Paraphrase and Solve: Exploring and Exploiting the Impact of Surface Form on Mathematical Reasoning in Large Language Models

Anonymous ACL submission

Abstract

This paper delves into the relationship between the surface form of a mathematical problem and its solvability by large-scale language models. We found that subtle alterations in the surface form can significantly impact the answer distribution and the solve rate, exposing the language model's lack of robustness and sensitivity to the surface form in reasoning through complex problems. To improve the mathematical reasoning performance, we propose Self-Consistency-over-Paraphrases (SCoP), which diversifies reasoning paths from specific surface forms of the problem. We evaluate our approach on four mathematics reasoning benchmarks over three large language models and show that SCoP improves mathematical reasoning performance over vanilla self-consistency, particularly for problems initially deemed unsolvable. Finally, we provide additional experiments and discussion regarding problem difficulty and surface forms, including cross-model difficulty agreement and paraphrasing transferability, Variance of Variations (VOV) for language model evaluation, the data difficulty map, and more.

1 Introduction

While large-scale language models (LLMs) have taken the NLP landscape by storm, outperforming the state-of-the-art in various tasks, their ability to reason through complex problems such as mathematics remains a bottleneck (Rae et al., 2022; Srivastava et al., 2023; Liang et al., 2023). The performance of language models in solving mathematical problems is sometimes paradoxical and distanced from human intelligence: they can solve problems that are challenging for humans but can also struggle with seemingly simple ones. This raises a question: what factors contribute to the difficulty of a math problem for an LLM?

Specifically, the information in a math problem can be divided into two types. The first is the *semantics information*, which involves the content, complexity, and knowledge involved in the math problem. The second is the *surface forms*, *i.e.*, how the questions, assumptions, and constraints are described in the math problem. Intuitively, one would believe that the semantics information is the primary determining factor of the difficulty of the math problem, and that the surface form should only have a marginal impact at best, because as long as it is clear, the way the problem is described would not change the actual solution. Would this be the case for LLMs?

In this paper, we delve into the relationship between the problem's surface form and its solvability with respect to the language model. Specifically, we follow the self-consistency setting (Wang et al., 2022) to sample multiple answers to the same math problem and compute *solve rate* as the percentage of correct answers. Our primary finding is that, counter-intuitively, subtle alterations in the surface form of a math problem can significantly impact the answer distribution and solve rate. Consider an example in Figure 1, where the left and right panels contain an identical math problem described in two different ways. However, from left to right, the solve rate increases from 5% to 100%, with all reasoning paths leading to the correct answer - what initially appears to be a difficult problem to the language model unreasonably transforms into an effortlessly solvable one. This phenomenon exposes the language model's lack of robustness and sensitivity to the surface form in reasoning through complex problems.

Motivated by this finding, we explore improving the mathematical reasoning performance of the language model by diversifying reasoning paths from specific surface forms of the problem. We leverage the language model's paraphrasing ability to generate surface forms with identical semantics[1]

[1]Rigorously, the surface forms can be regarded as "quasi-paraphrases that convey approximately the same meaning using different words" (Bhagat and Hovy, 2013).

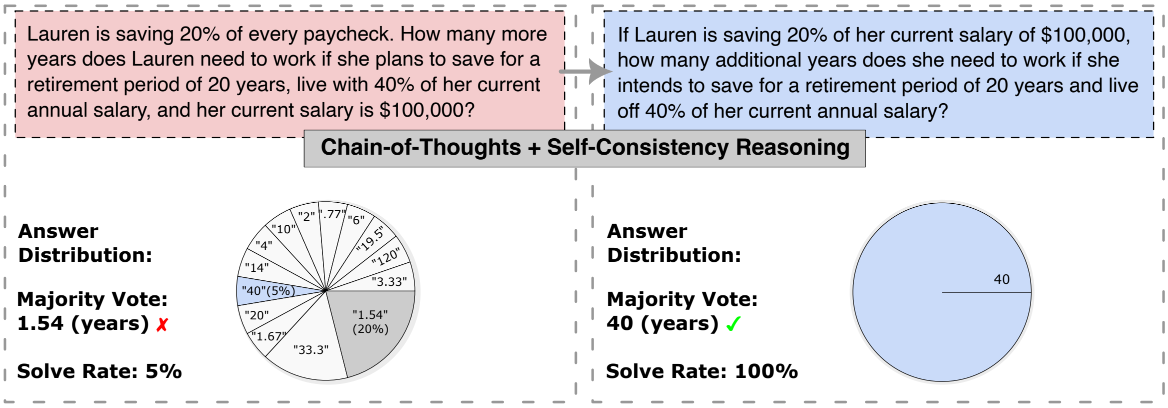


Figure 1: A comparison of the answer distribution and solve rate between surface form variations of a math word problem from GSM8K, when prompted to GPT-3.5-turbo using Self-Consistency, with 40 sampled reasoning paths. Solve rate can vary dramatically between surface forms with equivalent semantics.

and propose **Self-Consistency-over-Paraphrases (SCoP)**, which consists of two steps: ❶ For each math problem, generate K paraphrase using an LLM; and ❷ Ask the LLM to generate N/K reasoning paths for each paraphrase, and then select the most consistent answer among the N answers. The intuition is that if a problem exhibits a low solve rate and ineffective reasoning paths due to its original surface form, introducing diversity in its surface forms can be beneficial. We also introduced in-context exemplars to the language model when paraphrasing, which are the paraphrases that obtain a solve rate improvement over their original problem, aiming to generate surface forms with the same semantics yet a higher solve rate through language models in context learning abilities (Min et al., 2022; Brown et al., 2020).

We evaluate our approach on four mathematics reasoning benchmarks: GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), MATH (Hendrycks et al., 2021), and MMLU-Math (Hendrycks et al., 2020), over three large language models: LLaMA-2-70b (Touvron et al., 2023), GPT-3.5-turbo and GPT-4 (OpenAI, 2023). Our experiments show that SCoP improves mathematical reasoning performance over the traditional Self-Consistency method, particularly for problems initially deemed unsolvable. In additional experiments, we show that the difficulty ranks across language models are positively correlated, with higher agreement within the GPT model family and simpler datasets. The rank alignment of difficulty may influence the transferability of the paraphrases. Moreover, we propose Variance of Variations (VOV), a metric for evaluating language model robustness against surface form variations. Finally, we explain why SCoP could work by defining a data difficulty map based on the entropy of answer distribution and the solve rate and bring discussions with qualitative examples.

2 Problem Difficulty and Surface Forms

In this section, we present our pilot study of the impact of surface form on LLMs' ability to solve the problem. In all our studies, we follow the self-consistency setting (Wang et al., 2022), which extends over chain-of-thought (Wei et al., 2022) by using sampling decoding to generate a variety of reasoning paths. From this setting, we quantify the *difficulty* of a problem *w.r.t* a language model by its **solve rate**, which is the proportion of the reasoning paths that lead to the correct answer. When the solve rate exceeds 50%, a majority vote guarantees the correct answer. Note the solve rate measures the difficulty of a single problem input and is also a model-dependent metric.

To study how surface form impacts the solve rate, we use the math word problem from the GSM8K dataset (Cobbe et al., 2021). For each math problem, we generate a paraphrase using GPT-3.5-turbo[2] (detailed instructions are shown in Appendix C). We then compare the solve rates of the original problem and the paraphrase solved by GPT-3.5-turbo using self-consistency with $N = 40$ and a temperature of 0.7.

Our finding is that the solve rate varies significantly across the surface forms. Figure 1 shows an example with the original problem on the left and the paraphrased one on the right. In the original

[2]https://platform.openai.com/docs/models/gpt-3-5

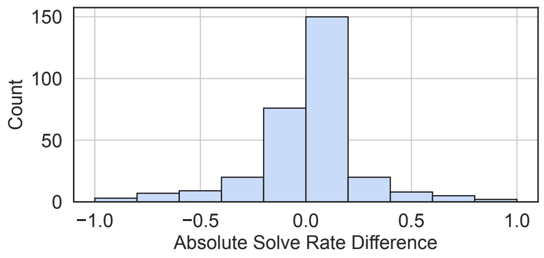


Figure 2: GSM8K - solve rate difference - from original to one of the random naive paraphrase.

problem, the reasoning paths result in a disarrayed answer distribution, with merely 5% achieving the correct answer "40" and the aggregated answer "1.54" (20%). In contrast, the solve rate of the paraphrase problem is 100%. We have identified many more such examples with drastic improvement in solve rate, presented in Table 6.

We further calculate the histogram of the solve rate changes in the paraphrased problem compared to the original one, shown in Figure 2. As can be observed, the distribution is heavy-tail, with 11.7% of the paraphrases resulting in over 25% absolute improvement in solve rate and with 13% resulting in over 25% absolute deterioration.

This phenomenon exposes the language model's deficiency in robustness and sensitivity to a comprehensive problem's surface form. It suggests that the challenge of some problems may not be due to the model's limitations, but rather the ineffective generation of reasoning paths from certain surface forms. Therefore, can we take advantage of this phenomenon to improve language model reasoning through surface form modifications, mirroring the way paraphrasing aids a student's cognitive and problem-solving processes (Swanson et al., 2019)?

3 Self-Consistency over Paraphrases

Motivated by the findings in Section 2, we propose a framework, called **Self-Consistency-over-Paraphrases (SCoP)**, which leverages the LLMs to generate paraphrases of math problems to improve their ability in solving them.

3.1 Framework Overview

As shown in Figure 3, SCoP consists of two steps.

- **Step 1: Paraphrase.** Prompt the LLM to generate K paraphrases of the original problem. For notational ease, denote p as the original problem, and $\bigcup_{k=1}^{K}\{q_k\}$ as the K paraphrases.
- **Step 2: Solve.** For each paraphrase, we ask the LLM to generate N/K reasoning paths, and thus the total number of generated answers is N. We then select the most consistent answer across the N reasoning paths as the final answer.

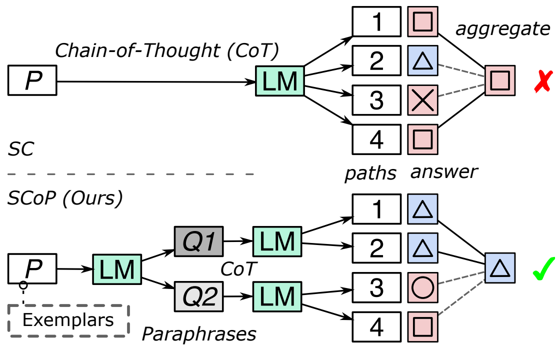


Figure 3: A comparison between Self-Consistency and our SCoP. SCoP splits N reasoning paths over K in-context learned paraphrases, instead of devoting all N reasoning paths to the single original problem P. The final answer is selected by aggregating all reasoning paths from these paraphrases with a majority vote.

The intuition behind SCoP is that if a problem exhibits a low solve rate and ineffective reasoning paths due to its original surface form, introducing diversity in its surface forms would be beneficial.

There are two important notes regarding SCoP. First, when we increase K, the total number of reasoning paths N is held fixed, which separates the effect of increasing the diversity of reasoning paths from increasing the number of reasoning paths. This also ensures a fair comparison with other self-consistency baselines.

Second, there are two procedures in SCoP that involve an LLM, one to generate paraphrases (Step 1) and one to generate answers (Step 2). We use the *same* LLM to perform both tasks. In this way, we can ensure that any performance improvement of SCoP is due to the diversity of paraphrasing itself, rather than cross-sharing of knowledge across different LLMs. In addition, there is no human annotation, training, fine-tuning, or auxiliary models involved in our SCoP framework.

3.2 Paraphrase Generation

The paraphrase generation in Step 1 is crucial to the success of SCoP. In this work, we explore two paraphrase generation methods.

Naïve. The naïve approach instructs the language model to generate K paraphrases of the math problem. However, this could generate many paraphrases with *worse* solve rate, because the solve

Algorithm 1 Paraphrase Exemplar Search

```
Input: Training data D^tr, N_shot, margin δ. Init. Candidates list C.
for step t in {1, 2, ..., T} do
    if Length(C) = N_shot then
        break
    Sample a problem p from D^tr without replacement.
    Compute solve rate SR(p)
    Obtain K Paraphrases {q_1, ..., q_K} of p.
    for k = 1 to K do
        Compute solve rate SR(q_k)
        if SR(q_k) >= SR(p) + δ then
            Add {p, q_k} to Candidates list C.
            break
```

rate change has high variability in both directions (as shown in Figure 2).

In-Context Learning. To increase the chance of generating 'good' paraphrases, we propose an in-context learning approach[3], where we obtain N_{shot} 'good' paraphrases as the in-context exemplars (marked as `[Exemplars]` in Figure 3). The 'good' paraphrases are formally defined as paraphrases that contribute to a solve rate improvement (by a preset margin δ) over the original problem. To obtain the 'good' paraphrases, we first generate some candidate paraphrases using the aforementioned naïve approach on a small number of math problems with labeled answers. We then compute the solve rate of the original problem and the paraphrases and select those whose improvement is over the margin δ. The detailed algorithm is presented in Algorithm 1.

4 Experiments

In this section, we will describe our experiment results evaluating the effectiveness of SCoP, as well as additional studies on how SCoP works.

4.1 Experimental Settings

Datasets We evaluate our approach on the following public mathematics reasoning benchmarks:
• **GSM8K** (Cobbe et al., 2021) contains 8.5K linguistically diverse grade school-level math questions with moderate difficulties.
• **AQuA** (Ling et al., 2017) consists of 100K algebraic word problems, including the questions, the possible multiple-choice options, and natural language answer rationales from GMAT and GRE.
• **MATH** (Hendrycks et al., 2021) is a competition mathematics dataset containing 12,500 problems with challenging concepts such as Calculus, Linear Algebra, Statistics, and Number Theory.
• **MMLU** (Hendrycks et al., 2020) is a comprehensive dataset containing various subjects. We specifically utilized the mathematics section of the dataset, which comprises college and high-school-level mathematics, statistics, and abstract algebra.

Language Models We utilize three popular LLMs trained with RLHF (Ouyang et al., 2022): LLaMA-2 (70B) (Touvron et al., 2023), an open-source LLM by Meta AI, GPT-3.5-turbo (version 0613), and GPT-4 (OpenAI, 2023), accessed via the OpenAI API. All experiments are conducted in zero-shot or few-shot settings, without training or fine-tuning the language models. We choose the temperature $T = 0.7$ and Top-p $= 1.0$ for sampling decoding for all three language models. The total number of reasoning paths N we sample for each problem is 40, following Wang et al. (2022).

Implementation Details For paraphrase generation (Step 1), we evaluate the two aforementioned schemes ❶ **Naïve**: We use the template " *Paraphrase the following math problem:* `{question}`" to prompt the language model to paraphrase the original problem; ❷ **In-Context Learning (ICL$_{\text{para}}$)**: We randomly select a set of 8 paraphrase exemplars by Algorithm 1 with margin[4] $\delta = 0.3$. The details of the prompt templates are available in Appendix C.

For answer generation (Step 2), we also implement two schemes: ❶ **Zero-Shot Chain-of-Thought (CoT)** (Kojima et al., 2023), which appends "*Let's think step by step.*" to the question text; and ❷ **Four-Shot CoT**, where we append four-shot in-context examples with CoT to the LLM when *solving* the math problems. Note that the in-context examples for answer generation are different in functionality and format from the ones for ICL$_{\text{para}}$.

4.2 Main Results

Zero-Shot CoT Table 1 illustrates the performance of SCoP under the zero-shot CoT setting, compared with the vanilla self-consistency (SC), using LLaMA-2-70b and GPT-3.5-turbo. We vary the number of paraphrases K across {1, 2, 4, 8} while keeping the total number of reasoning paths

[3]An alternative can be automatic prompt engineering, see Appendix B.

[4]We performed an ablation study of the margin effect on a separate development sets and found that using an extremely large margin can damage performance. See Appendix A.

		GPT-3.5-Turbo				LLaMA-2-70b			
		GSM8K	AQuA	MATH	MMLU	GSM8K	AQuA	MATH	MMLU
	HPR (%)	31.33	42.52	68	64	52	76.3	98.2	82.4
SC		76.33 (24.47)	66.93 (22.22)	59.0 (39.71)	52.8 (26.25)	58.67 (20.51)	40.47 (21.95)	10.53 (8.93)	32.8 (17.37)
SCoP (Naïve)	$k = 1$	72.23 (27.66)	63.39 (28.89)	55.0 (37.50)	48.4 (27.5)	51.0 (28.21)	38.14 (26.83)	24.56 (23.21)	27.2 (20.21)
	$k = 2$	76.0 (34.04)	65.75 (28.89)	56.5 (39.71)	52.8 (32.5)	54.33 (26.92)	39.53 (25.0)	29.82 (28.57)	29.6 (22.28)
	$k = 4$	77.67 (36.17)	67.32 (29.81)	57.5 (38.97)	56.0 (36.25)	55.67 (32.05)	41.4 (25.61)	**31.58 (30.36)**	32.02 (24.87)
	$k = 8$	79.33 (39.36)	68.11 (33.52)	59.5 (43.38)	55.6 (33.75)	60.33 (33.33)	41.4 (25.61)	28.07 (26.79)	35.6 (27.98)
SCoP (ICL_{para})	$k = 1$	77.85 (38.98)	66.43 (29.81)	54.0 (36.76)	52.5 (32.56)	58.67 (39.92)	42.91 (29.84)	24.68 (22.50)	34.6 (23.74)
	$k = 2$	**80.51** (39.24)	68.50 (31.67)	57.5 (39.13)	55.5 (34.11)	59.33 (36.31)	43.70 (30.37)	23.45 (21.24)	37.6 (26.26)
	$k = 4$	79.18 (38.27)	**70.47 (35.37)**	58.0 (41.18)	**58.0 (39.53)**	61.67 (40.48)	44.49 (30.37)	24.07 (21.87)	**37.8 (26.52)**
	$k = 8$	80.18 **(40.62)**	69.69 (34.44)	**60.0 (44.12)**	56.5 (34.88)	**63.33 (40.48)**	**46.46 (31.94)**	26.52 (24.39)	37.6 (25.76)

Table 1: Accuracy of SCoP distributing N/K reasoning paths over K in {1, 2, 4, 8} paraphrases in Naïve and ICL_{para} settings, against Self-Consistency (SC). Hard Problem Ratio (HPR%) represents the percentage of problems with an original solve rate ≤ 0.5 by Self-Consistency (SC). Accuracy is reported for both Hard Problems (HPR% ≤ 0.5) (inside parentheses) and global accuracy across the entire dataset (outside parentheses).

Model		GSM8K	AQuA	MATH	MMLU
LLaMA-2-70b	HPR (%)	56	75.2	95.2	81.6
	Self-Consistency	61.11 (30.0)	44.09 (25.65)	13.41 (9.17)	34.40 (19.61)
	SCoP (ICL_{para}, $k = 8$)	**65.08 (38.57)**	**48.82 (33.51)**	**23.62 (20.18)**	**36.40 (26.96)**
GPT-3.5-Turbo	HPR (%)	22	36	75	62
	Self-Consistency	80.0 (9.09)	70.0 (16.67)	51.6 (29.78)	54.4 (26.92)
	SCoP (ICL_{para}, $k = 8$)	**82.0 (36.36)**	**74.0 (27.78)**	**57.6 (38.22)**	**58.4 (36.54)**
GPT-4	HPR (%)	4	18	58	38
	Self-Consistency	98.0 (50.0)	84.0 (11.11)	64 (37.93)	74.0 (31.58)
	SCoP (ICL_{para}, $k = 8$)	98.0 (50.0)	**86.0 (33.33)**	**66 (41.38)**	**78.0(57.89)**

Table 2: A comparison of the performance (accuracy) between SC and SCoP (ICL_{para} paraphrasing, with $k = 8$) using 4-shot in-context chain-of-thought exemplars over three language models. Accuracy is reported for both Hard Problems (HPR% ≤ 0.5) (inside parentheses) and global accuracy across the entire dataset (outside parentheses).

fixed as 40. Due to resource constraints, we sampled 300 data points from each test set, except for AQuA, which contains 254 testing examples.

The performance metric is the accuracy of the self-consistency answer. We also report the accuracy over hard problems, defined as the problems whose original accuracy is below 50%. The accuracies over all problems and hard problems are reported inside and outside parentheses respectively. HPR% (Hard Problem Ratio) denotes the percentage of such hard problems.

There are three general observations. First, SCoP with the two paraphrasing schemes both outperform the vanilla self-consistency baseline. Surprisingly, even the naïve paraphrasing can lead to performance improvement, despite the high chances of generating paraphrases with worse solve rate (see Figure 2). We will discuss a hypothesis in Section 5. Between the two schemes, ICL_{para} consistently outperforms Naïve. Second, the performance improvement generally increases as K increases. Third, more significant performance gain over LLaMA-2-70B.

The results further indicate that MATH and MMLU are considerably more challenging than GSM8K and AQuA, as evidenced by their high HPR% and low overall accuracy. Moreover, significant accuracy gains are from the original "Hard Problems", suggesting that changing surface forms can solve the problems initially deemed unsolvable by self-consistency. Finally, when solving the MATH dataset with LLaMA-2-70b, ICL_{para} underperforms Naïve paraphrasing. We hypothesize that the MATH problems present a significant challenge for LLaMA-2-70b, making it difficult to effectively learn paraphrasing from in-context examples.

Four-Shot CoT One caveat of the zero-shot CoT results is that SCoP (ICL_{para}) has indirect access to additional ground-truth information from in-context exemplars. There is also a question of whether the advantage of SCoP over SC will diminish as both are exposed to more examples. To ensure a fair comparison and further validate the effectiveness of SCoP, Table 2 shows results under the four-shot CoT setting, where the baselines also have access to some ground-truth answer information. Due to resource constraints, we evaluate GPT4 with 100 random samples from each dataset. The results show that while four-shot CoT can improve SC and SCoP in general (compared

	GPT3.5, GPT4	GPT3.5, LLaMA-2	GPT4, LLaMA-2
GSM8K	0.573**	0.649***	0.445*
AQUA	0.543***	0.227***	0.314*
MATH	0.554***	0.242*	0.433*
MMLU	0.313*	0.320***	0.233

Table 3: Spearman's rank correlation of original problems' solve rate across language models.

Solver	Paraphraser	MMLU	GSM8K
GPT-3.5	Self	50.0	78.0
GPT-3.5	GPT-4	**54.0**	**84.0**
LLaMA-2	Self	**37.0**	61.0
LLaMA-2	GPT-4	34.0	**69.0**

Table 4: Performance of SCoP (Naïve, $k = 8$) on MMLU and GSM8K, with different paraphrasers.

		GSM8K	AQuA	MATH	MMLU
LLaMA2	Naïve	20.29	17.48	12.89	17.54
	ICL_{para}	18.88	15.7	12.20	16.58
GPT-3.5	Naïve	20.61	16.13	15.75	16.86
	ICL_{para}	16.18	10.74	15.59	15.57
GPT-4	ICL_{para}	9.69	11.46	16.96	21.26

Table 5: VOV values across datasets and language models, shown as standard deviation.

with zero-shot CoT), SCoP still consistently outperforms SC over all three language models. The only exception is GPT4 on GSM8K, which already achieves near-perfect performance with SC, thus SCoP only achieves equivalent performance.

4.3 Additional Studies

Searching for Exemplars Since our in-context learning paraphrasing scheme requires access to ground-truth answers, we would like to study how many problems with ground-truth answers are needed. Figure 4 illustrates how many data points in the training set, on average, need to be sampled to obtain N_{shot} 'good' paraphrases (x-axis) with different margins. We can observe that, although satisfying a large margin requires more samples, it is relatively easy (typically every ± 5 example) to find a sample that substantially improves solve rate after paraphrasing. This, again, indicates the sensitivity of the language model to surface form variations in mathematical reasoning.

Difficulty Beliefs Across Language Models An intriguing question is how different language models rank the difficulty of the problems. We measure the agreement between language models on problem difficulty by Spearman's rank correlation[5] of the solve rate for original problems across four datasets. As shown in Table 3, the ranks of the difficulty (by solve rate) are all positively correlated. However, the degree of correlation varies, with higher agreement observed within the GPT model family and on simpler datasets.

Paraphrase Transfer We investigate whether paraphrases from a stronger LLM can be transferred to weaker ones and improve SCoP. Table 4 demonstrates the paraphrase transfer performance of SCoP (Naïve, $k = 8$) on 100 randomly sampled data points from MMLU and GSM8K under the zero-shot CoT setting. In general, paraphrases produced by GPT-4 can be utilized by GPT3.5-turbo or LLaMA-2-70b for further performance improvements, with an exception with LLaMA-2 on MMLU, where GPT4 and LLaMA-2 exhibit the lowest Spearman rank correlation of solve rate. We hypothesize that the benefits of transferring paraphrases across models may depend on the agreement in their beliefs of problem difficulty.

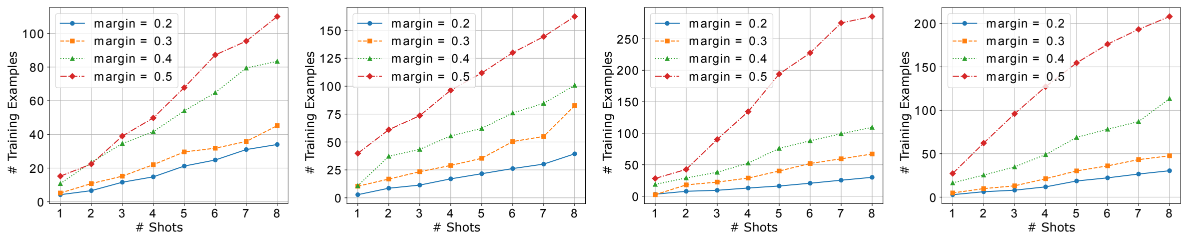
Variance of Variations In light of the considerable variability observed in solve rates among problem surface forms (Figure 2), we propose and advocate **Variance of Variations (VOV)** for evaluating language models on reasoning robustness. Let $X(p) \in [0, 1]$ be the random variable representing the solve rates of various paraphrases of a problem p. Then the VOV value of the dataset D is then defined as:

$$\text{VOV} = \mathbb{E}_{p\sim D}[\text{Var}(X(p))] \quad (1)$$

where $\text{Var}(\cdot)$ is the variance. A large value of VOV indicates high variability in the language model's reasoning ability against problem surface forms. We compute VOV using the solve rate for the $k = 8$ paraphrases and the original problem as $X(p)$ for each p. As shown in Table 5, while VOV decreases when a robust model solves a more manageable dataset (*e.g.*, GPT-4 on GSM8K), and ICL_{para} generated paraphrases can generally reduce VOV, VOV remains unreasonably high over more challenging datasets and all language models.

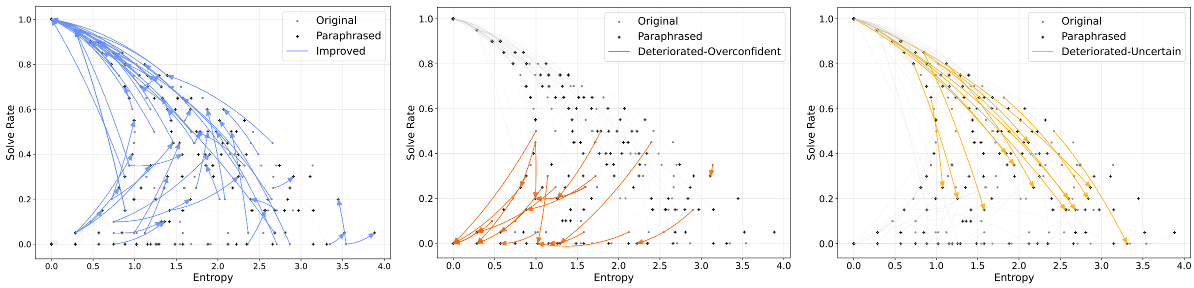
Examples of 'Good' Paraphrases We provide some qualitative examples comparing the solve rates between the original problem and a paraphrased version in Table 6. It is difficult to visually tell what contributes to a good paraphrase. We will publish these data to encourage future research.

[5]Using Python Scipy package.



(a) (b) (c) (d)

Figure 4: (a) GSM8K (b) AQuA (c) MATH (d) MMLU. The average number of data points in the training set needed for obtaining N_{shot} exemplars at different margins.



(a) (b) (c)

Figure 5: Data Difficulty Map for GSM8K using GPT3.5, with three types of changes from solving the original problem to one of its random paraphrases: (a) Improvement, (b) Overconfidence, and (c) Uncertainty.

5 Discussion

We have an intriguing observation that even the naïve scheme of generating math paraphrases can improve the overall accuracy. However, the naïve scheme has a significant chance of generating worse paraphrases. Why would aggregating over the mixture of better and worse paraphrases still significantly improve the performance?

To explain this, Figure 5 shows three scatter plots of the solve rate against the entropy of answer distributions. The outcome of solving each random paraphrase is represented as a black dot. As can be observed, the dots roughly form a triangular region. The top left corner represents the ideal case with high solve rates and high confidence. The bottom corners, on the other hand, represent two failure modes. The bottom right corner represents the case with low solve rates and low confidence, and the bottom left corner with low soft rates but high confidence (commonly known as over-confidence).

The blue arrows in Figure 5(a) visualize the cases where the paraphrases improve the solve rate, and they mostly point to the top-left corner. The arrows in Figures 5(b) and (c) represent the cases where the paraphrases lower the solve rate, and we can observe that the arrows pointing to the bottom right corner (yellow arrows in (b)) far outnumber those to the bottom left corner (red arrows in (c)). This indicates that while the 'good' paraphrases would sharpen the answer distribution, the 'bad' paraphrases mostly would flatten the distribution. Since the final aggregated answer distribution is predominantly influenced by the sharp distributions, the damage brought by the 'bad' paraphrases is small compared to the benefit brought by the 'good' paraphrases, and thus the aggregate effect across all the paraphrases is still positive.

6 Related Work

Mathematical Reasoning in LLMs The complexity of mathematics necessitates System-2 reasoning, characterized by a slow, step-by-step cognitive process (Kahneman, 2011). Numerous works have sought to emulate this process in solving mathematics with LLMs (Wei et al., 2022; Wang et al., 2022; Kojima et al., 2023; Lightman et al., 2023; Qiao et al., 2022). As a prominent framework, chain-of-thought (Wei et al., 2022; Kojima et al., 2023) prompts the language model to generate a sequence of reasoning steps instead of a direct answer; Wang et al. (2022) extended chain-of-thought

Problem	Source	Label	SR	Voted (F.)
Original: Jenna has 4 roommates. Each month the electricity bill is $100. How much will each roommate pay per year for electricity, if they divide the share equally? **Paraphrased**: Jenna shares an apartment with 4 other people. The electricity bill is $100 per month. If they split the bill equally, how much will each roommate contribute towards the electricity bill in a year?	GSM8K	"240"	0.15 0.95	"300" (0.8) "240"
Original: Jenny goes to the florist to buy some flowers. Roses cost $2 each and $15 for a dozen. If she bought 15 roses and arrived with five 5 dollar bills and they only have quarters for change, how many quarters does she leave with? **Paraphrased**: Jenny visits the flower shop to purchase flowers. She can buy roses individually for $2 each or buy a dozen roses for $15. Jenny decides to buy 15 roses in total. She pays with five $5 bills and the florist can only give her change in quarters. The question asks how many quarters Jenny receives as change.	GSM8K	"16"	0.0 0.55	"20" (0.3) "16"
Original: Assistants are needed to prepare for preparation. Each helper can make either 2 large cakes or 35 small cakes/hr. The kitchen is available for 3 hours and 20 large cakes & 700 small cakes are needed. How many helpers are required? **Paraphrased**: Jenny visits the flower shop to purchase flowers. She can buy roses individually for $2 each or buy a dozen roses for $15. Jenny decides to buy 15 roses in total. She pays with five $5 bills and the florist can only give her change in quarters. The question asks how many quarters Jenny receives as change. **Options**: `[A)8,B)10,C)12,D)15,E)19]`	AQUA	B	0.2 0.8	A (0.25) B
Original: A starts a business with Rs.40,000. After 2 months, B joined him with Rs.60,000. C joined them after some more time with Rs.120,000. At the end of the year, out of a total profit of Rs.375,000, C gets Rs.150,000 as his share. How many months after B joined the business, did C join? **Paraphrased**: A starts a business with Rs.40,000 and after 2 months, B joins with Rs.60,000. C joins the business at some point later with Rs.120,000. At the end of the year, the total profit is Rs.375,000, and C receives Rs.150,000 as their share. How many months after B joined the business did C join? **Options**: `[A)2,B)4,C)23,D)24,E)84]`	AQUA	B	0.1 0.45	C (0.4) B
Original: A star-polygon is drawn on a clock face by drawing a chord from each number to the fifth number counted clockwise from that number. That is, chords are drawn from 12 to 5, from 5 to 10, from 10 to 3, and so on, ending back at 12. What is the degree measure of the angle at each vertex in the star-polygon? **Paraphrased**: What is the measure of the angle at each vertex in the star-polygon formed by drawing a chord from each number on the clock face to the fifth number counted clockwise from that number?	MATH	"30"	0.05 0.5	"150" (0.4) "30"
Original: By partial fractions, $\frac{1}{ax^2+bx+c} = \frac{A}{x-\dfrac{-b+\sqrt{b^2-4ac}}{2a}} + \frac{B}{x-\dfrac{-b-\sqrt{b^2-4ac}}{2a}}$ Find $A+B$. **Paraphrased**: Find the sum of A and B in the expression $\frac{1}{ax^2+bx+c} = \frac{A}{x-\frac{-b+\sqrt{b^2-4ac}}{2a}} + \frac{B}{x-\frac{-b-\sqrt{b^2-4ac}}{2a}}$. *Note the LaTeXcode was paraphrased from `\frac` to `\dfrac`.	MATH	"0"	0.2 0.65	"1" (0.25) "0"
Original: Statement 1 | For every positive integer n there is a cyclic group of order n. Statement 2 | Every finite cyclic group contains an element of every order that divides the order of the group. **Paraphrased**: Statement 1 says that there exists a cyclic group of any positive integer n. Statement 2 says that in any finite cyclic group, there is an element for every possible order that divides the order of the group. **Options**: `[A)True,True, B)False,False, C)True,False, D)False,True]`	MMLU	A	0.05 0.5	C (0.95) A
Original: What is the probability that a randomly selected integer in the set $\{1, 2, 3, \ldots, 100\}$ is divisible by 2 and not divisible by 3? Express your answer as a common fraction. **Paraphrased**: What is the chance that if we randomly choose an integer from the set of numbers 1 to 100, it will be divisible by 2 but not divisible by 3? Write your answer as a fraction. **Options**: [A) $\frac{31}{66}$, B) $\frac{17}{66}$, C) $\frac{17}{31}$, D) $\frac{17}{50}$]	MMLU	D	0.25 0.8	A (0.4) D

Table 6: Qualitative examples where the original problems and corresponding surface form variations exhibit substantial solve rate difference using GPT-3.5-turbo.

by Self-Consistency, in which they replaced greedy decoding with sampling decoding to generate a variety of *reasoning paths*, with multiple paths potentially leading to the same answer from different angles. Other multi-step reasoning variations with verifiers exist (Lu et al., 2023; Besta et al., 2023; Yao et al., 2023); however, they are less related to our focus primarily on the language model's internal ability to solve mathematical problems.

Paraphrasing Variability Previous research on the impact of paraphrasing mathematical problems on their solvability by language learning models (LLMs) is limited. The study by (Gonen et al., 2022) explored how paraphrased instructions affect the performance of traditional NLP benchmarks. This sensitivity of instructive prompts has inspired further research in prompting learning (Shin et al., 2020; Zhou et al., 2023; Sordoni et al., 2023) and in-context exemplar mechanisms (Min et al., 2022; Brown et al., 2020; Ye and Durrett, 2022). However, our work focuses on the sensitivity of the mathematical problem presentation itself instead of the instruction or in-context examples.

7 Conclusions

This work highlights the variability in the solve rate of large-scale language models to the surface form of mathematical problems. Leveraging this, we introduced the Self-Consistency-over-Paraphrases (SCoP), which improves mathematical reasoning performance over Self-Consistency. We hope our findings will inspire the need for more robust language models that can reason effectively regardless of how a problem is presented.

8 Limitations

While we derive thorough conclusions about the relationship between the surface form of a mathematical problem and its solvability by large-scale language models with the effectiveness of SCoP and additional studies, one limitation is the need for a mechanism for identifying or generating surface forms that are easier to solve than others. Future research could address this by exploring the rationalization of surface forms, *i.e.*, determining the optimal form given the original one, using either a discriminative or a generative framework.

9 Ethics Statement

The datasets that we used in experiments are publicly available. In our work, we explore the relationship between the surface form of a mathematical problem and its solvability by large-scale language models. We do not expect any direct ethical concern from our work.

References

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.

Rahul Bhagat and Eduard Hovy. 2013. What Is a Paraphrase? *Computational Linguistics*, 39(3):463–472.

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.

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

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

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*.

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

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, et al. 2023. Holistic evaluation of language models.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, et al. 2022. Scaling language models: Methods, analysis and insights from training gopher.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV au2, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts.

Alessandro Sordoni, Xingdi Yuan, Marc-Alexandre Côté, Matheus Pereira, Adam Trischler, Ziang Xiao, Arian Hosseini, Friederike Niedtner, and Nicolas Le Roux. 2023. Joint prompt optimization of stacked llms using variational inference.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

H Lee Swanson, Jennifer E Kong, Amber S Moran, and Michael J Orosco. 2019. Paraphrasing interventions and problem-solving accuracy: Do generative procedures help english language learners with math difficulties? *Learning Disabilities Research & Practice*, 34(2):68–84.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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.

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.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in neural information processing systems*, 35:30378–30392.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers.

A Choose Margin

We examine the effect of margin on selecting exemplars for in-context paraphrasing using GPT-3.5 and separate dev-sets from GMS8K and MMLU, each with 250 data points. The results in Table 7 show that a moderate margin outperforms a large one in SCOP, as the latter may decrease the diversity of exemplars.

Margin	MMLU (Dev, k = 8)	GSM8K (Dev, k = 8)
SC/HPR%	53.20 (26.87) / 64	73.60 (21.43) / 33.6
0.2	55.60 (34.38)	75.20 (30.95)
0.3	56.80 (35.63)	74.80 (32.14)
0.4	55.20 (33.75)	75.60 (33.33)
0.5	53.60 (32.50)	74.40 (35.71)

Table 7: Ablation on the margin effect of exemplar selection.

B APE alternatives

A potential alternative to find an optimal prompt for paraphrasing is to use the Automatic Prompt Engineering (APE) settings (Zhou et al., 2023). We formulate the procedure into four steps:

1. Present a set of input-output pairs where the inputs are the original problems and the outputs are the paraphrased exemplars. Prompt the language model to generate C candidate instructions that could produce the outputs from the inputs.
2. Prompt each candidate instruction to the language model to generate paraphrases for a batch size B of problems in the development set and compare the mean solve rate change before and after paraphrasing.
3. Choose the instruction that maximizes the mean solve rate change.
4. Repeat steps 1 - 3 E times.

We implemented this procedure using GPT-3.5 on the AQUA development set to obtain the instruction ($C = 15$, $B = 30$, $B = 0$). We tested the performance in both AQUA (in-domain) and GSM8K (out-of-domain), comparing it with ICL$_{\text{para}}$. Although the in-domain AQUA performance was similar to ICL$_{\text{para}}$, the out-of-domain performance worsened and APE required more data than ICL$_{\text{para}}$. Therefore, this approach has yielded negative results. The performance results are presented in Table 8.

C Prompt Templates

We list the prompt templates used in the paper below.

Naïve Paraphrasing

Paraphrase the following math problem: `{target problem}`

ICL Paraphrasing

Paraphrase the following math problem: `{input problem}`
Output: `{Paraphrased exemplar}`
(Repeat N_{shot})

Paraphrase the following math problem: `{target problem}`

APE Candidate Search

A student is completing a task that requires producing a text output from a text input. The student receives instruction about several rules that describe how to produce the outputs given the inputs. What is the instruction?

Best Candidate Found by APE

Reformat the input sentence to make it more clear and concise. Ensure that all relevant information from the input is retained in the output. Use proper grammar and punctuation in the output. If there are percentages or mathematical expressions in the input, ensure they are accurately represented in the output. Maintain the logical flow of information from the input to the output. Ensure that the information is effectively conveyed in a clear and understandable manner when transforming the input into the output.

Few-shot Chain-of-thought

Question: At Academic Academy, to pass an algebra test you must score at least 80. If there are 35 problems on the test, what is the greatest number you can miss and still pass?
Answer Choices: A) 7 B) 28 C) 35 D) 8
Rationale: First, we need to find 80% of 35. We can do this by multiplying 35 by 0.80: $35 \times 0.80 = 28$. So, if you get 28 problems correct, you will have scored 80% on the test.
To find the greatest number you can miss and still pass, subtract the number you can get correct from the total number of problems:$35 - 28 = 7$.

Therefore, the greatest number you can miss and still pass is (A) 7.
(Repeat N_{shot})

Question: `{target problem}`

	GSM8K		AQUA	
	ICL_{para}	APE	ICL_{para}	APE
SC	76.33 (24.47)		66.93 (22.22)	
N/1	77.85 (38.98)	73.67 (34.04)	66.43 (29.81)	66.05 (28.97)
N/2	80.51(39.24)	76.33 (36.17)	68.50 (31.67)	66.84 (29.99)
N/4	79.18 (38.27)	77.67 (32.98)	70.47 (35.37)	70.78 (32.03)
N/8	80.18 (40.62)	79.0 (41.49)	69.69 (34.44)	69.20 (31.01)

Table 8: A comparison between the performance of APE and ICL_{para} paraphrasing.