Evaluating Robustness of LLMs to Numerical Variations in Mathematical Reasoning

Yuli Yang
Institute of Science Tokyo
yang.y.aw@m.titech.ac.jp

Hiroaki Yamada
Institute of Science Tokyo
yamada@comp.isct.ac.jp

Takenobu Tokunaga
Institute of Science Tokyo
take@c.titech.ac.jp

Abstract

Evaluating an LLM's robustness against numerical perturbation is a good way to know if the LLM actually performs reasoning or just replicates patterns learned. We propose a novel method to augment math word problems (MWPs), producing numerical variations at a large scale utilizing templates. We also propose an automated error classification framework for scalable error analysis, distinguishing calculation errors from reasoning errors. Our experiments using the methods show LLMs are weak against numerical variations, suggesting they are not fully capable of generating valid reasoning steps, often failing in arithmetic operations.

1 Introduction

Recent LLMs (Achiam et al., 2023; Dubey et al., 2024; Team et al., 2023, 2024) have reported high accuracy rates on mathematical reasoning benchmarks, including GSM8K and MATH (Cobbe et al., 2021; Hendrycks et al., 2021). However, a natural concern is that the models just follow surface patterns observed in their pretraining data rather than performing mathematical reasoning (Levy et al., 2024; Valmeekam et al., 2024a,b; Jiang et al., 2024; Guo et al., 2024).

Perturbing superficial elements like names of individuals or specific numbers does not change how the problem should be solved. If an LLM can perform reasoning in solving a math question, it should give correct answers with similar reasoning steps for both the original and its perturbed one. Recent studies (Srivastava et al., 2024; Qian et al., 2024; Li et al., 2024; Mirzadeh et al., 2024) evaluated models' robustness against the perturbations based on this hypothesis.

These studies have the following limitations: a) the size of the introduced variations was limited, b) they did not discuss ranges of numerical values such as digit sizes, and c) they did not distinguish reasoning errors and computational errors and could not explain the source of errors.

To address the limitations, we propose a scalable method to augment a math word problem (MWP) dataset by changing numerical values based on template questions. To analyze the impact of digit sizes on models' mathematical reasoning, we generate two distinct subsets by controlling the range of the replaced values, one with questions containing a small number of digits and the other with questions containing a large number of digits. We construct a new dataset, GSM-ALT, generating 2,000 variants for each original question from GSM8K. Moreover, we propose a novel framework for automated error analysis to distinguish two sources of errors: logical reasoning and numerical calculation.

2 Related Work

Despite strong performance on math benchmarks, researchers are questioning whether current benchmarks can adequately evaluate the reasoning abilities of language models.

Levy et al. (2024) expanded questions by adding non-essential contents, showing that models' performance decreases when the number of tokens in a problem increases. PlanBench (Valmeekam et al., 2024a,b) is a benchmark to evaluate planning and reasoning capabilities. Their findings suggest that even the state-of-the-art models still struggle with this. Srivastava et al. (2024) functionalized the math questions to create a dynamic dataset, providing a robust evaluation metric against potential data leakage to models' pretraining. Jiang et al. (2024) demonstrated that the models' high accuracy depends on a specific token bias, and the models' reasoning capability depends on recognizing certain superficial patterns. Berglund et al. (2024) and Guo et al. (2024) constructed reversal versions of the original questions and showed that the current models performed poorly on the reversal ones.

The Sixth Workshop on Insights from Negative Results in NLP, pages 171–180

3 Method

3.1 Question Template Development

To develop a new dataset to assess the model's robustness against numerical variations, we manually generate new variants based on templates (Figure 1) composed from the questions of an existing dataset[1].

A question from an existing dataset (e.g., GSM8K) has tuples of (question Q, solution S). Q is a natural language text describing a question to be solved. S contains the process P and the final answer A. P shows a gold process for solving the question Q step by step, including equations. A stores a numerical value as a gold outcome from the P. Given (Q, S), we first replace all the numerical values in the Q with variables to get Q_{abs}, which is the abstracted Q. We apply the same operation to S and get S_{abs}. We keep variables consistent between Q_{abs} and S_{abs}. S_{abs} contains P_{abs} and A_{abs}, representing the abstracted P and A. The Q_{abs} and S_{abs} constitute a question template T.

3.2 Variant Set Generation

Given a template T of an original question, we generate variants by replacing the variables in T with random values. t_i denotes a variant generated from T, consisting of question Q_i, solution S_i. S_i contains the process P_i and final answer A_i. To ensure variant validity, we must replace the values with satisfying some constraints (Figure 1). For example, an answer should be positive and whole when it represents the number of objects. Intermediate values appearing in the process P_i also need to satisfy the constraints as well. We manually define constraints for each template. Variants are accepted only if they satisfy the constraints.

Suppose models conduct only superficial pattern-based inference instead of reasoning. In that case, they perform poorly in solving questions containing numbers that are rare in their training, such as large digit numbers. To examine this hypothesis, we control the replaced values within two ranges for each question template, to obtain two variant sets: 1-99 (namely, the Easy variant set) and 1-9,999 (namely, the Hard variant set).

4 Experimental Settings

We use GSM8K as the base dataset for our experiment. GSM8K consists of MWPs for primary and secondary school students and involves only the four basic arithmetic operations.

We randomly sample 250 questions from the GSM8K training set to create 250 question templates manually. Given the templates, we generate 1,000 hard variants and 1,000 easy variants for each template. As a result, our new dataset GSM-ALT consists of the Hard and Easy variant set, each containing 250,000 variants.

We use accuracy as a primary evaluation metric. For the original instances from the base dataset (original GSM8K), we use a standard accuracy. For generated variants from our dataset, we first calculate the accuracy for each template variant set containing 1,000 variants, and then we average them over all the 250 templates.

The target models to be evaluated include generic models (Llama-3-8b-Instruct, Llama-3.1-8b-Instruct, Llama-3.1-70b-Instruct, Mistral-7b-Instruct-v0.3) and math models that were fine-tuned on mathematical contents (Deepseekmath-7b-rl, Wizardmath-7b-v1.1).

Regarding the generation settings, we use greedy search to maximize the reproducibility and stability of results. To minimize the influence of few-shot examples while ensuring that the model can perform mathematical reasoning, we adopt the zero-shot CoT prompting for solution generation and extract the final answer in the same way as Kojima et al. (2022) for generic models. As for math models, we adopt the specifically designed prompts, which are recommended on their Web pages. The prompts used in the experiment can be found in Appendix E.2.

Models	GSM8K	GSM-ALT	
	Base	Easy	Hard
Llama-3-8b-Instruct	0.880	0.646	**0.289**
Llama-3.1-8b-Instruct	0.908	0.736	**0.345**
Llama-3.1-70b-Instruct	0.972	0.888	**0.521**
Mistral-7b-Instruct-v0.3	0.620	0.373	**0.194**
Deepseek-math-7b-rl	0.964	0.808	**0.467**
Wizardmath-7b-v1.1	0.868	0.629	**0.347**

Table 1: Accuracy scores

5 Results

Table 1 shows the results of each model's accuracy evaluated on the original GSM8K, and our GSM-ALT. The lowest scores are highlighted in boldface. GSM-ALT results show scores from the Easy variant set and the Hard variant set. All models showed

[1]Appendix I discusses automation of this process.

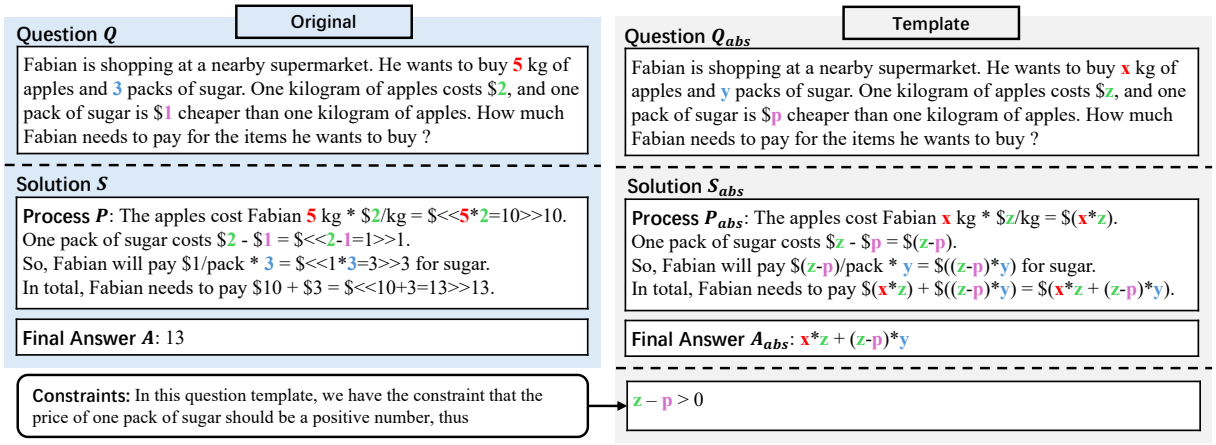

Figure 1: Example of Question Template Development

a significant performance drop in GSM-ALT from the base GSM8K. The drop was observed in both the Easy and the Hard variant sets. Even the two math-specialized models, especially Wizardmath-7b-v1.1, showed lower scores by more than 0.2 on the Easy and more than 0.5 on the Hard.

This result shows that numerical variations always degrade performance in both the Hard and the Easy variant sets. The fact that the Easy variant set degrades the performance indicates that the models are weak even against the numbers whose range is similar to the base GSM8K. Moreover, we found clearer score drops from the GSM8K scores in the Hard variant set than in the Easy variant set, suggesting the computational difficulty affects models' reasoning.

6 Error Analysis on Solutions

To identify the source of errors, we classify errors into two types: calculation errors and reasoning errors. If an incorrect solution only contains failures in calculations, we call it a calculation error. If an incorrect solution contains incorrect reasoning steps, we label it a reasoning error regardless of its incorrect calculations.

As GSM-ALT will be larger than its original dataset, manually checking each generated solution is not practical, and thus, we propose a novel framework that automatically classifies errors into calculation or reasoning errors.

6.1 Error Analysis Framework

To classify errors, we first transform a predicted solution $\hat{S}_i$ into its abstracted form $\hat{S}^i_{abs}$, which contains the abstracted $\hat{P}^i_{abs}$ and $\hat{A}^i_{abs}$. If $\hat{S}_i$ is incorrect because of a reasoning error, its transformed $\hat{P}^i_{abs}$ should contain a reasoning error resulting in incorrect $\hat{A}^i_{abs}$. If $\hat{S}_i$ contains a calculation error with correct reasoning steps, $\hat{P}^i_{abs}$ and $\hat{A}^i_{abs}$ should be correct. Thus, checking if $\hat{A}^i_{abs}$ is correct should give a proxy to determine the sources of errors.

In our framework, an LLM transforms a $\hat{S}_i$ into the $\hat{S}^i_{abs}$, as shown in Figure 2. Then, we can automatically check if $\hat{A}^i_{abs}$ is correct by comparing it with its gold answer A_{abs} from our templates. An input to the LLM is a model's predicted solution $\hat{S}_i$, its question Q_i, and its abstracted question Q_{abs} available from our templates. We auxiliaryly input the Q_{abs} guiding the LLM to use variables consistently, inspired by Gaur and Saunshi (2023). An output from the LLM is an abstracted solution $\hat{S}^i_{abs}$. We show our prompt for this framework in Appendix G. We employ Qwen2-math-72b-instruct (Yang et al., 2024) for this transformation.

We confirmed the LLM could obtain the abstracted solutions at 90% success rate on average in our preliminary experiment (Appendix F).

6.2 Results

Table 2 shows the results of error classification by our framework. Values in the table indicate the proportion of solutions classified as calculation errors or reasoning errors out of all solutions predicted by the models. Values in parentheses indicate the proportion of solutions classified as calculation errors or reasoning errors out of incorrect solutions.

In the Base set, most incorrect solutions are due to reasoning errors. However, they changes to calculation errors in the Easy and Hard variant sets except for the Mistral. This trend is especially evident in the Hard variant set, where more than 69% come from calculation errors. This result suggests

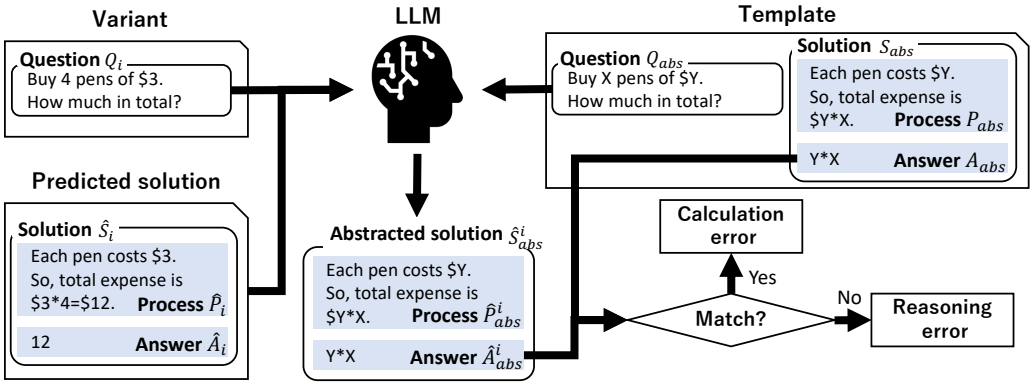

Figure 2: Error classification framework

	Base set		Easy variant set		Hard variant set	
	calculation err.	reasoning err.	calculation err.	reasoning err.	calculation err.	reasoning err.
Llama-3-8b-Inst.	.024 (20.0%)	.096 (80.0%)	.185 (52.3%)	.169 (47.7%)	.491 (69.1%)	.220 (30.9%)
Llama-3.1-8b-Inst.	.020 (21.7%)	.072 (78.3%)	.153 (57.9%)	.111 (42.1%)	.487 (74.4%)	.168 (25.6%)
Llama-3.1-70b-Inst.	.012 (42.9%)	.016 (57.1%)	.070 (62.5%)	.042 (37.5%)	.399 (83.3%)	.080 (16.7%)
Mistral-7b-Inst.-v0.3	.096 (25.3%)	.284 (74.7%)	.279 (44.5%)	.348 (55.5%)	.406 (50.4%)	.400 (49.6%)
Deepseek-math-7b-rl	.012 (33.3%)	.024 (66.7%)	.131 (68.2%)	.061 (31.8%)	.413 (77.5%)	.120 (22.5%)
Wizardmath-7b-v1.1	.032 (24.2%)	.100 (75.8%)	.255 (68.5%)	.117 (31.5%)	.489 (74.9%)	.164 (25.1%)
Macro avg.	.033 (25.0%)	.099 (75.0%)	.179 (59.0%)	.141 (41.0%)	.448 (71.6%)	.192 (28.4%)

Table 2: Error rate per error type and variant set

that the limited capability of arithmetic calculation is indeed a major issue of LLMs in solving mathematical problems rather than the reasoning capability of generating a valid process of solving steps when the numerical values in the questions are large.

Looking at the reasoning errors, all the models get more errors in both the Easy and Hard variant sets than the Base set. The same as calculation errors, the trend is evident in the Hard variant set. This result suggests that variants also introduce harmful changes in reasoning steps in addition to complex calculations, which result in incorrect solutions. Moreover, variants with larger digit sizes are more likely to introduce errors in reasoning steps.

Mistral shows different behavior from the others because its performance is considerably worse (Table 2). Mistral is the worst in mathematical reasoning. Note that a calculation error is considered only when a solution has valid reasoning steps. Thus, a lower number of calculation errors does not necessarily mean Mistral is good at calculation.

7 Conclusion

We proposed a novel method to augment MWP datasets, which produces a dataset for evaluating LLMs' robustness against numerical variations at a large scale. Using our templates, anyone can easily generate thousands of variants from one original question in the GSM8K, which was not possible with any preceding proposals. We also proposed an automated error classification framework for detailed error analysis, distinguishing calculation errors from reasoning errors.

Using the methods, we empirically showed that the six LLMs we tested were weak against numerical variations, especially when the numerical values were large. This finding is consistent with previous studies (Srivastava et al., 2024; Qian et al., 2024; Li et al., 2024; Mirzadeh et al., 2024), but we confirm it with more variants. Our error analysis uniquely identified that calculation errors contributed to a substantial proportion of incorrect solutions, suggesting LLMs' incapability of arithmetic operations is the main source of limited capabilities in math word problems. Moreover, we found that LLMs still fail in their reasoning steps, especially when they encounter variants with larger numerical values. Given our findings, it is still hard to say that current LLMs are robust against numerical variations.

8 Limitations

Variant Generation As we do not manually check every single variant generated from our template, our variants can contain some combinations of numbers that are not necessarily realistic in our real-world common sense. For example, there might be an apple that sells for more than 1,000 dollars. In our experiment, we accepted such variants. The reasons we made the compromise are: a) In some question templates, it is hard to generate enough variants if we strictly follow common sense. b) We believe that in the background of mathematical reasoning, models should strictly follow the conditions stated in the questions instead of common sense. c) Through a check on sampled solutions generated by models, we found that common sense issues will not affect the model's reasoning in most cases.

Another limitation is that the number of possible variants is limited in some of the templates, so there are some duplicated variants for such a template. This limitation is especially evident in the Easy variant set, where the range of numerical variation is more limited than the Hard variant set. We present the full results in the main body of the paper and put the results of the templates with no duplications in Appendix A.

Error Analysis Framework When we manually inspected the generated solutions, we found different reasoning patterns depending on the numerical variations. Moreover, we also observe different groups of reasoning patterns specific to one of the sets (Easy or Hard)[2].

Our error analysis framework is based on a classification approach, distinguishing between calculation errors and reasoning errors. Thus, it cannot identify what kind of failures happen within the reasoning errors. We also could not identify any reasonable clusters and their interpretations because it required a lot of human resources. It is an interesting future direction to extend our automatic error analysis framework, enabling it to cluster and aggregate the reasoning patterns, mitigating the limitation of our classification-based analysis.

Acknowledgements

This work was partly supported by JST, PRESTO Grant Number JPMJPR236B, Japan. The experiments were carried out using the TSUBAME4.0 supercomputer at Institute of Science Tokyo.

[2]Appendix H discusses it

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The reversal curse: Llms trained on "a is b" fail to learn "b is a". *Preprint*, arXiv:2309.12288.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Vedant Gaur and Nikunj Saunshi. 2023. Reasoning in large language models through symbolic math word problems. *Preprint*, arXiv:2308.01906.

Pei Guo, WangJie You, Juntao Li, Yan Bowen, and Min Zhang. 2024. Exploring reversal mathematical reasoning ability for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13671–13685, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo Jose Taylor, and Dan Roth. 2024. A peek into token bias: Large language models are not yet genuine reasoners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4722–4756, Miami, Florida, USA. Association for Computational Linguistics.

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

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 15339–15353, Bangkok, Thailand. Association for Computational Linguistics.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Kun Qian, Shunji Wan, Claudia Tang, Youzhi Wang, Xuanming Zhang, Maximillian Chen, and Zhou Yu. 2024. VarBench: Robust language model benchmarking through dynamic variable perturbation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16131–16161, Miami, Florida, USA. Association for Computational Linguistics.

Saurabh Srivastava, Annarose M B, Anto P V, Shashank Menon, Ajay Sukumar, Adwaith Samod T, Alan Philipose, Stevin Prince, and Sooraj Thomas. 2024. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *Preprint*, arXiv:2402.19450.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1218–1236, Miami, Florida, US. Association for Computational Linguistics.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2024a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024b. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *Preprint*, arXiv:2409.13373.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

A Results of the templates with no duplications

Models	GSM8K	GSM-ALT	
	Base	Easy	Hard
Llama-3-8b-Instruct	0.840	0.507	**0.156**
Llama-3.1-8b-Instruct	0.880	0.604	**0.193**
Llama-3.1-70b-Instruct	0.978	0.819	**0.355**
Mistral-7b-Instruct-v0.3	0.587	0.238	**0.104**
Deepseek-math-7b-rl	0.957	0.706	**0.307**
Wizardmath-7b-v1.1	0.891	0.489	**0.223**

Table 3: Accuracy scores (for 92 templates without duplications)

As explained in Section 8, duplicated variations might be generated for some templates. We counted the number of duplicated variants generated per template. We found that most templates have a small number of duplications, and only 92 templates (in 250 templates) contain no duplicated variants. To make our conclusions more rigorous, we removed those templates with duplications and presented the results of the remaining templates here (Table 3 and Table 4). According to the filtered results, we could get the same findings and conclusions as in Section 5 and Section 6.2.

B Dataset Lisence

The original GSM8K was distributed under the MIT license. We plan to make our templates publicly available under the MIT License.

C Computational Environment

All of our experiments were conducted on a GPU server implementing AMD EPYC 9654 2.4GHz × 2 Socket, 768GiB RAM, NVIDIA H100 SXM5 94GB HBM2e × 4. Our project's total hours spent on the server were approximately 480 hours, including preliminary experiments.

D Large Language Models

We list all of the LLMs used in our experiments.

	Base set		Easy variant set		Hard variant set	
	calculation err.	reasoning err.	calculation err.	reasoning err.	calculation err.	reasoning err.
Llama-3-8b-Inst.	.033 (20.0%)	.130 (80.0%)	.279 (56.6%)	.214 (43.4%)	.573 (67.9%)	.271 (32.1%)
Llama-3.1-8b-Inst.	.033 (27.3%)	.087 (72.7%)	.252 (63.6%)	.144 (36.4%)	.601 (74.5%)	.206 (25.5%)
Llama-3.1-70b-Inst.	.000 (00.0%)	.022 (100.0%)	.125 (69.1%)	.056 (30.9%)	.516 (80.0%)	.129 (20.0%)
Mistral-7b-Inst.-v0.3	.098 (23.7%)	.315 (76.3%)	.385 (50.5%)	.377 (49.5%)	.477 (53.2%)	.419 (46.8%)
Deepseek-math-7b-rl	.011 (25.0%)	.033 (75.0%)	.217 (73.8%)	.077 (26.2%)	.529 (76.4%)	.163 (23.6%)
Wizardmath-7b-v1.1	.043 (40.0%)	.065 (60.0%)	.383 (75.0%)	.128 (25.0%)	.586 (75.4%)	.191 (24.6%)
Macro avg.	.036 (24.8%)	.109 (75.2%)	.274 (62.3%)	.166 (37.7%)	.547 (70.4%)	.230 (29.6%)

Table 4: Error rate per error type and variant set (for 92 templates without duplications)

Generic LLMs

- Llama-3-8b-Instruct
  https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
- Llama-3.1-8b-Instruct
  https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
- Llama-3.1-70b-Instruct
  https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct
- Mistral-7b-Instruct-v0.3
  https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

LLMs for mathematical domain

- Deepseekmath-7b-rl
  https://huggingface.co/deepseek-ai/deepseek-math-7b-rl
- Wizardmath-7b-v1.1
  https://huggingface.co/WizardLMTeam/WizardMath-7B-V1.1

E Prompt Design

E.1 Zero-shot or Few-shot?

We used zero-shot CoT prompting to generate the solutions. The reasons are as follows: a) In our preliminary experiments, we found that few-shot examples do not necessarily improve the model's performance on mathematical reasoning, sometimes even degrading the accuracy. This might be because today's LLMs have already been sufficiently trained on similar MWPs, so providing random few-shot examples is just a kind of constraint and hinders the model's reasoning (Tam et al., 2024). b)The influence of few-shot examples on the model's reasoning is difficult to assess. For example, if two models perform differently under the same prompt and few-shot examples, it might be because this set of examples works for one model but not for the other. Therefore, we consider that few-shot CoT is not suitable for evaluation.

Generation Prompt – generic models

SYSTEM: You are an assistant that solves math word problems.

USER: {question} + Let's think step by step.

Figure 3: The prompt for generic models (generating solutions)

Answer Extraction Prompt – generic models

SYSTEM: You are an assistant that solves math word problems.

USER: {question} + Let's think step by step.
ASSISTANT: {model's completion}

USER: Therefore, what is the final answer? Only write the final answer without any texts.

Figure 4: The prompt for generic models (extracting final answer)

E.2 Prompts used for main experiments

For the generic LLMs, we developed prompts for solution generation (Figure 3) and answer extraction (Figre 4) based on the prompts used in Kojima et al. (2022).

For Deepseekmath-7b-rl and Wizardmath-7b-v1.1, we employed prompts based on templates suggested on their web pages. Figure 5 and 6 show them. In extracting answers from solutions generated by the two math models, we could simply use regular expressions since they always generate solutions in a fixed format.

F Performance of Transformation to Abstracted Solutions

To validate the performance of our error analysis framework, we had to know how well Qwen2-math-72b-Instruct could correctly transform a model's

	Error Types				Overall	
	Calculation		Reasoning			
Llama-3-8b-Instruct	93%	(26/28)	68%	(15/22)	82%	(41/50)
Llama-3.1-8b-Instruct	95%	(38/40)	80%	(8/10)	92%	(46/50)
Llama-3.1-70b-Instruct	97%	(38/39)	91%	(10/11)	96%	(48/50)
Mistral-7b-Instruct-v0.3	100%	(23/23)	59%	(16/27)	78%	(39/50)
Deepseek-math-7b-rl	97%	(38/39)	91%	(10/11)	96%	(48/50)
Wizardmath-7b-v1.1	97%	(37/38)	83%	(10/12)	94%	(47/50)
Macro avg.	96%		79%		90%	

Table 5: Abstracted solution transformation success rate

Generation Prompt – Deepseekmath-7b-rl

USER: {question}
Please reason step by step and put your final answer within \boxed{}.

Figure 5: The prompt for Deepseekmath-7b-rl (geneart-ing solutions)

Generation Prompt – Wizardmath-7b-v1.1

USER: Below is an instruction that describes a task. Write a response that appropriately completes the request.
Instruction:
{question}
Response:
Let's think step by step.

Figure 6: The prompt for Wizardmath-7b-v1.1 (generat-ing solutions)

solution into an abstracted form.

For this preliminary experiment, we constructed a small dataset for each LLM we employed in our main experiment (Section 4). For each model, we randomly sampled 50 incorrect solutions obtained from the results of the main experiment and manually categorized them into calculation errors or reasoning errors. For the evaluation, we manually checked whether the LLM could correctly transform the models' original solutions into abstracted forms according to the following criteria: a) the abstracted process exactly reflects the reasoning process in the original solution, b) the variable assignment in the abstracted solution should be correct, c) the abstracted final answer correctly represents the abstracted process.

The results are shown in Table 5. The table shows the success rate (percentages) of Qwen2-math-72b-Instruct's transformation on solutions predicted by different models in terms of the error type we manually categorized. Fractions in parentheses indicate the specific number we checked (the numerator is the number of successful transformations, and the denominator is the total number). It

Transformation Prompt

SYSTEM: Given the numeric version of a math question and its solution as references, you are a helpful assistant designed to copy the numeric solution to get a solution to the symbolic version of that question.
Instructions:
- Symbolic solution should strictly copy the numeric solution no matter whether it is correct or not.
- After completion of the solution, output the final answer with "###". The final answer should be a sole mathematical expression represented by variables appear in the symbolic question.
- Mathematical expression in the symbolic solution should not be represented in the format of LaTeX.

{few-shot examples}

USER: {target solution}

Figure 7: The prompt for solution transformation

demonstrates that Qwen2-math-72b-Instruct can correctly obtain symbolic solutions and answers from 90% of the instances overall. Based on the results, we consider our method to be reasonably valid for our purpose.

G Prompt Design for Error Analysis Framework

Figure 7 presents the prompt used to transform a predicted solution into the abstracted form.

H Case Study in Error Analysis

Through our manual inspection of the generated solutions for variants generated from the template, we identified that there were groups of reasoning patterns due to numerical variations. Figure 8 shows an example of the reasoning patterns we observed. This example is taken from outputs from Llama-3.1-70b-Instruct. The solutions in the figure show the different typical reasoning patterns identified by the underlined fonts. The performance of the easy variants generated from the same template of the example was 66.6% of accuracy (666 correct solutions), and there were 334 incorrect solutions consisting of 286 calculation errors and 48 reasoning errors. For its Hard set variants, accuracy was at 0%. Its incorrect solutions consist of 145 calcu-

Example of Variant from Easy set

Question: At a laundromat, it costs $50 for a washer and a quarter for every 3 minutes in the dryer. Samantha does 23 loads of laundry in the wash and then divides all the washed clothes equally into 21 separate dryers. If she lets the dryers run for 28 minutes each, how much does she spend altogether?

Process: To find the total cost, let's break it down into steps.
1. First, we need to find the cost of washing 23 loads of laundry. **Since it costs $50 for a washer, the total cost for washing is 23 * $50.**
Total washing cost = 23 * $50 = $1150

2. Next, we need to find the cost of drying the clothes. Since Samantha divides the clothes equally into 21 separate dryers, each dryer will have 23 / 21 loads. However, the cost of drying is based on the number of dryers and the time they run, not the number of loads. So, we'll focus on the number of dryers and the time. Since there are 21 dryers, and each dryer runs for 28 minutes, we need to find the total number of 3-minute intervals for all the dryers. We can do this by dividing the total minutes by 3.
Total minutes = 21 dryers * 28 minutes = 588 minutes
Total 3-minute intervals = 588 minutes / 3 = 196 intervals
Since it costs a quarter for every 3 minutes, we multiply the total intervals by the cost per interval.
Total drying cost = 196 intervals * $0.25 = $49

3. Finally, we add the total washing cost and the total drying cost to find the overall cost. Total cost = Total washing cost + Total drying cost
Total cost = $1150 + $49
Total cost = $1199
So, Samantha spends $1199 altogether.

Example of Variant from Hard set

Question: At a laundromat, it costs $5274 for a washer and a quarter for every 2 minutes in the dryer. Samantha does 8762 loads of laundry in the wash and then divides all the washed clothes equally into 8968 separate dryers. If she lets the dryers run for 8314 minutes each, how much does she spend altogether?

Process: To find the total cost, we need to calculate the cost of the washer and the cost of the dryer separately, then add them together.
1. The cost of the washer: Since Samantha does 8762 loads of laundry, and it costs $5274 for a washer, **we can assume that the cost of the washer is a one-time payment.** So, the cost of the washer is $5274.

2. The cost of the dryer: Samantha divides the clothes into 8968 separate dryers. Each dryer runs for 8314 minutes. The cost of the dryer is a quarter for every 2 minutes.

First, let's find the total number of minutes for all dryers:
8968 dryers * 8314 minutes = 74,555,072 minutes

Since the cost is a quarter for every 2 minutes, we need to divide the total minutes by 2:
74,555,072 minutes / 2 = 37,277,536 quarters

Now, let's convert the quarters to dollars (1 dollar = 4 quarters):
37,277,536 quarters / 4 = 9,319,384 dollars

Now, we add the cost of the washer and the cost of the dryer:
$5274 + $9,319,384 = $9,324,658
So, Samantha spends $9,324,658 altogether

Figure 8: Example of typical patterns in evaluation datasets

lation errors and 855 reasoning errors. The variants of the template in the Hard variant set cause a clear reasoning degradation.

We observed similar phenomena across other models besides Llama-3.1-70b-Instruct. We hypothesize that the reason why numerical variation could cause such changes in reasoning patterns was in the sizes of digits. As variants in the Hard variant set have more digits, LLMs get questions with more tokens, which makes their output different.

I Necessity of Manual Operations in Creating Question Templates

Although we have considered using regular expressions and rule-based approaches to automate tem-

plate creation, they have the following problems: a) Not all numerical values in the original instance are "symbolizable." Some numbers in the instance are specific; altering them would make the instance ill-defined. b) As shown in Figure 1, when generating the template, it is necessary to keep the usage of variable consistent between Q_{abs} and S_{abs}. It is hard to catch the relationship with rule-based replacement and requires human insight. Therefore, we created the question templates manually.