement
Eliza's rate per hour for the first 40 hours she works each week is $10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week?

Gold Answer
$460

Lowest perplexity solution
- Eliza works for 40 hours at a rate of $10 per hour, so her earnings for the first 40 hours are $400.
- Since she worked 5 more hours than the 40 hours at her regular rate, she will receive 1.2 times her regular hourly rate for those 5 hours. (1.2 times 10 is 12)
- Therefore, Eliza's earnings for the 45 hours she worked this week are $420. ($400 + $20)
- So, Eliza's earnings for this week are $420.

Solution according to verifiers
- Eliza's regular hourly rate for the first 40 hours is $10.
- Her overtime rate is 1.2 times the regular rate, which is $12.
- She worked for 45 hours this week.
- So, for the first 40 hours, her earnings are $400 ($10/hour x 40 hours), and for the 5 overtime hours, her earnings are $60 ($12/hour x 5 hours).
- Adding these two amounts, Eliza's total earnings for this week are $460 ($400 + $60).

Figure 17: Qualitative example of the types of solutions preferred by our proposed method.