An Analysis of the Impact of Problem Paraphrasing on LLM-Based Mathematical Problem Solving

Yerim Han, Hyein Seo, Hyuk Namgoong, Sangkeun Jung*
Computer Science and Engineering, Chungnam National University, Republic of Korea
{namu.rim2, hyenee97, hyuk199, hugmanskj}@gmail.com

Abstract

Recent advances in large language models (LLMs) have significantly improved mathematical problem-solving. Among various techniques, paraphrasing problem statements has emerged as a promising strategy to enhance model understanding and accuracy. We define twelve paraphrasing types grounded in mathematics education theory and analyze their impact on LLM performance across different configurations. To automate selection, we propose a *Paraphrase Type Selector* that predicts effective paraphrases for each problem. Experiments on MATH-500, SVAMP, and AIME shows consistent performance gain from paraphrased problems. On MATH-500 with LLAMA 3.1-8B, combining the original with the best five paraphrased problems improves accuracy by **+8.4%**, with the selector achieving an additional **+1.33%** gain.

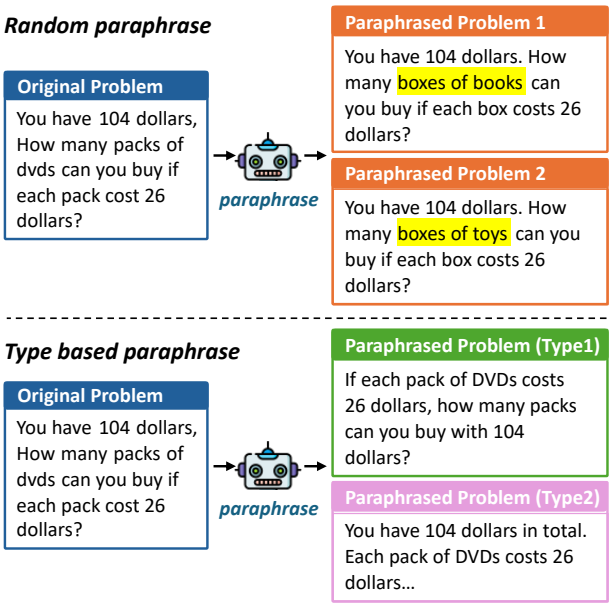

Figure 1: How Type based paraphrase works. A math problem is paraphrased into different forms. Random paraphrases may appear similar.

1 Introduction

Recent advances in large language models (LLMs) have improved their success in mathematical problem-solving, demonstrating high accuracy and efficiency in complex reasoning tasks.

Key approaches include Chain-of-Thought (CoT) (Wei et al., 2022) prompting, (Wang et al., 2023) and retrieval-based reasoning (Guan et al., 2025; Gao et al., 2023). Recently, paraphrasing problem statements has gained attention as a way to enhance model understanding and performance.

Problem paraphrasing is a technique that converts the original problem Q into semantically equivalent variants $\tilde{Q}$; numerous studies have shown that a model's accuracy can vary significantly depending on the specific paraphrasing method used. For instance, Zhou et al. (2024) report that paraphrasing leads to a $+10.2\%$ accuracy improvement on the MATH dataset for LLAMA-2-70B (Touvron et al., 2023) and a $+6.0\%$ improvement for GPT-3.5-TURBO (Brown et al., 2020).

Several strategies have been proposed, such as determining the final answer by measuring cross-paraphrase consistency (Lai et al., 2025), reformulating the input in the model's preferred style (Fu et al., 2024b), or allowing the model to reformulate math problems and solve them through code-based reasoning (Zhang et al., 2025).

However, current paraphrasing methods have key limitations. They often rely on random, prompt-based generation without a clear framework or taxonomy, and lack quantitative analysis of effective combinations of paraphrase types. In other words, **although paraphrasing is known to influence performance, empirical guidance on which paraphrases to use, when to apply them, and how many to include remains scarce**.

To address these limitations, we analyze how

*Corresponding author

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 383–395

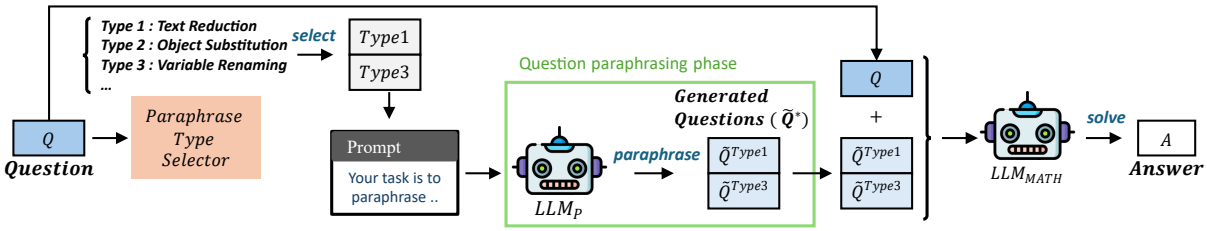

Figure 2: Our proposed multi-paraphrase problem-solving framework. Q denotes the original math problem. The *Paraphrase Type Selector* predicts a subset of effective paraphrase types. Based on these, the paraphrasing model LLM_P generates type-specific paraphrased problems $\tilde{Q}^{\text{Type}}$, which form a paraphrase set $\tilde{Q}^*$. The original problem Q and the selected paraphrased problems $\tilde{Q}^*$ are jointly fed into the solver model LLM_{MATH}, which produces the final answer A.

paraphrased problems affect LLM-based math problem solving. Grounded in mathematics education theory and prior work on problem paraphrasing and data augmentation, we define twelve paraphrasing types and generate multiple variants per problem. This setup enables quantitative analysis of how performance varies with paraphrase *number*, *combination*, and *type*, revealing when specific types are most effective. Figure 1 shows the key distinction between random and type-based paraphrasing. Randomly generated paraphrases are produced without constraints, and as a result, they often end up being overly similar to one another. In contrast, type-based paraphrasing applies predefined transformation rules to generate structurally distinct variants, promoting greater expression diversity and improving learning efficacy.

For each original problem, we generate a variable number of paraphrases and provide them to the model alongside the original. This design enables a systematic evaluation of how performance varies with the *number*, *combination*, and *type* of paraphrases, allowing us to identify the conditions under which specific types are most effective.

Building on these findings, we propose a Paraphrase Type Selector that selects effective paraphrase combinations based on the original problem. Experiments show it generalizes to unseen problems and consistently improves performance over the baseline. As shown in Figure 2, we further integrate the selector into a full multi-paraphrase problem-solving framework that leverages multiple paraphrased variants to enhance reasoning accuracy.

Experiments were conducted on three standard math reasoning benchmarks: MATH dataset (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), and AIME (Veeraboina, 2023). For MATH, we used a 500-item subset (MATH-500) obtained via random sampling. When providing the Top-n paraphrases alongside each original problem, we observed accuracy improvements on MATH-500 of up to **+9.73%** for LLaMA 3.1-8B, **+3.54%** for LLaMA 3.3-70B, **+1.33%** for Qwen 2.5-7B, and **+0.07%** for Qwen 2.5-72B. Furthermore, applying our proposed *Paraphrase Type Selector* provided additional improvements up to **+7.7%** over the original-only baseline on MATH-500, confirming the effectiveness of automated paraphrase selection.

The key contributions of this paper are as follows:

1. We present a taxonomy of twelve paraphrasing types, rooted in mathematics education theory, that preserve semantics while varying surface form.
2. We develop a multi-paraphrase prompting strategy and provide quantitative analysis on how paraphrase *type*, *number*, and *combination* affect LLM accuracy.
3. We propose a *Paraphrase Type Selector* that automatically selects effective paraphrases per problem and empirically improves performance.

2 Related Works

Problem Paraphrasing and Expression Variation Multiple studies have demonstrated that the surface formulation of a math problem can substantially affect an LLM's reasoning accuracy. Zhou et al. (2024) analyze how solving accuracy varies when the same problem is paraphrased in different ways and introduce Self-Consistency-over-Paraphrases (SCoP) to aggregate answers across variants. Zhang et al. (2025) combine paraphrasing with code-based reasoning in the RM-PoT framework, while Fu et al. (2024b) mitigate paraphrasing variance by aligning problems to

a model-preferred style in their PEARL framework. Deng et al. (2023) further show that letting a model first rephrase a question and then respond (Rephrase-and-Respond) yields clearer formulations and higher accuracy. Research has also addressed problem perturbations beyond paraphrasing, such as logical, computational, and conceptual transformations (Hong et al., 2024).

Analogical and Retrieval-based Reasoning Research has also explored exploiting analogous or retrieved problems. Lin et al. (2025) propose Meta-Ladder, which provides similar examples and solutions to induce analogy-based reasoning. Yasunaga et al. (2023) introduce analogical prompting, where the model self-generates similar examples for use in reasoning. Conversely, Qin et al. (2024) compare relevant and irrelevant examples to critically assess whether LLMs truly exploit analogy. Additional lines of work include memory-augmented solvers that recall previously solved problems (Huang et al., 2021) and retrieval-augmented generation (RAG) that grounds reasoning with textbook knowledge (Levonian et al., 2023).

Hint-based and Structured Prompting Prompting techniques that supply hints or induce structural analysis have also been proposed. Agrawal et al. (2024) show that providing explicit hints before the problem statement boosts performance, whereas Fu et al. (2024a) have the model generate its own hints via Hint-before-Solving Prompting (HSP). Yugeswardeenoo et al. (2024) introduce Question-Analysis Prompting, which guides the model to structurally parse the problem prior to solving.

Aggregation and Consistency-based Approaches A complementary strand aggregates diverse inputs or reasoning paths to increase answer reliability. Imani et al. (2023) propose MathPrompter, which explores multiple solution strategies and compares outcomes. Lai et al. (2025) present the MRC framework, performing self-consistency over variations in formulation, example order, and even language. In the classification domain, Yadav et al. (2024) improve accuracy by aggregating predictions across multiple paraphrases, illustrating the utility of expression diversity in voting schemes.

Prior work has thus shown that changing problem formulation can influence LLM performance and that diversity can be exploited for greater consistency or accuracy. Building on this line of research, we propose an approach that presents the original problem alongside multiple paraphrased variants to better highlight core concepts.

Category	Index	Abbr.	Paraphrase Types
Expression	1	MET	Mathematical Expression Transformation
	2	TR	Text Reduction
	3	TW	Text Rewriting
	4	SOM	Sentence Order Modification
	5	SM	Sentence Merging
	6	MR	Metaphorical Representation
	7	ST	Syntactic Transformation
Object	8	OS	Object Substitution
	9	VR	Variable Renaming
	10	CR	Contextual Rewriting
Structural	11	SBS	Step-by-Step Breakdown
	12	IS	Information Segmentation

Table 1: Mathematical problem paraphrasing types.

3 Mathematical Problem Paraphrasing

3.1 Education-based Paraphrasing Types

For decades, the mathematics-education community has viewed *problem paraphrasing* as a key means of cultivating mathematical thinking and improving problem-solving skills. Activities that modify existing problems or construct new ones in varied ways are widely used as pedagogical strategies to foster creative reasoning.

In parallel, the NLP community has investigated *paraphrase types* (Wahle et al., 2023), typically assessing sentence-level semantic equivalence and classifying linguistic paraphrases such as lexicon-based, syntax-based, morphology-based, while proposing general types for generating semantically diverse variants.

Building on both lines of work, we extend these ideas to the mathematical domain. Drawing on prior research in mathematics education and data augmentation (Cai et al., 2015; Christou et al., 2005; Baumanns and Rott, 2022; Singer et al., 2015), we systematically define twelve paraphrase types that preserve the underlying mathematical operations and logical flow required for problem solving, while altering only surface-level features such as wording, sentence structure, and contextual framing.

Table 1 summarizes our taxonomy of twelve paraphrasing types, organized into three high-level categories based on expression features specific to mathematical problems. These types are designed to preserve the core mathematical logic and operations while varying only surface-level elements such as wording, structure, or context.

The twelve types can be classified into three

Problem Paraphrase Prompt (P_T)

You are a helpful math assistant. Your task is to paraphrase a math problem using the following technique:

Technique : **{paraphrase type}**
Description : **{description}**

Here is an example :
Original : **{original example}**
Paraphrased : **{paraphrased example}**

Now, apply the technique to the following problem.
Only output the paraphrased problem statement.
Ensure that the solution remains logically equivalent.

Original Problem: **{original problem}**

Figure 3: Prompt for problem paraphrase generation.

broader functional categories:

- **Expression Paraphrase** (MET, TR, TW, SOM, SM, MR, ST): Modify equations, phrasing, and syntax to create diverse surface forms.
- **Object Paraphrase** (OS, VR, CR): Change variables, objects, or contextual settings without altering the problem's logic.
- **Structural Paraphrase** (SBS, IS): Restructure information by breaking down step-by-step or reorganizing the problem presentation.

These types are defined based on expression features unique to mathematical problems—such as how mathematical objects are referenced, how conditions are phrased, and whether the problem is staged step-by-step. Unlike general paraphrasing that aims for semantic equivalence, our taxonomy emphasizes preserving the mathematical problem-solving process itself.

3.2 Problem Paraphraser

To transform mathematical problems into diverse expressions, our study employs an LLM-based mathematical problem paraphraser LLM_P and generates problems as follows:

$$\tilde{Q}_k^i = LLM_P(P_T^i,\ Q_k)$$

Using the k-th original problem Q_k and the paraphrase prompt P_T^i corresponding to paraphrase type i, the paraphrased problem $\tilde{Q}_k^i$ is produced.

Each paraphrase type is summarized in Table 1, and an example prompt is presented in Figure 3. The generated paraphrased problem is designed to be logically equivalent to the original problem and therefore shares the same A_k. The main notation used is summarized in Table 2. Further details can be found in Appendix B, C.

Notation	Description
Q_k	k-th original problem
$\tilde{Q}^i$	Problem from i-th paraphrase type
$\tilde{Q}^*$	Best-per-Instance paraphrase set
$\tilde{Q}^{*,\text{selector}}$	Selector-chosen paraphrase set
A	Ground-truth answer
$\hat{A}$	Predicted answer
$\hat{A}^i$	Predicted answer for $\tilde{Q}^i$
$\hat{A}^{i,n}$	n-th attempt for $\tilde{Q}^i$
C_k^i	Correct count for $\tilde{Q}^i$ of Q_k
C_k^{sorted}	Paraphrases sorted by correctness
N	Number of attempts per problem
P_S	Standard problem-solving prompt
$\tilde{P}_S$	Extended prompt with paraphrases
P_T	Prompt used to paraphrases
P^i	Prompt for i-th paraphrase type
$\mathcal{I}^+$	Index set of effective paraphrase type

Table 2: Notation used in the paraphrased problem-solving framework.

4 Analyzing the Impact of Paraphrased Problems on Math Problem Solving

4.1 Problem Solver

Mathematical problem solving is performed with a solver LLM_{Math}, defined as follows:

$$\tilde{\boldsymbol{Q}}_{\boldsymbol{k}}^* = \underbrace{\left\{\tilde{Q}_k^1,\ \tilde{Q}_k^2,\ \ldots,\ \tilde{Q}_k^I\right\}}_{\text{Selected Paraphrased Questions}}$$

$$Base: \quad \hat{A}_k = LLM_{Math}(P_S,\ Q_k)$$

$$Ours: \quad \hat{A}_k = LLM_{Math}(\tilde{P}_S,\ Q_k;\ \tilde{\boldsymbol{Q}}_{\boldsymbol{k}}^*)$$

As shown in Figure 4, our prompt provides the original problem together with several semantically equivalent paraphrased versions in a single input, allowing the model to reason with more information. These paraphrased versions form a selected set $\tilde{\boldsymbol{Q}}_{\boldsymbol{k}}^*$ for problem Q_k; the set can be chosen manually via heuristic rules or automatically by an LLM-based selector to aim maximize performance.

The standard prompt P_S is used for ordinary problem solving, whereas the extended prompt $\tilde{P}_S$

Original Only Prompt (P_S)

You are a math expert.

Solve the following math problem step-by-step.
At the end, provide your final answer in the form \boxed{...}.

Problem: **{original problem}**

Answer:

(a) Prompt for original-only math problem solving.

Paraphrase Augmented Prompt (Ours) - $\tilde{P}_S$

You are a math expert.
The following is an original math problem along with five paraphrased versions of the same problem.
Analyze them together and solve the original problem step-by-step.
At the end, provide your final answer in the form \boxed{...}.
If paraphrased versions help clarify the meaning, use them.

Original problem: {**original problem**}

Paraphrased 1 ({**type**}): {**paraphrased Problem**}
Paraphrased 2 ({**type**}): {**paraphrased Problem**}

Answer:

(b) Prompt for math problem solving with paraphrases. Blue text marks the paraphrase-generating instruction.

Figure 4: Prompts for math problem solving.

is employed when the paraphrased versions are included. The model's final output is the predicted answer $\hat{A}_k$ for the given problem.

4.2 Experimental Analysis of Paraphrase Effects

Experiments were conducted on the MATH-500 (Hendrycks et al., 2021), SVAMP (Patel et al., 2021) and AIME (Veeraboina, 2023) datasets with the LLAMA 3.1-8B-INSTRUCT, LLAMA 3.3-70B-INSTRUCT (Grattafiori et al., 2024), and QWEN 2.5-7B-INSTRUCT, QWEN2.5-72B-INSTRUCT (Yang et al., 2024) models and GPT-4O-2024-11-20 (Achiam et al., 2023). Each model was evaluated over three runs for robustness. Paraphrased problems were generated by GPT-4.1-MINI (Achiam et al., 2023) and finalized through dual verification by a human reviewer and the same model. All answers were evaluated using Harness (Gao et al., 2024) to ensure consistent and accurate scoring. Refer to Appendix D for details.

To evaluate the impact of paraphrased versions on LLM performance in mathematical problem solving, we conducted four experiments. Further experiments can be found in the Appendix E:
(1) using paraphrased versions only,
(2) varying the number of paraphrases provided,
(3) supplying the problem-specific optimal paraphrase set, and
(4) selecting paraphrases using a trained *Paraphrase Type Selector*.

4.2.1 Performance Analysis with Paraphrased Versions Only

This experiment investigates whether LLMs can solve problems using only paraphrased versions, without access to the original. For each problem in MATH-500, we generated paraphrased problems using predefined types and solved each three times with LLAMA 3.1-7B. We then identified the five highest-performing paraphrase types (SBS, TW, TR, ST, MET) based on aggregate accuracy.

Using these top types, we constructed input sequences containing n paraphrased versions (excluding the original) and compared their performance to a baseline that used only the original problem.

As shown in Table 3, using only paraphrased inputs improved accuracy by up to **+14.8%** for LLAMA 3.1-7B, **+2.54%** for LLAMA 3.3-70B, and **+1.2%** for QWEN 2.5 on MATH-500. These results indicate that certain paraphrases align more closely with model preferences, and that problem surface formulation plays a critical role in mathematical reasoning performance.

To analyze which paraphrase types are consistently effective, we applied all 12 paraphrase types to each problem in the MATH-500 dataset using the LLAMA-3.1-8B-INSTRUCT model.

4.2.2 Performance Analysis with Varying Numbers of Paraphrased Versions

Experiments were conducted under settings where **one to five** paraphrased versions were added to the original problem, in order to analyze how the number of paraphrases affects accuracy. The paraphrased versions were selected from the top-performing types identified in the previous experiment.

As shown in Table 3, adding paraphrases leads to accuracy gains of up to **+8.4%** for LLAMA 3.1-8B, **+3.27%** for LLAMA 3.3-70B, and **+2.66%** for QWEN 2.5-7B on MATH-500. While performance generally improves as more paraphrases are added, the trend does not always hold: in some models, adding too many paraphrases results in a slight drop in accuracy. These results suggest that

# Paraphrased Questions (n)	SVAMP					MATH-500					AIME 2024				
	LLaMA		Qwen		GPT-4o	LLaMA		Qwen		GPT-4o	LLaMA		Qwen		GPT-4o
	8B	70B	7B	72B		8B	70B	7B	72B		8B	70B	7B	72B	
Original Problem Only															
Q only (baseline)	73.61 ± 0.73	96.17 ± 0.25	95.35 ± 0.15	95.81 ± 0.38	89.43 ± 1.00	35.40 ± 1.28	67.13 ± 1.87	67.27 ± 0.47	77.20 ± 0.69	67.07 ± 1.04	2.22 ± 1.92	26.67 ± 3.34	11.11 ± 6.94	16.67 ± 6.67	11.11 ± 5.09
Paraphrased Problem Only															
$\tilde{Q}$ (n=1)	69.73 ± 0.32	95.35 ± 0.19	95.60 ± 0.20	95.76 ± 0.29	**92.96** ± 0.54	42.80 ± 0.28	67.87 ± 0.75	68.40 ± 0.57	73.19 ± 1.60	64.67 ± 1.47	3.33 ± 3.34	24.45 ± 3.85	8.89 ± 1.92	16.67 ± 3.34	6.67 ± 3.34
$\tilde{Q}$ (n=2)	74.17 ± 0.69	**96.53** ± 0.19	**95.66** ± 0.50	**96.63** ± 0.33	92.70 ± 0.94	43.93 ± 0.19	67.80 ± 1.56	67.53 ± 0.34	72.47 ± 1.32	64.20 ± 0.28	5.55 ± 6.94	**25.56** ± 5.09	7.78 ± 5.09	**21.11** ± 6.94	11.11 ± 1.92
$\tilde{Q}$ (n=3)	74.62 ± 1.04	95.40 ± 0.22	95.40 ± 0.66	95.91 ± 0.29	91.27 ± 0.33	44.60 ± 1.14	70.40 ± 0.59	69.40 ± 0.40	**74.93** ± 0.74	65.87 ± 0.57	**5.56** ± 1.93	21.11 ± 3.85	**12.22** ± 5.09	17.78 ± 3.85	**12.22** ± 3.85
$\tilde{Q}$ (n=4)	74.68 ± 0.94	95.61 ± 0.29	95.41 ± 0.13	96.02 ± 0.12	91.22 ± 0.07	49.80 ± 0.99	69.13 ± 0.25	**70.67** ± 0.50	73.87 ± 0.90	66.00 ± 0.99	4.44 ± 5.09	23.33 ± 3.34	7.78 ± 5.09	15.55 ± 6.94	7.78 ± 3.85
$\tilde{Q}$ (n=5)	**75.85** ± 1.28	96.02 ± 0.12	**95.77** ± 0.15	96.12 ± 0.32	91.12 ± 0.57	**50.20** ± 0.71	**69.67** ± 0.38	68.47 ± 0.90	74.33 ± 0.57	**66.80** ± 0.49	2.22 ± 1.92	17.78 ± 3.85	11.11 ± 1.92	15.56 ± 1.93	8.89 ± 1.92
Original Problem + Top-N Paraphrased															
$Q + \tilde{Q}$ (n=1)	75.75 ± 1.44	96.22 ± 0.38	95.51 ± 0.40	**96.53** ± 0.19	**91.37** ± 0.56	37.73 ± 1.23	70.27 ± 0.41	68.27 ± 0.09	76.00 ± 1.28	**67.60** ± 0.71	2.22 ± 1.92	24.44 ± 1.93	11.11 ± 7.70	12.22 ± 3.85	10.00 ± 3.33
$Q + \tilde{Q}$ (n=2)	**79.48** ± 0.75	96.22 ± 0.07	95.56 ± 0.90	96.48 ± 0.13	89.54 ± 0.64	37.33 ± 2.13	69.33 ± 0.90	68.80 ± 1.56	76.33 ± 0.19	**67.60** ± 0.75	4.44 ± 1.93	**25.56** ± 5.09	10.00 ± 3.33	**17.78** ± 6.94	10.00 ± 6.67
$Q + \tilde{Q}$ (n=3)	77.89 ± 1.00	96.02 ± 0.33	95.97 ± 0.26	95.36 ± 0.31	90.15 ± 0.62	38.20 ± 0.75	69.87 ± 0.09	68.33 ± 1.39	75.80 ± 0.71	67.40 ± 0.16	**7.78** ± 1.92	18.89 ± 3.85	11.11 ± 1.92	**17.78** ± 1.92	**15.56** ± 1.93
$Q + \tilde{Q}$ (n=4)	77.69 ± 1.16	95.97 ± 0.31	95.66 ± 0.15	96.17 ± 0.43	89.38 ± 1.04	43.53 ± 1.43	**70.40** ± 0.57	69.20 ± 1.50	75.60 ± 0.28	67.13 ± 0.25	2.22 ± 1.92	21.11 ± 5.09	11.11 ± 1.92	16.67 ± 6.67	10.00 ± 3.33
$Q + \tilde{Q}$ (n=5)	78.61 ± 0.58	**96.27** ± 0.07	**96.20** ± 0.47	95.56 ± 0.22	89.18 ± 0.19	**43.80** ± 1.56	69.87 ± 0.34	**69.93** ± 1.20	**76.87** ± 0.09	66.40 ± 0.33	5.55 ± 3.85	18.89 ± 3.85	**12.22** ± 6.94	14.45 ± 3.85	11.11 ± 1.92
Best-per-Instance (Original Problem + Problem-Specific Best Top-N Paraphrased)															
$Q + \tilde{Q}$ (n=1)	75.65 ± 1.25	96.88 ± 0.15	96.22 ± 0.19	96.84 ± 0.40	**91.01** ± 0.08	39.20 ± 0.00	**71.87** ± 0.77	70.20 ± 1.14	75.60 ± 0.28	**68.00** ± 0.57	3.33 ± 3.34	18.89 ± 1.92	13.33 ± 3.34	16.67 ± 3.34	6.67 ± 3.34
$Q + \tilde{Q}$ (n=2)	80.19 ± 1.05	96.83 ± 0.08	95.71 ± 0.38	**97.09** ± 0.22	90.51 ± 1.19	39.40 ± 1.56	71.33 ± 0.62	**70.33** ± 1.23	76.13 ± 0.68	67.73 ± 0.98	2.22 ± 1.92	23.33 ± 0.00	12.22 ± 1.92	15.56 ± 5.09	8.89 ± 1.92
$Q + \tilde{Q}$ (n=3)	80.35 ± 1.63	**96.99** ± 0.51	**96.38** ± 0.40	96.79 ± 0.22	90.45 ± 0.40	41.67 ± 0.68	70.47 ± 0.57	68.60 ± 1.42	76.40 ± 1.13	67.33 ± 0.41	3.33 ± 3.34	**24.44** ± 5.09	13.34 ± 5.77	15.55 ± 3.85	**13.33** ± 3.34
$Q + \tilde{Q}$ (n=4)	**81.37** ± 0.80	96.83 ± 0.08	95.87 ± 0.13	96.94 ± 0.50	90.71 ± 0.38	42.27 ± 0.75	71.13 ± 0.09	68.80 ± 0.16	76.73 ± 1.25	67.00 ± 0.49	2.22 ± 3.85	**24.44** ± 5.09	13.33 ± 0.00	**20.00** ± 0.00	8.89 ± 1.92
$Q + \tilde{Q}$ (n=5)	81.06 ± 0.83	96.53 ± 0.40	**96.38** ± 0.26	96.22 ± 0.40	89.69 ± 0.52	**45.13** ± 0.62	70.67 ± 0.19	68.60 ± 0.59	**77.27** ± 1.00	67.13 ± 0.09	**6.67** ± 3.34	22.22 ± 3.85	**15.55** ± 3.85	**20.00** ± 3.33	7.78 ± 3.85
Selector (Original Problem + Selected Top-N Paraphrased)															
$Q + \tilde{Q}$ (n=5)	78.97 ± 1.03	96.68 ± 0.26	96.02 ± 0.46	96.47 ± 0.55	90.76 ± 0.54	43.10 ± 1.56	70.47 ± 0.57	69.67 ± 1.84	78.11 ± 1.56	68.73 ± 0.73	8.34 ± 2.36	25.56 ± 1.54	7.78 ± 3.09	15.56 ± 1.54	8.89 ± 1.54

Table 3: Performance across different paraphrasing conditions. Q denotes the original problem, and $\tilde{Q}$ represents its paraphrased problems. The number n indicates how many paraphrased problems were included in the prompt. We compare settings where only the original problem is used, only paraphrased problems are used, or both are combined. The "Top-n" setting uses fixed high-performing paraphrase types, "Best-per-Instance" selects the best combination for each problem, and "Selector" automatically predicts effective paraphrase types.

a moderate number of paraphrased versions can enhance performance, but excessive additions may introduce noise and lead to diminishing or even negative returns.

4.2.3 Performance Analysis with Problem-Specific Best Paraphrased Versions

This experiment shows the upper bound of paraphrase effectiveness by selecting the best-performing paraphrases for each problem. Unlike previous settings that use the same fixed top-n paraphrase types across all problems, this Best-per-Instance setting identifies which paraphrases actually led to correct answers on a per-problem basis, and provides those in the input along with the original problem. If fewer than n effective paraphrases were available for a given problem, the remaining slots were filled using paraphrases from

Paraphrase Type	Algebra	Counting & Probability	Geometry	Intermediate Algebra	Number Theory	Prealgebra	Precalculus
Original	43.01 (±3.98)	14.04 (±1.52)	21.14 (±1.41)	13.74 (±1.56)	30.11 (±3.34)	**37.40** (±2.79)	12.50 (±1.79)
MET	36.29 (±5.29)	14.91 (±1.52)	20.32 (±1.41)	**15.12** (±2.63)	25.14 (±6.81)	34.56 (±1.25)	9.52 (±0.99)
TR	40.59 (±3.36)	**16.66** (±8.04)	**21.14** (±7.84)	**16.49** (±0.00)	27.96 (±3.23)	31.71 (±4.56)	8.93 (±3.57)
TW	**43.01** (±1.86)	14.91 (±8.04)	17.07 (±4.88)	**16.50** (±2.70)	**31.72** (±6.48)	29.68 (±8.23)	11.90 (±0.99)
SOM	42.20 (±5.84)	**17.54** (±3.04)	18.70 (±5.08)	13.40 (±1.03)	28.50 (±3.23)	35.37 (±3.66)	8.93 (±0.00)
SM	40.05 (±2.46)	8.77 (±4.02)	14.63 (±0.00)	12.69 (±1.55)	26.88 (±7.44)	33.74 (±4.01)	8.93 (±1.78)
MR	19.89 (±2.83)	9.65 (±5.48)	13.01 (±3.72)	5.84 (±1.57)	16.13 (±2.79)	15.44 (±3.48)	5.95 (±2.69)
ST	40.32 (±2.91)	12.28 (±1.52)	17.88 (±1.41)	12.69 (±1.55)	29.70 (±4.04)	37.00 (±1.73)	9.52 (±2.69)
OS	35.48 (±0.81)	12.28 (±5.48)	13.82 (±2.81)	10.65 (±3.11)	19.36 (±4.14)	26.68 (±2.44)	9.52 (±2.69)
VR	38.17 (±1.68)	14.04 (±4.02)	13.82 (±3.72)	13.75 (±5.19)	24.73 (±0.82)	32.12 (±4.93)	8.93 (±0.00)
CR	33.07 (±3.52)	14.04 (±3.04)	**21.14** (±2.82)	11.68 (±0.57)	26.88 (±1.87)	34.15 (±3.16)	11.90 (±0.99)
SBS	**46.24** (±3.64)	**23.68** (±5.27)	**30.90** (±5.63)	13.74 (±1.56)	**33.33** (±4.08)	36.58 (±5.33)	**13.69** (±5.16)
IS	40.59 (±1.68)	5.26 (±2.63)	20.32 (±3.73)	14.43 (±2.64)	28.50 (±3.73)	30.49 (±6.14)	10.71 (±1.78)

Table 4: Accuracy by paraphrase type across subjects with LLAMA-3.1-8B-INSTRUCT. Best-performing cells are highlighted in Blue.

the globally top-ranked list that achieved the same score, prioritized by their overall performance.

As shown in Table 3, the Best-per-Instance setting achieves the highest accuracy across most models. For instance, accuracy improves by **+9.73%** for LLAMA 3.1-8B, **+4.74%** for LLAMA 3.3-70B, **+3.06%** for QWEN 2.5-7B, and **+0.07%** for QWEN 2.5-72B, relative to the original-only baseline on MATH-500. These findings confirm that selecting problem-specific best paraphrases can substantially improve the quality of inputs for mathematical reasoning tasks.

4.2.4 Analysis of Performance Improvement

As shown in Table 3, all settings using paraphrased versions outperformed the original-only baseline. The highest gains were achieved when, selecting the best-fitting paraphrases per problem, highlighting the importance of aligning paraphrases with the problem's formulation to boost LLM performance. Further details can be found in Appendix E.1

4.2.5 Analysis of Individual Paraphrase Types

As shown in Table 4, Step-by-Step (SBS), Text Rewriting (TW), and Text Reduction (TR) frequently outperformed the original across multiple subjects. Notably, SBS achieved substantial improvements in Algebra and Counting & Probability, while TW consistently boosted performance in Algebra and Number Theory. These results suggest that paraphrase types that either (i) make the reasoning path explicit or (ii) align the problem with linguistic patterns familiar to the model are particularly beneficial.

5 Multi-paraphrase problem solving framework

The Best-per-Instance study identified the best paraphrase set per problem in hindsight, but practical systems must select effective paraphrase types for new problems. To this end, we propose a *Paraphrase Type Selector Model* that predicts helpful paraphrase types given an original problem, formulated as a multi-label classification task.

Building on prior results showing the benefits of combining original and paraphrased inputs, we also introduce a *multi-paraphrase problem-solving framework*. As shown in Figure 2, the selector guides paraphrase generation, and the original and paraphrased problems are jointly provided to the solver. This enables automated, problem-specific augmentation that improves math reasoning accuracy.

5.1 Construction of the Paraphrase Dataset

1. Starting from the 6,500 training problems in MATH, we used GPT-4.1-MINI to generate I paraphrased versions $\tilde{Q}_k^1, \tilde{Q}_k^2, \ldots, \tilde{Q}_k^I$ for each original problem Q_k by applying the predefined paraphrase types. In this study $I = 12$; each paraphrase was produced with a paraphrase type-specific prompt.
2. For every paraphrased problem $\tilde{Q}_k^i$, the mathematical solver LLM_{Math} was invoked to obtain a predicted answer $\hat{A}_k^i$.
3. This procedure was repeated N times for each problem, yielding per-trial predictions $\hat{A}_k^{i,n}$. We set $N = 3$.
4. For each type i, we counted how many of the N trials were correct and denoted that count by

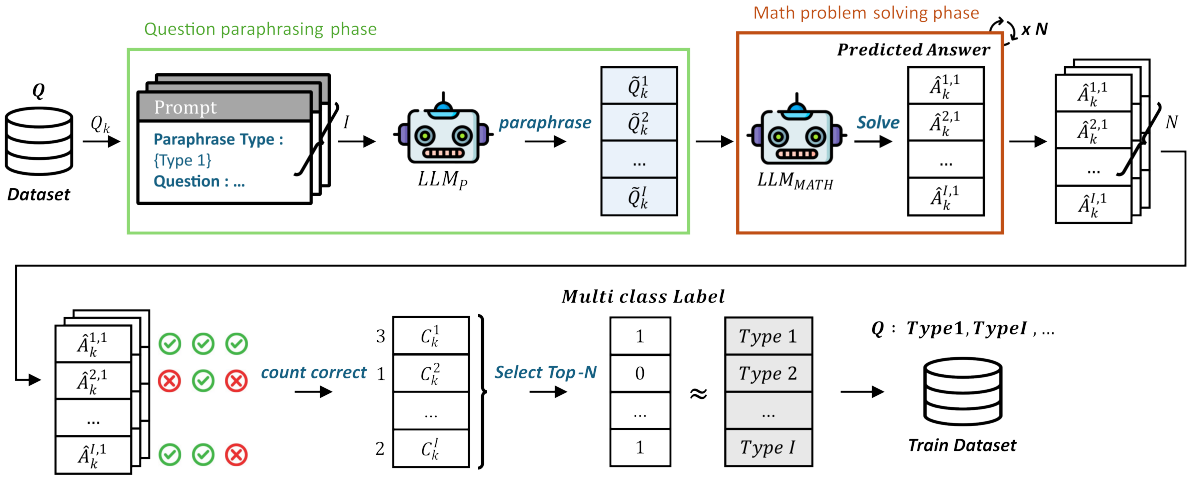

Figure 5: Construction of the paraphrase dataset. For each original problem Q_k in the dataset, we generate I paraphrased variants $\tilde{Q}_k^i$ using prompts specific to each paraphrasing type via LLM_P. These paraphrased problems are then solved by LLM_{MATH} to obtain predicted answers $\hat{A}_k^{i,j}$, where i indexes the paraphrasing type and $j \in \{1, \ldots, N\}$ denotes repeated trials. The number of correct predictions per type is counted to compute a score C_k^i, which is used to select the top-N effective paraphrasing types. These are then converted into multi-class labels (1 for selected, 0 otherwise), yielding training instances that map each original problem Q_k to a set of effective paraphrase types.

C_k^i. For example, if technique 1 produced two correct answers out of three trials, then $C_k^1 = 2$.

5. For every problem Q_k, the types were ranked in descending order of C_k^i to form the priority list $\mathcal{C}_k^{\text{sorted}}$.
6. The top n technique indices were extracted as the effective paraphrasing set for that problem and used as positive multi-class labels to train the paraphrase selection model.

The overall data-generation pipeline is illustrated in Figure 5.

5.2 Paraphrase Type Selector

The paraphrase selection task is formulated as a multi-label classification problem: given an original problem, the model predicts which of the predefined paraphrase types are likely to be beneficial for solving it. We use MODERNBERT-BASE (Warner et al., 2024) as the selection model. The input is the original problem text, and the output is a 12-dimensional probability vector, one dimension per types. For each problem, types that actually improved performance are labeled 1, and the rest are labeled 0.

Training details are as follows: learning rate 2×10^{-5}, batch size 16, maximum input length 512 tokens, and 50 training epochs. Early stopping is triggered if no improvement is observed on the validation set for three consecutive evaluations.

5.3 Problem Solver

Mathematical problem solving is carried out with the solver LLM_{Math}, defined as follows:

$$\hat{A}_k = LLM_{Math}(\tilde{P}_S,\ Q_k;\ \tilde{\boldsymbol{Q}}_{\boldsymbol{k}}^{*,\text{selector}})$$

where, $\tilde{\boldsymbol{Q}}_{\boldsymbol{k}}^{*,\text{selector}}$ denotes the set of paraphrased versions automatically assembled from the types predicted by the selection model.

5.4 Performance Analysis with the Paraphrase Selection Model

This experiment evaluates the effectiveness of our *Paraphrase Type Selector*, which aims to automatically select the most beneficial paraphrase types for each problem. For each original question, the selector predicts the top-n paraphrase types expected to improve accuracy. The corresponding paraphrased versions are then generated using GPT-4.1-MINI and appended to the original problem in the prompt. We fix n=5, as it consistently produced strong results across datasets.

As shown in Table 3, the selector consistently outperforms the *baseline* across all datasets and models. For instance, on MATH-500, it improves LLaMA 3.1-8B from 35.40% to 43.10%, and Qwen 2.5-7B from 67.27% to 69.67%. Similar gains are observed on SVAMP and AIME 2024. Although the selector does not surpass the *Best-per-Instance* upper bound, its performance is competitive with

the average top-n strategy and demonstrates strong generalization to unseen problems.

These results show that learning problem-specific paraphrase preferences is feasible, and that automated selection is an effective way to boost LLM performance in math reasoning.

6 Discussion

We organize our discussion around four key observations from the experiments:

Effectiveness of Paraphrasing Over Baseline Across all datasets and models, providing paraphrased versions—either alone or in combination with the original question—consistently outperformed the baseline using only the original problem. Notably, even when the original question was excluded, LLMs could solve many items correctly, indicating that certain reformulations better align with model preferences. Furthermore, the *Best-per-Instance* condition, which selects the best paraphrases per problem, showed substantial gains over the baseline, confirming that well-chosen paraphrases can significantly enhance performance.

Impact of the Number of Paraphrased Versions Increasing the number of paraphrased problems generally improved performance, with the n=5 setting achieving the highest accuracy in most cases. However, the marginal benefits diminished with larger n, suggesting that while diverse formulations are helpful, too many inputs may introduce redundancy or noise.

Selector Performance and Automation Potential Our proposed *Paraphrase Type Selector* model, which predicts effective paraphrase types per problem, achieved meaningful improvements over the baseline. On MATH-500, the selector reached near *Best-per-Instance* performance for some models (e.g., QWEN2.5-7B), demonstrating the feasibility of automated paraphrase selection. Importantly, our contribution is not the selector's performance itself, but the finding that some paraphrase combinations are consistently more effective than others, and that such combinations can be automatically identified even with a simple classifier.

Dependency on Problem Difficulty and Model Capability Paraphrasing is most effective for moderately difficult problems. Easy problems are solved regardless of phrasing, while hard ones remain unsolved even with paraphrases. Stronger models benefit less, as they often solve the original directly, but paraphrasing still boosts average accuracy and helps recover otherwise missed solutions.

Limited Generalization to AIME While the selector performed well on MATH-500 and SVAMP, its performance on AIME 2024 was relatively weaker. This may reflect differences in problem style or structure, suggesting that paraphrase preferences learned from one dataset may not directly carry over to others.

7 Conclusion

Our study examines the impact of problem paraphrasing on LLM performance using the MATH-500 dataset under four settings: baseline, parallel paraphrases, Best-per-Instance, and selector. We propose a Paraphrase Type Selector that automatically predicts beneficial paraphrase types for each problem. This selector consistently improves performance over the baseline across models and datasets. For example, on MATH-500 with LLaMA 3.1-8B, combining the original with the best five paraphrased versions improves accuracy by **+8.4%**, and the selector achieves an additional **+1.33%** gain.

Acknowledgements

This work was supported by research fund of Chungnam National University, Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2022-00155857, Artificial Intelligence Convergence Innovation Human Resources Development (Chungnam National University)) and the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(No. RS-2025-00556217314820926401 01).

8 Limitaion

The Paraphrase Type Selector was trained solely on the MATH dataset, which may limit its ability to generalize to datasets with different styles or distributions.

In addition, our evaluation relied on the Eval-Harness framework, which measures only final answer correctness, without accounting for the quality of intermediate reasoning or partial solutions.

References

Josh Achiam et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Vansh Agrawal, Pratham Singla, Amitoj Singh Miglani, Shivank Garg, and Ayush Mangal. 2024. Give me a hint: Can llms take a hint to solve math problems? *arXiv preprint arXiv:2410.05915*.

Lukas Baumanns and Benjamin Rott. 2022. The process of problem posing: Development of a descriptive phase model of problem posing. *Educational Studies in Mathematics*, 110(2):251–269.

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.

Jinfa Cai et al. 2015. Problem-posing research in mathematics education: Some answered and unanswered questions. *Mathematical problem posing: From research to effective practice*, pages 3–34.

Constantinos Christou et al. 2005. An empirical taxonomy of problem posing processes. *Zdm*, 37:149–158.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

Jinlan Fu, Shenzhen Huangfu, Hang Yan, See-Kiong Ng, and Xipeng Qiu. 2024a. Hint-before-solving prompting: Guiding llms to effectively utilize encoded knowledge. *arXiv preprint arXiv:2402.14310*.

Junbo Fu et al. 2024b. Learning to paraphrase for alignment with llm preference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2394–2407.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. The language model evaluation harness.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2:1.

Aaron Grattafiori et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieval step by step for large language models. *arXiv preprint arXiv:2502.01142*.

Dan Hendrycks et al. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Pengfei Hong, Navonil Majumder, Deepanway Ghosal, Somak Aditya, Rada Mihalcea, and Soujanya Poria. 2024. Evaluating llms' mathematical and coding competency through ontology-guided interventions. *arXiv preprint arXiv:2401.09395*.

Shifeng Huang, Jiawei Wang, Jiao Xu, Da Cao, and Ming Yang. 2021. Recall and learn: A memory-augmented solver for math word problems. *arXiv preprint arXiv:2109.13112*.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Huiyuan Lai, Xiao Zhang, and Malvina Nissim. 2025. Multidimensional consistency improves reasoning in language models. *arXiv preprint arXiv:2503.02670*.

Zachary Levonian, Chenglu Li, Wangda Zhu, Anoushka Gade, Owen Henkel, Millie-Ellen Postle, and Wanli Xing. 2023. Retrieval-augmented generation to improve math question-answering: Trade-offs between groundedness and human preference. *arXiv preprint arXiv:2310.03184*.

Honglin Lin et al. 2025. Metaladder: Ascending mathematical solution quality via analogical-problem reasoning transfer. *arXiv preprint arXiv:2503.14891*.

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

Chengwei Qin et al. 2024. Relevant or random: Can llms truly perform analogical reasoning? *arXiv preprint arXiv:2404.12728*.

Florence Mihaela Singer et al. 2015. Mathematical problem posing. *New York: Springer. doi*, 10:978–1.

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*.

Hemish Veeraboina. 2023. Aime problem set 1983-2024.

Jan Philip Wahle, Bela Gipp, and Terry Ruas. 2023. Paraphrase types for generation and detection. *arXiv preprint arXiv:2310.14863*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

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.

Vikas Yadav, Zheng Tang, and Vijay Srinivasan. 2024. Paraphrase and aggregate with large language models for minimizing intent classification errors. *arXiv preprint arXiv:2406.17163*.

An Yang et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Michihiro Yasunaga et al. 2023. Large language models as analogical reasoners. *arXiv preprint arXiv:2310.01714*.

Dharunish Yugeswardeenoo, Kevin Zhu, and Sean O'Brien. 2024. Question-analysis prompting improves llm performance in reasoning tasks. *arXiv preprint arXiv:2407.03624*.

Yu Zhang, Shujun Peng, Nengwu Wu, Xinhan Lin, Yang Hu, and Jie Tang. 2025. Rm-pot: Reformulating mathematical problems and solving via program of thoughts. *arXiv preprint arXiv:2502.12589*.

Yue Zhou et al. 2024. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. *arXiv preprint arXiv:2404.11500*.

A Experiments Setup

In this study, experiments were conducted on a high-performance computing system equipped with a 104-core CPU, 4 NVIDIA V100 GPUs (32GB each), 256GB of RAM, and 33TB of HDD storage. This infrastructure provided the necessary computational resources for executing model inference, evaluation, and data processing. Additionally, for models such as GPT-4o, we used the OpenRouter API to conduct inference.

B Prompts used for Paraphrasing

Table 5 summarizes the twelve paraphrasing types with their corresponding prompt descriptions, clarifying how each paraphrase was operationalized in our study.

C Examples of Paraphrasing Types

Table 6 presents examples of the twelve paraphrasing types, each applied to the same math problem to demonstrate how the original is transformed.

D Verification of Paraphrased Problems

To ensure the quality and validity of the paraphrased problems, we conducted a two-stage verification process involving both human reviewers and GPT-4.1-Mini.

A total of 200 paraphrased problems covering all twelve paraphrase types were independently evaluated. For each problem, both annotators provided:

- Binary judgment on logical consistency (Yes/No)
- Binary judgment on technique match (Yes/No)
- A brief justification

In the initial evaluation, the agreement rate between the two annotators was approximately **89%**. After refining the evaluation prompt to address common sources of error, the same set of 200 samples was re-evaluated, producing:

- **94%** agreement on logical consistency
- **89%** agreement on technique match
- **92%** overall agreement

For cases where GPT-4.1-Mini's justifications were insufficient or ambiguous, the final judgment was always made by the human reviewer. If a paraphrase failed either criterion, up to three regeneration attempts were allowed. When still unsuitable, the paraphrase was manually rewritten by a human annotator with a Master's degree or higher in Computer Science.

E Additional Experiments

E.1 Random vs. Structured Paraphrase Selection

We investigated whether structured paraphrase sets, defined by our 12 paraphrase types, offer advantages compared to randomly selected paraphrases. The following settings were evaluated on the MATH-500 dataset:

- **Q + Random 5**: Original problem plus 5 randomly sampled paraphrases.
- **Q + Avg Top 5**: Original problem plus the 5 types with the highest global average performance.

Paraphrase Type	Prompt Description
Mathematical Expression Transformation	Modify mathematical expressions while preserving their meaning.
Text Reduction	Remove unnecessary details while retaining key information.
Text Rewriting	Rephrase the problem using different linguistic styles (formal vs. informal).
Sentence Order Modification	Change the order of the given statements while keeping the problem intact.
Sentence Merging	Combine multiple sentences into a single, coherent statement.
Metaphorical Representation	Use analogies or metaphors to describe the problem more intuitively.
Syntactic Transformation	Change the sentence structure while maintaining the original meaning.
Object Substitution	Do not change any numerical values, expressions, or mathematical structure. Replace objects or subjects (e.g., people, items, context) with equivalent ones.
Variable Renaming	Change variable names without altering the logic of the problem.
Contextual Rewriting	Change the context or background setting of the problem while keeping its logic intact.
Step-by-Step Breakdown	Decompose a complex problem into multiple smaller steps.
Information Segmentation	Split a single sentence into multiple statements for clarity.

Table 5: Prompts for the 12 paraphrase types used in our study.

Paraphrase Type	Example
Original	How many ounces of pure water must be added to \$30\$ ounces of a \$30%\$ solution of acid to yield a solution that is \$20%\$ acid?
Mathematical Expression Transformation	How many ounces of pure water need to be mixed with 30 ounces of a solution that contains 30 acid in order to produce a solution that has an acid concentration of 20?
Text Reduction	How many ounces of pure water must be added to \$30\$ ounces of a \$30%\$ acid solution to make it \$20%\$ acid?
Text Rewriting	**Formal:** Determine the quantity of pure water, measured in ounces, that needs to be incorporated into a \$30\$ ounce solution containing \$30%\$ acid in order to produce a resultant solution that contains \$20%\$ acid. **Informal:** How much pure water do you need to mix with \$30\$ ounces of a \$30%\$ acid solution to make it a \$20%\$ acid solution?
Sentence Order Modification	To yield a solution that is \$20%\$ acid, how many ounces of pure water must be added to \$30\$ ounces of a \$30%\$ solution of acid?
Sentence Merging	How many ounces of pure water must be added to \$30\$ ounces of a \$30%\$ acid solution in order to create a new solution that has an acid concentration of \$20%\$?
Metaphorical Representation	Imagine you have a large pot filled with a thick soup that is 30% spice. The pot contains 30 ounces of this spicy soup. You want to create a milder version of the soup that only has 20% spice. To achieve this, you need to add pure water to the pot. How many ounces of pure water should you pour in to dilute the spicy soup down to the desired flavor?
Syntactic Transformation	What is the amount of pure water, in ounces, that needs to be added to \$30\$ ounces of a \$30%\$ acid solution in order to create a solution that has an acid concentration of \$20%\$?
Object Substitution	How many ounces of pure juice must be added to \$30\$ ounces of a \$30%\$ mixture of syrup to yield a mixture that is \$20%\$ syrup?
Variable Renaming	How many ounces of pure liquid must be added to \$30\$ ounces of a \$30%\$ mixture of acid to yield a mixture that is \$20%\$ acid?
Contextual Rewriting	In a laboratory, a chemist has \$30\$ ounces of a \$30%\$ saline solution. To dilute this solution to a concentration of \$20%\$ saline, how many ounces of pure water must be added?
Step-by-Step Breakdown	1. Calculate how much acid is in 30 ounces at 30%. 2. Set up the equation with unknown x added. 3. Solve for x.
Information Segmentation	We have 30 ounces of solution. It is 30% acid. We want 20% acid. How much water must we add?

Table 6: Examples of 12 paraphrase types applied to the same math problem.

Setting	LLaMA-8B	LLaMA-70B	Qwen-7B	Qwen-72B	GPT-4o
Q + Random 5	43.80 ($\pm$2.28)	69.47 ($\pm$1.16)	67.67 ($\pm$1.34)	75.33 ($\pm$0.38)	66.80 ($\pm$0.17)
Q + Avg Top 5	43.80 ($\pm$1.56)	69.87 ($\pm$0.34)	**69.93** ($\pm$1.20)	76.87 ($\pm$0.09)	66.40 ($\pm$0.33)
Q + Best Top 5	**45.13** ($\pm$0.62)	**70.67** ($\pm$0.19)	68.60 ($\pm$0.59)	**77.27** ($\pm$1.00)	**67.13** ($\pm$0.09)

Table 7: Comparison of paraphrase selection strategies on MATH-500. Results are accuracy (%) with standard deviation. The best value for each model is highlighted in Blue.

Paraphrasing Type	Q + 1	Q + 2	Q + 3	Q + 4	Q + 5
Contextual Rewriting	38.20 (±1.91)	37.20 (±1.22)	37.53 (±0.83)	37.00 (±2.31)	**39.27** (±2.10)
Information Segmentation	35.53 (±1.89)	38.80 (±0.20)	37.67 (±1.85)	**39.07** (±2.66)	38.60 (±1.04)
Mathematical Expression	**40.20** (±1.56)	38.27 (±1.81)	38.93 (±1.92)	39.47 (±1.81)	37.53 (±1.01)
Metaphorical Representation	35.40 (±1.31)	36.33 (±0.76)	36.80 (±1.06)	**37.00** (±0.20)	35.33 (±1.40)
Object Substitution	**37.40** (±1.64)	37.00 (±1.31)	36.13 (±2.66)	36.13 (±2.10)	36.00 (±0.53)
Sentence Merging	38.00 (±0.80)	39.27 (±1.68)	39.47 (±0.90)	39.73 (±0.61)	**39.80** (±0.53)
Sentence Order Modification	37.40 (±1.44)	39.27 (±1.86)	**40.53** (±1.62)	38.80 (±2.50)	38.73 (±1.75)
Step-by-Step Breakdown	33.67 (±0.83)	37.60 (±0.35)	**38.40** (±0.80)	36.00 (±1.11)	37.00 (±1.11)
Syntactic Transformation	38.80 (±0.80)	38.33 (±1.03)	**38.93** (±2.04)	37.80 (±1.25)	38.87 (±0.81)
Text Reduction	39.13 (±0.64)	38.47 (±1.45)	38.47 (±1.17)	**39.33** (±0.31)	39.27 (±0.31)
Text Rewriting	38.73 (±1.29)	39.00 (±2.26)	39.27 (±1.33)	**40.33** (±1.21)	39.60 (±1.20)
Variable Renaming	36.27 (±2.42)	**37.67** (±0.76)	36.40 (±0.53)	37.40 (±1.11)	36.73 (±1.17)

Table 8: Performance (Mean ± Std, %) of different paraphrasing types when incrementally adding 1–5 paraphrases of the same type. The best result per row is highlighted in blue.

- **Q + Best Top 5**: Original problem plus the 5 most effective types selected per problem.

Table 7 shows the results. Overall, structured paraphrase sets tend to outperform random combinations, and the Best Top 5 setting provides the strongest upper bound across models.

E.2 Controlled Experiment on Paraphrase Quantity

To disentangle the effects of paraphrase quantity and type diversity, we conducted a controlled experiment in which multiple paraphrases were added from the *same* paraphrasing technique. The experiment was performed on the MATH-500 dataset using the LLaMA-3.1-8B-Instruct model, and each configuration was executed three times.

As shown in Table 8, adding paraphrases from the same technique does not guarantee monotonically increasing gains, and peak performance tends to occur around Q+2–Q+4 depending on the type. For instance, *Text Rewriting* achieves its best performance when four paraphrases are added. *Text Reduction* exhibits a similar upward trend followed by a slight decline. *Sentence Order Modification* reaches its peak when three paraphrases are added, after which accuracy decreases. *Mathematical Expression* shows its best performance with a small number of additions (40.2 at Q+1), with diminishing returns as more are introduced. *Object Substitution* also saturates relatively early. Overall, these results suggest that, rather than simply increasing the number of paraphrases from a single technique, it is more effective to carefully select and combine high-impact types.