

AUTOMATIC ROBUSTNESS STRESS TESTING OF LLMS AS MATHEMATICAL PROBLEM SOLVERS

Anonymous authors

Paper under double-blind review

ABSTRACT

Large language models (LLMs) have achieved distinguished performance on various reasoning-intensive tasks. However, LLMs might still face the challenges of robustness issues and fail unexpectedly in some simple reasoning tasks. Previous works evaluate the LLM robustness with hand-crafted templates or a limited set of perturbation rules, indicating potential data contamination in pre-training or fine-tuning datasets. In this work, inspired by stress testing in software engineering, we propose a novel framework, Automatic Robustness Checker (AR-CHECKER), to generate mathematical problem variants that maintain the semantic meanings of the original one but might fail the LLMs. The AR-CHECKER framework generates mathematical problem variants through multi-round parallel streams of LLM-based rewriting and verification. Our framework can generate benchmark variants dynamically for each LLM, thus minimizing the risk of data contamination. Experiments on GSM8K and MATH-500 demonstrate the strong performance of AR-CHECKER on mathematical tasks. We also evaluate AR-CHECKER on benchmarks beyond mathematics, including MMLU, MMLU-Pro, and CommonsenseQA, where it also achieves strong performance, further proving the effectiveness of AR-CHECKER.

1 INTRODUCTION

Large Language Models (LLMs) have seen rapid advancements in recent years and are becoming increasingly important in daily life and research (Brown et al., 2020; OpenAI, 2022; Anthropic, 2023; llama3, 2024). Numerous benchmarks (Chang et al., 2024) have been established to assess their performance on various tasks, such as mathematical problem solving (Cobbe et al., 2021), coding generation (Chen et al., 2021), and instruction following (Zhou et al., 2023). Although evaluation results on these benchmarks show that LLMs perform very well and even achieve human-level performance on many tasks, some studies (Zhu et al., 2023; Huang et al., 2024; Cheng et al., 2024) have found that LLMs might perform poorly on certain instructions or questions that humans believe to be simple. These observations indicate that LLMs have significant robustness issues.

Existing works (Shi et al., 2023; Li et al., 2024; Mirzadeh et al., 2024) focus on the evaluation of LLM robustness by creation of benchmark variants on tasks like math reasoning (Cobbe et al., 2021). They observe that existing LLMs can still fail on problems that are slightly modified from the original samples. These findings hint at potential data contamination in pre-training or fine-tuning datasets (Mirzadeh et al., 2024). However, previous works evaluate the LLM robustness by creating variants with template-based approaches (Shi et al., 2023;

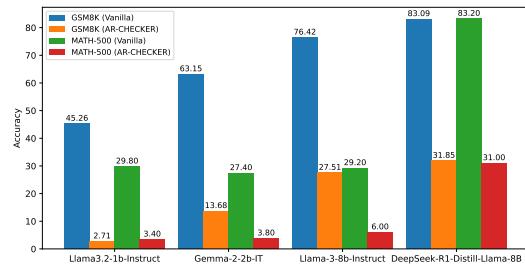


Figure 1: Accuracy comparison of various LLMs on GSM8K and MATH-500 benchmarks, showing performance before and after applying our AR-CHECKER robustness testing framework. The significant drop in accuracy after applying our method highlights the robustness issues of LLMs on mathematical problems and the unreliability of current evaluations.

Mirzadeh et al., 2024) or relying on a fixed set of perturbation rules (Li et al., 2024; Zhu et al., 2024). They underutilize the characteristics of the target LLMs during the benchmark creation process and fail to explore more diverse robustness challenges for the target LLMs.

In this work, inspired by stress testing in software engineering (Jiang & Hassan, 2015), we frame the robustness evaluation of LLMs as a stress testing problem and propose a novel framework, AR-CHECKER, for automatic robustness evaluation of LLMs as mathematical problem solvers. The framework consists of three LLMs: a rewriter, a verifier, and a target LLM. Given a testing budget of M , the rewriter rewrites mathematical problems into their variants, at most M times, while keeping the core meaning unchanged. The verifier checks whether the core meaning of the original question is preserved and whether the target LLM’s answer to the rewritten variant is different from the correct answer. The AR-CHECKER framework removes the reliance on hand-crafted guidelines to generate challenging instruction variants, instead, the rewriter explores to find the optimization direction via the iterations of multi-round parallel streams. Experiments on GSM8K (Cobbe et al., 2021) and MATH-500¹ prove the effectiveness and scalability of our proposed framework on various target LLMs. As shown in Figure 1, four LLMs obtain 48.04% and 31.35% less averaged accuracy on our GSM8K and MATH-500 evaluation variants, respectively. As the evaluation instructions are generated dynamically for each LLM, our framework is expected to minimize the risk of data contamination. We also evaluate AR-CHECKER on benchmarks beyond mathematics, including MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024), and CommonsenseQA (Talmor et al., 2019), where it also achieves strong performance, further proving the effectiveness of AR-CHECKER.

2 RELATED WORK

2.1 ROBUSTNESS EVALUATION OF LLMs

There has been a line of work on evaluating the robustness of LLMs. For instance, AdvGLUE (Wang et al., 2021) and Promptbench (Zhu et al., 2023) are two widely used static benchmarks to evaluate the robustness of LLMs on various tasks. As mathematical problems offer the advantage of having standard answers that enable objective evaluation, many previous works have focused on evaluating the robustness of LLMs in mathematical reasoning (Shi et al., 2023; Huang et al., 2025; Srivastava et al., 2024; Mirzadeh et al., 2024; Yu et al., 2025; Gulati et al., 2024). Shi et al. (2023) find that LLMs’ performance is dramatically decreased when irrelevant context is included in GSM8K problems. Li et al. (2024) propose a variant dataset of the GSM8K benchmark, GSM8K-PLUS, with various mathematical perturbations and show that LLMs are far from robust. Mirzadeh et al. (2024) create symbolic templates that allow for the generation of a diverse set of questions from GSM8K problems. They find that LLMs exhibit noticeable variance when responding to different instantiations of the same question. Unlike these works, which evaluate LLM robustness through template-based approaches or fixed sets of perturbation rules, our framework adopts a more flexible and dynamic optimization strategy.

2.2 LLM-BASED PROMPT OPTIMIZATION

Two lines of work on LLM-based prompt optimization are particularly relevant to our study. One line of work focuses on enhancing model performance through automatic prompt engineering (Yang et al., 2023; Zhou et al., 2022). These works, like ours, aim to optimize prompts while maintaining the core meaning, but they focus on improving the performance of the target LLMs, whereas we aim to find prompts that cause the target LLMs to fail. Another line of work is the jailbreak attack, which intentionally designs malicious prompts that adversarially trigger aligned LLMs to produce uncensored content (Dong et al., 2024; Xu et al., 2024; Yu et al., 2023; Ding et al., 2024). Prompt Automatic Iterative Refinement (PAIR) (Chao et al., 2023) employs an LLM-based attacker to reformulate malicious instructions and utilizes GPT-4 as an evaluator to assess the harmfulness of the target model’s responses. Different from the PAIR method, which attacks the LLMs to generate malicious content, our focus is specifically on evaluating the robustness of LLMs as mathematical problem solvers.

¹<https://huggingface.co/datasets/HuggingFaceH4/MATH-500>

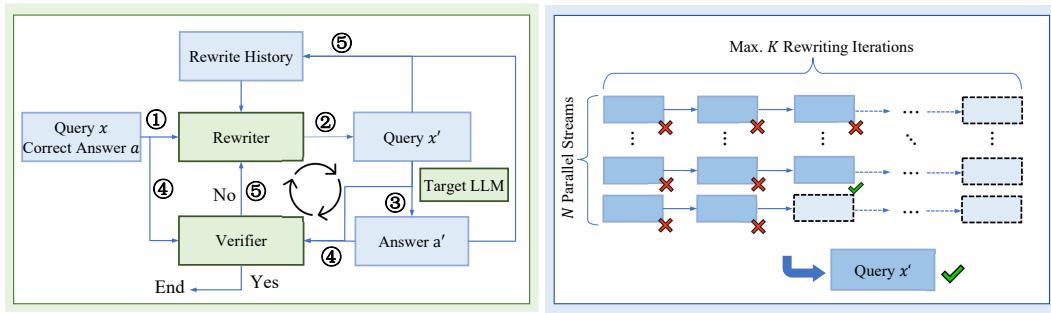


Figure 2: Overview of AR-CHECKER framework. Left: given a query x , AR-CHECKER iteratively rewrites x with the rewriter (Section 3.3) until the verifier (Section 3.4) detects a query variant x' that preserves the core mathematical meaning of the original query x but fails the target LLM. Right: for each query x , AR-CHECKER executes N parallel streams, allowing for a maximum K rewriting iterations per stream, enhancing both the search breadth and depth of the query variants.

3 METHODOLOGY

3.1 PROBLEM DEFINITION

Given a mathematical problem x and corresponding answer a from dataset D , our goal is to rewrite it into a variant x' within a given budget M such that x and x' have the same core meaning, and the target LLM's answer to x' , denoted as a' , is different from a . The budget M is defined as the maximum number of rewrites.

3.2 AR-CHECKER FRAMEWORK

Stress testing, a software testing technique, is used to evaluate the robustness of software systems by subjecting them to stressful conditions in order to uncover bugs, crashes, and potential vulnerabilities (Jiang & Hassan, 2015). In this work, inspired by stress testing, we propose a novel framework AR-CHECKER, to stress test the robustness of LLMs as mathematical problem solvers by automatically generating mathematical problem variants, as shown in Figure 2.

The framework consists of three components: a rewriter R , a verifier V , and a target LLM T , all of which are implemented using LLMs. Given a testing budget of $M = N * K$, the framework operates by running N parallel streams, each performing up to K rewriting iterations, as shown in Algorithm 1.

For each original mathematical problem x with correct answer a , the rewriter R generates variants x' while attempting to preserve the core mathematical meaning, which means the correct answer to x' should also be a . For each variant x' , the target LLM T produces an answer a' . The verifier V then evaluates whether: (1) the core meaning of x' matches that of x , and (2) the target LLM's answer a' differs from the correct answer a . A test is considered to pass only if the target LLM correctly answers all these variants.

Our AR-CHECKER framework incorporates two key designs that enhance its effectiveness in finding challenging variants: multi-round iteration and parallel streams. These complementary strategies work together to better explore the space of potential variants while enhancing both search depth and breadth, significantly improving the efficiency of identifying robust test cases:

Multi-round Iterations. The optimization of problem variants is formulated as a simulated multi-round chat between the rewriter and the verifier. Based on the previous problem variants and feedbacks from the verifier, the rewriter first generates chain-of-thoughts about the strategies to modify the instruction, and then proposes a novel and diverse variant of the problem. This multi-round iteration approach increases the search depth and encourages diverse proposals, which further challenges the robustness of the target LLM.

162

Algorithm 1 AR-CHECKER Framework

```

163 1: Input: Rewriter  $R$ , Verifier  $V$ , Target  $T$ , Original problem  $x$ , Correct answer  $a$ , Number of
164 2: Initialize: Successful variants set  $\mathcal{V} \leftarrow \emptyset$ 
165 3: for all stream  $n \in \{1, 2, \dots, N\}$  in parallel do
166 4:   Initialize conversation history  $C_n \leftarrow [(x, a)]$ 
167 5:   for  $k = 1$  to  $K$  do
168 6:      $x'_n \leftarrow R(x, C_n)$                                  $\triangleright$  Generate variant
169 7:      $a'_n \leftarrow T(x'_n)$                                  $\triangleright$  Query target LLM
170 8:      $v_n \leftarrow V(x, a, x'_n, a'_n)$                    $\triangleright$  Verify if meaning preserved and  $a'_n \neq a$ 
171 9:      $C_n \leftarrow C_n \cup \{(x'_n, a'_n, v_n)\}$            $\triangleright$  Update conversation history
172 10:    if  $v_n = D$  then                                 $\triangleright$  If meaning preserved and  $a'_n \neq a$ 
173 11:       $\mathcal{V} \leftarrow \mathcal{V} \cup \{(x'_n, a'_n)\}$            $\triangleright$  Add successful variant
174 12:    terminate all streams and return  $\mathcal{V}$   $\triangleright$  Stop all streams once a successful variant is
175 13:    found
176 14:  end if
177 15: end for
178 16: end for
179 17: return  $\mathcal{V}$                                  $\triangleright$  Return all successful variants
180
181
182

```

Parallel Streams. We design the framework to perform multiple streams of rewriting conversations in parallel, each of which is independent of the others. Specifically, the rewriter generates multiple variants in parallel, and any successful variant is considered a successful perturbation. This design increases the search width, allowing the rewriter to explore more possibilities and generate more effective adversarial questions, benefiting from different trajectories in different streams.

3.3 PROBLEM REWRITING

Given the strong instruction-following capability of current LLMs, we choose to use LLMs as problem rewriter. In the rewriter’s system prompt, the LLM role, background of the task, and detailed scenario are specified. Unlike prior work (Zhu et al., 2024), we do not provide specific rewriting methods or principles; instead, we encourage the rewriter LLM to explore and propose diverse rewriting suggestions for the user instruction based on its internal knowledge as well as feedbacks from the verifier. This self-evolvement approach is more scalable and flexible compared with existing template-based or rule-based methods. On one hand, as LLMs evolve, manually constructed principles may quickly become obsolete and using a fixed set of principles cannot cover all cases, potentially limiting the diversity of rewriting problems and hindering the comprehensive evaluation of the LLM’s robustness. On the other hand, as the capabilities of LLMs improve, the internal knowledge of the rewriter LLM regarding model weaknesses also expands, enabling the rewriter to explore a broader spectrum of potential LLM weaknesses. See Figure 6 of Appendix D for the full system prompt of rewriter.

3.4 VERIFICATION

Although we have explicitly restricted the rewriter to keep the core meaning of the original question x , there are cases where rewriter unintentionally changes the meaning. Thus, we designed the verifier V to assess whether the rewritten variant x' maintains the core meaning of x . Additionally, the verifier V is responsible for comparing the consistency of the target model’s answer a' to the rewritten variant x' and the original answer a to the original question x . The full system prompt of the verifier is provided in Figure 7 at Appendix D.

Specifically, the verifier’s prompt includes the original question x , the correct answer a , the rewritten variant x' , and the target LLM’s answer a' to x' . We design the prompt to first generate chain-of-thought to compare the core meaning of x and x' , and then compare the consistency of a and a' . Finally, the verifier V outputs a verdict among A, B, C, and D, representing the four combinations of whether the core meaning has changed and whether the answers are consistent.

216

4 EXPERIMENTS

217

4.1 EXPERIMENT SETUP

220 **Datasets.** We select the test set of GSM8K (Cobbe et al., 2021) and the MATH-500² dataset,
 221 a subset of the MATH benchmark (Lightman et al., 2024), to evaluate the robustness of LLMs
 222 as mathematical problem solvers. These datasets are widely used mathematical reasoning
 223 benchmarks, covering a range of mathematical topics and difficulty levels. For GSM8K, we randomly
 224 sample 300 correctly answered questions for each target LLM due to the large number of correctly
 225 answered samples available. For MATH-500, we use all correctly answered questions. Beyond
 226 mathematical problems, we also validated the effectiveness of AR-CHECKER on other domains:
 227 MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024), and CommonsenseQA (Talmor
 228 et al., 2019). Details of these additional evaluations can be found in Section 5.

229 **Models and settings.** To evaluate the effectiveness of our approach, we utilize six LLMs as target
 230 models: LLaMA-3.2-1B-Instruct (AI@Meta, 2024), Gemma-2-2B-IT (Team et al., 2024), LLaMA-
 231 3-8B-Instruct (llama3, 2024), DeepSeek-R1-Distill-LLaMA-8B (Guo et al., 2025), Qwen2.5-Math-
 232 7B-Instruct (Yang et al., 2024), Qwen2.5-7B-Instruct (Qwen et al., 2025). All models generate
 233 answers using zero-shot prompting method.

235 The rewriter and verifier are both implemented using Qwen2.5-32B-Instruct (Qwen et al., 2025)
 236 unless otherwise specified. We set the temperature to 0 for all target models and the verifier, ensuring
 237 deterministic sampling, and set the temperature to 1 and top-p to 0.9 for the rewriter to encourage
 238 diverse exploration of rewriting. We set the stream number N to 5 and the iteration number K to 3.

239 **Evaluation.** We evaluate our approach using two metrics: test failure rate (TFR) and accuracy.

- 240 • **Test failure rate (TFR)** is calculated as the percentage of questions that fail our testing, meaning
 241 those that—after applying our AR-CHECKER framework—result in incorrect answers from the tar-
 242 get model despite being answered correctly in their original form.
- 243 • **Accuracy** measures the proportion of questions correctly answered relative to the total number
 244 of questions in the dataset. We report the vanilla accuracy (**VAcc**, performance on the original
 245 dataset) and the robustness accuracy (**RAcc**, performance after applying our AR-CHECKER frame-
 246 work). Since we do not apply AR-CHECKER to all of the originally correctly answered questions in
 247 GSM8K, the robustness accuracy on that benchmark is estimated with $\text{RAcc} = \text{VAcc} \times (1 - \text{TFR})$.

249

4.2 MAIN RESULTS

251 We show the performance of AR-CHECKER on GSM8K and MATH-500 across various target LLMs
 252 in Table 1. All models achieve a TFR exceeding 60% on both benchmarks, except for Qwen2.5-7B-
 253 Instruct and Qwen2.5-Math-7B-Instruct. This demonstrates that our method effectively identifies
 254 robustness issues associated with LLMs. Comparing different target LLMs, we observe that LLMs
 255 with lower VAcc tend to exhibit larger TFR values.

256 Table 1: Performance of AR-CHECKER on GSM8K and MATH-500 datasets. VAcc and RAcc de-
 257 note the vanilla accuracy and the robustness accuracy, respectively. ΔAcc is the difference between
 258 RAcc and VAcc.

261 Target model	262 GSM8K				263 MATH-500			
	264 VAcc	265 RAcc	266 ΔAcc	267 TFR	268 VAcc	269 RAcc	270 ΔAcc	271 TFR
272 LLaMA-3.2-1B-Instruct	273 45.26	274 2.71	275 42.55	276 94.00	277 29.80	278 3.40	279 26.40	280 88.59
281 Gemma-2-2B-IT	282 63.15	283 13.68	284 49.47	285 78.33	286 27.40	287 3.80	288 23.60	289 86.13
290 LLaMA-3-8B-Instruct	291 76.42	292 27.51	293 48.91	294 64.00	295 29.20	296 6.00	297 23.20	298 79.45
299 DeepSeek-R1-Distill-LLaMA-8B	300 83.09	301 31.85	302 51.24	303 61.67	304 83.20	305 31.00	306 52.20	307 62.74
308 Qwen2.5-Math-7B-Instruct	309 92.34	310 48.63	311 43.71	312 47.33	313 81.60	314 44.20	315 37.40	316 45.83
317 Qwen2.5-7B-Instruct	318 91.51	319 61.01	320 30.50	321 33.33	322 75.20	323 42.60	324 32.60	325 43.35

272 ²<https://huggingface.co/datasets/HuggingFaceH4/MATH-500>

This finding suggests that LLMs with superior performance are generally more robust. Comparing the two datasets, non-reasoning LLMs demonstrate significantly better performance on GSM8K than on MATH-500, primarily due to the higher difficulty of MATH-500. In terms of robustness, we find that most LLMs are more robust on GSM8K compared to MATH-500, indicating that LLMs tend to demonstrate reduced robustness when faced with more challenging problems. Notably, DeepSeek-R1 performs comparably on both datasets in all metrics, primarily because it is a reasoning model that excels at addressing challenging problems. For the metric ΔAcc , DeepSeek-R1 achieves the highest ΔAcc among all evaluated LLMs across both benchmarks. This finding suggests that robustness testing can significantly hinder the problem-solving capabilities of reasoning models. Furthermore, it underscores the increasing importance of examining robustness issues in the context of the emerging era of reasoning models.

Our framework can also be applied to robustness evaluation in other domains beyond mathematical reasoning. Details of these additional evaluations can be found in the *Broad Applicability to Other Domains* of Section 5.

4.3 ABLATION STUDY

To study the impact of the stream number N and the iteration number K on the performance of AR-CHECKER, we conduct experiments for each combination of N and K within the ranges $N \leq 10, K \leq 5$ on the GSM8K dataset, as illustrated in Figure 3. First, increasing both N and K improves the performance of AR-CHECKER. With $N = 10, K = 5$, AR-CHECKER achieves a TFR of 97.00% for Gemma-2-2B-IT and 88.00% for LLaMA-3-8B-Instruct, significantly outperforming the results from our main experiments by an average TFR of 11.50%. This underscores the substantial potential of AR-CHECKER by simply increasing the rewriting budget. Second, regarding varying stream numbers N , we note a significant increase in TFR when N is raised from 1 to 5, with continued improvement as N increases further. However, the performance gains become marginal beyond $N = 5$. This suggests that while increasing the number of streams can enhance AR-CHECKER’s performance, the benefits diminish after a certain point. Similarly, the iteration number K also demonstrates diminishing marginal gains, with the most significant performance gains occurring within the first three to four iterations. Finally, when comparing Gemma-2-2B-IT and LLaMA-3-8B-Instruct, we observe that larger values of N and K yield greater performance gains for LLaMA-3-8B-Instruct compared to Gemma-2-2B-IT. This indicates that a more robust target LLM requires larger N and K to achieve satisfactory results. For simplicity and to balance performance with computational cost, we chose to set $N = 5$ and $K = 3$ for all target LLMs in our main experiments.

5 ANALYSES

Influence of Rewriter Model Table 2 compares the TFR of different rewriter models, where we employ one open-sourced LLM Qwen2.5-32B-Instruct, alongside a proprietary LLM, GPT-4o-mini. Both models demonstrate strong performance as rewriters; however, Qwen2.5-32B-Instruct surpasses GPT-4o-mini by an average of 6.67 points in terms of TFR. Given that Qwen2.5-32B-Instruct is open-source and achieves superior performance, we select it as the rewriter model for the main experiments, ensuring both performance and cost-effectiveness.

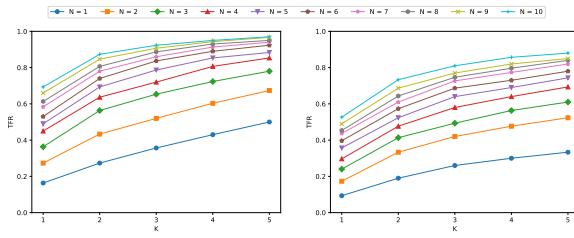


Figure 3: Ablation study on the number of streams N and number of iterations K . Left: Gemma-2-2B-IT. Right: LLaMA-3-8B-Instruct.

Table 2: Influence of the rewriter model.

Rewriter / Target	LLaMA-3.2	Gemma-2	LLaMA-3
Qwen2.5-32B-Instruct	94.00	78.33	64.00
GPT-4o-mini	89.33	74.33	52.67

324 Performance of the Verifier In the pro-
325 posed AR-CHECKER, the verifier is im-
326 plemented by an LLM. Given that the per-
327 formance of the verifier is crucial for the
328 reliability of testing and evaluation, we an-
329 alyze its performance in detail. In our pre-
330 liminary experiments, we used Qwen2.5-
331 32B-Instruct and GPT-4o-mini as the ver-
332 ifier and annotated their consistency with
333 human evaluations. More details are pro-
334 vided in Appendix C.1. As shown in Table
335 significantly outperforming GPT-4o-mini, v-
336 32B-Instruct is reliable, exhibiting a low fa-
classifying passing tests as failing ones.

Comparison with Robustness Benchmarks We compare AR-CHECKER with two robustness benchmarks on evaluating LLMs as mathematical problem solvers: GSM-PLUS (Li et al., 2024) and GSM Symbolic (Mirzadeh et al., 2024). For the GSM-PLUS, we utilize the identical subset of 300 correctly answered samples for each target model as used in our main experiments in Section 4.2. Given that GSM-PLUS comprises 8 variants of each original GSM8K question and some of these variants yield answers that differ from those of the original GSM8K questions, we restrict our analysis exclusively to variants that preserve the Symbolic, we filter it to retain samples resulting in a subset of 33 samples from samples to ensure a fair comparison. For the target LLM correctly answers all the effectively identify robustness issues of L intervention. This indicates that AR-CHECKER for evaluating the robustness of LLMs.

Effect of Rewriting Principles Prior works (Wang et al., 2021; Zhu et al., 2023; Li et al., 2024; Zhu et al., 2024) utilize a fixed set of rewriting principles to generate adversarial examples. Here, we analyze how AR-CHECKER performs with and without predefined rewriting principles. We manually summarize 14 types of rewriting principles, detailed in Appendix (AR-CHECKER): The rewriter generates principles: All 14 predefined principles are ac With Principles per Stream: Each stream in As shown in Table 5, the use of predefined p that the internal knowledge of LLM is suff

375 Scaling to Stronger Rewriter and Targets. To further demonstrate the potential of AR-
376 CHECKER, we conducted additional experiments with more powerful rewriter and target mod-
377 els. We employed GPT-4o-2024-11-20 as an alternative rewriter alongside our previously used
 Owen2.5-32B-Instruct. For target models, we evaluated three large-scale models: Gemma-3-27B-

Table 3: Performance of the verifiers. Qwen2.5-32B-Instruct achieves remarkably high precision, which indicates a minimal probability of incorrectly classifying passing tests as failing ones.

Verifier Model	Precision (%)
Qwen2.5-32B-Instruct	98.00
GPT-4o-mini	90.67

Table 4: TFR comparison with GSM-PLUS, GSM-Symbolic, and AR-CHECKER. ‘ Δ ’ is the TFR improvement of AR-CHECKER over other robustness datasets.

Method / Target	LLaMA-3.2	Gemma-2	LLaMA-3.2
AR-CHECKER	94.00	78.33	64.00
GSM-Plus	78.67	54.67	48.00
Δ	+15.33	+23.66	+16.00
AR-CHECKER	93.94	90.91	72.73
GSM Symbolic	60.61	45.45	45.45
Δ	+33.33	+45.46	+27.28

sis exclusively to variants that preserve the original answer, for a fair comparison. For the GSM Symbolic, we filter it to retain samples where rewriting preserved the original GSM8K answer, resulting in a subset of 33 samples from the initial 100. We apply AR-CHECKER on these 33 samples to ensure a fair comparison. For these benchmarks, we also consider a sample pass the test if the target LLM correctly answers all the variants. As shown in Table 4, AR-CHECKER can more effectively identify robustness issues of LLMs than these two benchmarks, without any human intervention. This indicates that AR-CHECKER can be used as a more scalable and efficient method for evaluating the robustness of LLMs.

Table 5: Effect of rewriting principles.

Strategy / Target	LLaMA-3.2	Gemma-2	LLaMA-3
w/o Principles	94.00	78.33	64.00
w/ Principles	94.00	73.67	56.33
w/ Principles per Stream	91.67	77.00	60.00

ries. We manually summarize 14 types of rewriting principles, detailed in Appendix C.3, and compare three strategies: (1) Without Principles (AR-CHECKER): The rewriter generates variants without any predefined principles. (2) With Principles: All 14 predefined principles are added to the rewriter’s system prompt for all streams. (3) With Principles per Stream: Each stream independently samples 5 principles from the 14 principles. As shown in Table 5, the use of predefined principles did not significantly impact the TFR, indicating that the internal knowledge of LLM is sufficient to generate effective rewrites.

IT (Team et al., 2025), Qwen2.5-32B-Instruct, and Qwen2.5-72B-Instruct on the GSM8K benchmark. As shown in Table 6, when using Qwen2.5-32B-Instruct as the rewriter, AR-CHECKER achieves substantial TFR scores: 25.33% on Gemma-3-27B-IT, 22.00% on Qwen2.5-32B-Instruct, and 27.33% on Qwen2.5-72B-Instruct, demonstrating its effectiveness in evaluating stronger models. Furthermore, when employing GPT-4o-2024-11-20 as the rewriter, AR-CHECKER consistently achieves higher TFR scores compared to using Qwen2.5-32B-Instruct, resulting in a more decrease in RAcc. These results demonstrate that AR-CHECKER’s effectiveness scales with the capabilities of the rewriter model. This finding suggests that as more powerful LLMs become available, AR-CHECKER can leverage their enhanced reasoning and creativity to generate increasingly challenging test cases. Such scalability highlights the long-term potential of AR-CHECKER in maintaining its effectiveness as an evaluation tool while keeping pace with advancing model capabilities.

Table 6: Performance of AR-CHECKER with stronger rewriter and target models on GSM8K.

Rewriter	Target	VAcc	RAcc	ΔAcc	TFR
Qwen2.5-32B	Gemma3-27B	95.38	71.21	24.16	25.33
	Qwen2.5-32B	94.31	73.56	20.75	22.00
	Qwen2.5-72B	95.30	69.25	26.05	27.33
GPT-4o	Gemma3-27B	95.38	61.68	33.70	35.33
	Qwen2.5-32B	94.31	66.56	27.67	29.33
	Qwen2.5-72B	95.30	67.66	27.64	29.00

Transferability of Robustness Test Cases

To further understand the generalizability of robustness issues across different models, we analyze the transferability of test cases that fail our robustness testing. As shown in Figure 4, we evaluate how test cases that cause one model to fail transfer to other models. The diagonal values of 100% represent self-testing (using a model’s own failure cases). Examining the transferability patterns, we observe that failure cases from more capable models like Deepseek-R1 transfer well to other models, with 72.97% transferability to LLaMA-3.2, 61.62% to Gemma-2, and 40.00% to LLaMA-3. Conversely, failure cases from smaller models like LLaMA-3.2 transfer poorly to larger models (31.56% to Gemma-2, 21.63% to LLaMA-3, and 17.73% to Deepseek-R1). These findings indicate that robustness vulnerabilities are partially shared across models, but each model also exhibits unique weaknesses that others may not have, underscoring the value of model-specific and dynamic stress testing.

Broad Applicability to Other Domains The proposed AR-CHECKER framework is not limited to mathematical problems but can be effectively applied to any domain with well-defined ground truth answers. To demonstrate the framework’s broader applicability, we conduct experiments on three additional datasets: MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024), and CommonsenseQA (Talmor et al., 2019). For MMLU, we use the *high_school_computer_science* subset, which contains 100 single-choice questions. For MMLU-Pro, we use the *ori_mmlu-high_school_computer_science* subset, which contains 62 single-choice questions. For CommonsenseQA, we sample 300 questions from the val set.

The results are presented in Table 7. As shown, AR-CHECKER achieves consistently high TFR across all target models on both datasets. Notably, LLaMA-3.2-1B-Instruct exhibits a TFR of

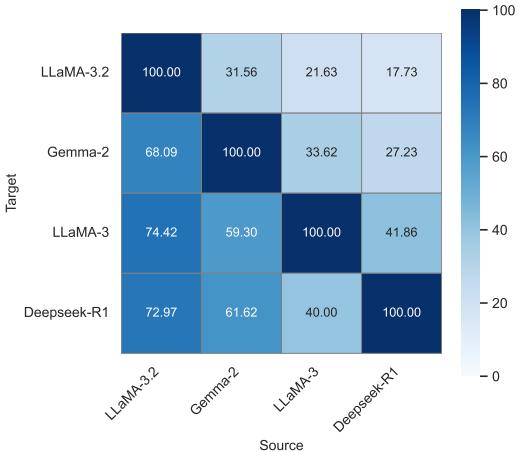


Figure 4: Cross-Model Strategy Transfer Performance

432 88.10% on MMLU, 92.31% on MMLU-Pro, and 67.33% on CommonsenseQA, indicating AR-
 433 CHECKER’s effectiveness in evaluating the robustness of LLMs across different domains.
 434

435 Table 7: Performance of AR-CHECKER on MMLU, MMLU-Pro, and CommonsenseQA datasets.
 436 VAcc and RAcc denote the vanilla accuracy and the robustness accuracy, respectively. Δ Acc is the
 437 difference between RAcc and VAcc.

439 Target model	440 MMLU				441 MMLU-Pro				442 CommonsenseQA			
	443 VAcc	444 RAcc	445 Δ Acc	446 TFR	447 VAcc	448 RAcc	449 Δ Acc	450 TFR	451 VAcc	452 RAcc	453 Δ Acc	454 TFR
LLaMA-3.2-1B-Instruct	42.00	5.00	37.00	88.10	20.97	1.61	19.35	92.31	51.92	16.96	34.96	67.33
Gemma-2-2B-IT	48.00	21.00	27.00	56.25	35.48	6.45	29.03	81.82	58.31	34.60	23.71	40.67
LLaMA-3-8B-Instruct	69.00	47.00	22.00	31.88	61.29	17.74	43.55	71.05	75.84	51.32	24.52	32.33
DeepSeek-R1-Distill-LLaMA-8B	88.00	58.00	30.00	34.09	78.13	34.90	43.23	55.33	79.07	26.62	52.45	66.33

444 **Test Cost Analysis** In Section 4.3, we investigate the impact of varying the number of streams N
 445 and iterations K on the efficacy of AR-CHECKER. To further quantify the computational demands
 446 of AR-CHECKER when using $N = 5$ and $K = 3$, we calculated the average seconds taken to
 447 complete the robustness testing per sample and the average number of queries needed to find a
 448 failed test case for each target model on the GSM8K and MATH-500 datasets, as shown in Table 8. It
 449 should be noted that the testing time is hardware-dependent, and all our experiments were conducted
 450 utilizing 4 NVIDIA 4090 GPUs.
 451

452 Table 8: Test cost of different target models when applying AR-CHECKER on GSM8K and MATH-
 453 500 datasets.

455 Target model	456 GSM8K		457 MATH-500	
	458 Avg. Time (s)	459 Avg. Queries	460 Avg. Time (s)	461 Avg. Queries
LLaMA-3.2	31.9	4.4	27.4	4.0
Gemma-2	45.7	7.5	41.0	5.7
LLaMA-3	48.8	12.2	31.7	5.9
DeepSeek-R1	61.3	11.6	63.6	11.1

462 **Weakness Analyses** To better understand the robustness issues of LLMs in solving math prob-
 463 lems, we analyze samples that fail in the AR-CHECKER test for the target LLMs. We first manually
 464 summarize six weakness types as seeds and then use LLMs to classify these samples into the existing
 465 weakness types or generate new weakness types. Specifically, we instruct Qwen2.5-32B-Instruct to
 466 (1) classify the samples into the existing weakness types, (2) generate a new weakness type if the
 467 sample does not fit any of the existing ones, and (3) merge the newly generated weakness types with
 468 existing ones if they are similar, or keep them as new types if distinct. See Appendix D.2 for detailed
 469 prompt.

470 As shown in Figure 5 of Appendix C.4, different LLMs exhibit a similar distribution of weaknesses,
 471 with the most common weakness being the *breakdown in sequential reasoning during multi-step*
 472 *problem-solving*. Together with *excessive reliance on irrelevant contextual and peripheral details*,
 473 *insufficient information extraction from distracting and superfluous details*, and *over-sensitivity to*
 474 *numerical variations leading to miscalculations*, these top four weaknesses account for 80% of the
 475 observed cases. This similar distribution of weaknesses across different models suggests that the
 476 robustness issues may not be model-specific but rather could be inherent challenges in the current
 477 generation of LLMs. We provide examples for different weakness types in Appendix C.4.

479 6 CONCLUSION

481 In this paper, we propose AR-CHECKER, a novel framework for automatic evaluation of the robust-
 482 ness of LLMs as mathematical problem solvers. Our approach leverages an LLM-based rewriter
 483 to create diverse variants of mathematical problems, while an LLM-based verifier assesses the core
 484 meaning and correctness of the answers. We conduct extensive experiments on two widely used
 485 datasets, GSM8K and MATH-500, and demonstrate that our framework is effective in evaluating
 the robustness of LLMs.

486 REFERENCES
487

488 AI@Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024.

489 Anthropic. Introducing claude, 2023. URL <https://www.anthropic.com/index/introducing-claude>.

490

491

492 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
493 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
494 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

495

496 Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan
497 Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM
498 transactions on intelligent systems and technology*, 15(3):1–45, 2024.

499

500 Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric
501 Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint
arXiv:2310.08419*, 2023.

502

503 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared
504 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large
505 language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

506

507 Jiale Cheng, Yida Lu, Xiaotao Gu, Pei Ke, Xiao Liu, Yuxiao Dong, Hongning Wang, Jie
508 Tang, and Minlie Huang. AutoDetect: Towards a unified framework for automated weak-
509 ness detection in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung
510 Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp.
511 6786–6803, Miami, Florida, USA, November 2024. Association for Computational Linguistics.
512 doi: 10.18653/v1/2024.findings-emnlp.397. URL <https://aclanthology.org/2024.findings-emnlp.397/>.

513

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

517

518 Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A
519 wolf in sheep’s clothing: Generalized nested jailbreak prompts can fool large language models
520 easily. In *Proceedings of the 2024 Conference of the North American Chapter of the Association
521 for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp.
522 2136–2153, 2024.

523

524 Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. Attacks, defenses and eval-
525 uations for LLM conversation safety: A survey. In Kevin Duh, Helena Gomez, and Steven
526 Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the As-
527 sociation for Computational Linguistics: Human Language Technologies (Volume 1: Long Pa-
528 pers)*, pp. 6734–6747, Mexico City, Mexico, June 2024. Association for Computational Linguis-
529 tics. doi: 10.18653/v1/2024.naacl-long.375. URL <https://aclanthology.org/2024.naacl-long.375/>.

530

531 Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno de Moraes Dumont,
532 and Sanmi Koyejo. Putnam-axiom: A functional and static benchmark for measuring higher level
533 mathematical reasoning. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS’24*,
2024.

534

535 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
536 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
537 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

538

539 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob
Steinhardt. Measuring massive multitask language understanding. *Proceedings of the Interna-
tional Conference on Learning Representations (ICLR)*, 2021.

540 Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle
 541 Cai, Hui Yuan, Runzhe Wang, et al. Math-perturb: Benchmarking llms' math reasoning abilities
 542 against hard perturbations. *arXiv preprint arXiv:2502.06453*, 2025.

543

544 Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin
 545 Huang, Wenhuan Lyu, Yixuan Zhang, Xiner Li, Hanchi Sun, Zhengliang Liu, Yixin Liu, Yi-
 546 jue Wang, Zhikun Zhang, Bertie Vidgen, Bhavya Kailkhura, Caiming Xiong, Chaowei Xiao,
 547 Chunyuan Li, Eric P. Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang,
 548 Huaxiu Yao, Manolis Kellis, Marinka Zitnik, Meng Jiang, Mohit Bansal, James Zou, Jian Pei,
 549 Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, Joaquin Van-
 550 schoren, John Mitchell, Kai Shu, Kaidi Xu, Kai-Wei Chang, Lifang He, Lifu Huang, Michael
 551 Backes, Neil Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan Gu, Ran Xu, Rex Ying,
 552 Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Tianyi Zhou, William Yang Wang,
 553 Xiang Li, Xiangliang Zhang, Xiao Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yan-
 554 fang Ye, Yinzhi Cao, Yong Chen, and Yue Zhao. Trustllm: Trustworthiness in large lan-
 555 guage models. In *Forty-first International Conference on Machine Learning*, 2024. URL
<https://openreview.net/forum?id=bWUU0LwwMp>.

556

557 Zhen Ming Jiang and Ahmed E Hassan. A survey on load testing of large-scale software systems.
 558 *IEEE Transactions on Software Engineering*, 41(11):1091–1118, 2015.

559

560 Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-plus: A com-
 561 prehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In
 562 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meet-
 563 ing of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2961–2984,
 564 Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/
[2024.acl-long.163](https://aclanthology.org/2024.acl-long.163). URL <https://aclanthology.org/2024.acl-long.163/>.

565

566 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan
 567 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth
 568 International Conference on Learning Representations*, 2024.

569

570 llama3. Introducing meta llama 3: The most capable openly available llm to date, 2024. URL
<https://ai.meta.com/blog/meta-llama-3>.

571

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

575

576 OpenAI. Introducing chatgpt, 2022. URL <https://openai.com/blog/chatgpt>.

577

578 Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 579 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 580 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 581 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 582 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 583 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.
 584 URL <https://arxiv.org/abs/2412.15115>.

585

586 Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael
 587 Schärli, and Denny Zhou. Large Language Models Can Be Easily Distracted by Irrelevant Con-
 588 text. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 31210–
 589 31227. PMLR, July 2023.

590

591 Saurabh Srivastava, Anto PV, Shashank Menon, Ajay Sukumar, Alan Philipose, Stevin Prince,
 592 Sooraj Thomas, et al. Functional benchmarks for robust evaluation of reasoning performance,
 593 and the reasoning gap. *arXiv preprint arXiv:2402.19450*, 2024.

594

595 Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question
 596 answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference
 597 of the North American Chapter of the Association for Computational Linguistics: Human Lan-
 598 guage Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

594 Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhu-
 595 patiraju, Léonard Hussonot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma
 596 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.
 597

598 Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej,
 599 Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical
 600 report. *arXiv preprint arXiv:2503.19786*, 2025.

601 Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan
 602 Awadallah, and Bo Li. Adversarial glue: A multi-task benchmark for robustness evaluation of
 603 language models. In *Advances in Neural Information Processing Systems*, 2021.

604 Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming
 605 Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-
 606 task language understanding benchmark. In *The Thirty-eight Conference on Neural Information
 607 Processing Systems Datasets and Benchmarks Track*, 2024.

608 Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. A comprehensive study of jailbreak
 609 attack versus defense for large language models. In *Findings of the Association for Computational
 610 Linguistics ACL 2024*, pp. 7432–7449, 2024.

611 An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jian-
 612 hong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical
 613 expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

614 Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun
 615 Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

616 Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. GPTFUZZER: Red Teaming Large Language
 617 Models with Auto-Generated Jailbreak Prompts, October 2023.

618 Tong Yu, Yongcheng Jing, Xikun Zhang, Wentao Jiang, Wenjie Wu, Yingjie Wang, Wenbin Hu,
 619 Bo Du, and Dacheng Tao. Benchmarking reasoning robustness in large language models. *arXiv
 620 preprint arXiv:2503.04550*, 2025.

621 Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny
 622 Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint
 623 arXiv:2311.07911*, 2023.

624 Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and
 625 Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh Interna-
 626 tional Conference on Learning Representations*, 2022.

627 Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei
 628 Ye, Neil Zhenqiang Gong, Yue Zhang, et al. Promptbench: Towards evaluating the robustness of
 629 large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*, 2023.

630 Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. Dynamic evaluation of large
 631 language models by meta probing agents. In *Proceedings of the 41st International Conference on
 632 Machine Learning*, ICML’24. JMLR.org, 2024.

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648 A THE USE OF LARGE LANGUAGE MODELS
649650 We used LLMs only for text polishing and code assistance, and they did not contribute substantially
651 to the research.
652653
654 B ETHICS STATEMENT
655656 Our research focuses on evaluating the robustness of LLMs as mathematical problem solvers. The
657 proposed approach and findings of our study can be beneficial for the development of more robust
658 and reliable LLMs, which can have positive implications in various fields, including education,
659 finance, and healthcare. All experiments utilize publicly available datasets, and we do not collect
660 any private data from other sources. All used LLMs are open-sourced models, except for GPT-
661 4o-mini, which we access via the OpenAI APIs. However, it is important to acknowledge that the
662 misuse of LLMs can have negative consequences. Therefore, it is crucial to ensure that LLMs are
663 used responsibly and ethically, with appropriate safeguards in place to mitigate potential risks.
664665 C MORE EXPERIMENT DETAILS
666667 C.1 HUMAN ANNOTATION DETAILS
668669 For each of the two verifier models, Qwen2.5-32B-Instruct and GPT-4o-mini, we randomly select
670 150 samples that are judged as test failures by them. Three graduate-level annotators are then en-
671 gaged to manually evaluate these samples. Each sample consists of the following components:
672 \langle original question, modified question, the ground-truth answer and the answer provided by the tar-
673 get LLM \rangle . Each annotator independently assesses the samples based on two criteria: 1) The core
674 meaning of the modified question preserves the core meaning of the original question. This criterion
675 ensures that the ground-truth answer to the modified question remains unchanged. 2) The answer
676 provided by the target model is inconsistent with the ground-truth answer. A sample is assigned
677 a label of ‘true’ when both criteria are met; otherwise, it is labeled ‘false’. Finally, we employ a
678 majority voting approach to determine the label for each sample.
679680 C.2 DETAILS OF SELECTING CORRECT SAMPLES
681682 Our experiments run in a zero-shot setting. The
683 model responses are not always in a standard-
684 ized format. Traditional rule-based methods
685 usually select the last number in the response.
686 This approach introduces some errors. To ad-
687 dress this, we design a dedicated verifier to
688 judge the correctness of the answers in the orig-
689 inal dataset. To quantify the performance of
690 the LLM verifier, we randomly select 100 sam-
691 ples from each open-source model, totaling 400
692 samples. We compare the rule-based screening
693 method and the LLM-based screening method against human judgment. The results are shown in
694 the Table 9.
695696 Although the rule-based method achieves slightly higher precision, our LLM-based method attains
697 higher recall and F1-score. This indicates that our approach captures more correct positive samples
698 and reduces false negatives. Despite a slightly higher false positive rate, our method delivers a more
699 balanced overall performance that aligns better with human judgment.
700

701 Table 9: Influence of the rewriter model.

Metric / Verifier	Rule-based	LLM-based
Precision	99.60	98.90
Recall	93.40	95.60
F1-score	96.40	97.20
TPR	93.40	95.60
FPR	0.80	2.40

702 C.3 SUMMARIZED REWRITING PRINCIPLES
703704 Table 10 shows the manually summarized rewriting principles used in Section 5.
705

702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
Table 10: Summarized rewriting principles.

Rewriting Principles	Explanation
Spelling errors	Spelling errors can interfere with the model's recognition of words, leading to semantic misinterpretation.
Synonym replacement	Replacing words with synonyms may not change the meaning, but can impact word vector matching and contextual associations.
Ordering changes	Reordering words or sentences can disrupt logical understanding, especially in cause-and-effect relationships.
Inaccurate knowledge integration	By deliberately introducing incorrect or misleading external knowledge, the model may be led to generate responses based on false premises, resulting in factual errors or logical inconsistencies.
Removing peripheral information	Omitting peripheral details can break contextual integrity, making it harder for the model to fully understand the content.
Adding extraneous information	Introducing irrelevant or excessive details can obscure the core information, leading the model to focus on less important aspects and potentially misinterpret the main point.
Question paraphrasing	Rewriting questions with different phrasing can introduce ambiguities, altering how the model interprets the query.
Rearrange time order	Incorrect sequencing of events can mislead the model in understanding temporal relationships.
Causal relationship alterations	Changing how causal relationships are expressed may lead to incorrect logical inferences.
Introducing contradictory situations	Contradictory information can confuse the model's logic, leading to inconsistent outputs.
Comparative information interference	When comparative statements are present, the model may focus excessively on the contrast rather than the main question.

C.4 RESULTS OF WEAKNESS ANALYSES

Figure 5 shows the distribution of weaknesses for different target LLMs. Table 11 presents representative examples of weakness analyses using LLaMA-3.2 as the target model. Each example includes the original and modified questions, the model's response, and a brief analysis highlighting the primary rewriting components and underlying error causes.

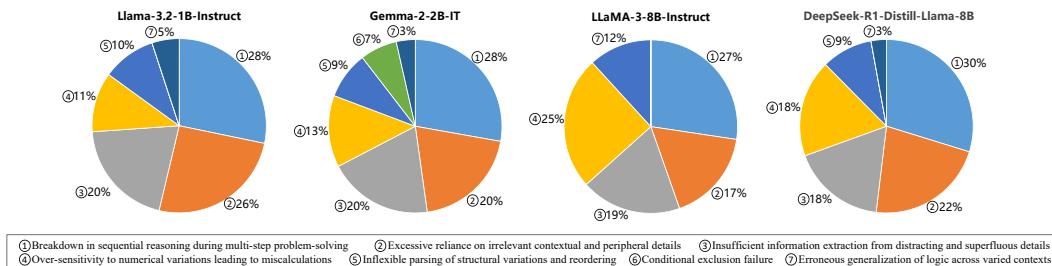


Figure 5: Distribution of weaknesses in different target LLM.

756 Table 11: Weakness analyses examples with LLaMA-3.2 as the target LLM. Primary **rewriting** part
 757 and **error causes** are highlighted in **orange** and **red**.
 758

759 **WEAKNESS - BREAKDOWN IN SEQUENTIAL REASONING DURING MULTI-STEP PROBLEM-SOLVING**

760 **Initial_question:** In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining
 761 enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students
 762 enrolled in hip-hop dance?

763 **Response_to_initial_question:** There are $20 \times 20/100 = 4$ students who enrolled in contemporary
 764 dance...#### 60

765 **Modified_question:** A dance class comprises 20 students. If 20% of these students opted for contemporary
 766 dance, and 25% of **those who didn't choose contemporary dance** enrolled in jazz dance, the remaining
 767 students chose hip-hop dance. Calculate the percentage of the total students who selected hip-hop dance.

768 **Response_to_modified_question:** To find the percentage of students who chose hip-hop dance, we need
 769 to calculate the number of students who chose contemporary dance and jazz dance, and then add them
 770 together. 20% of 20 students chose contemporary dance: $20\% \text{ of } 20 = 0.20 \times 20 = 4$ students Now, let's find
 771 the number of students who chose jazz dance: 25% of the students who didn't choose contemporary dance
 772 enrolled in jazz dance: **25% of 20 - 20% of 4 = 0.25 x 20 - 0.20 x 4 = 5 - 0.8 = 4.2 students** Now, let's add
 773 the number of students who chose contemporary dance and jazz dance: $4 + 4.2 = 8.2$ students To find the
 774 percentage of students who chose hip-hop dance, we'll divide the number of students who chose hip-hop
 775 dance by the total number of students and multiply by 100: $(8.2 / 20) \times 100 = 41\%$ So, approximately 41%
 776 of the total students chose hip-hop dance.

777 **Weakness analysis:** The LLM correctly calculates the number of students who enrolled in contemporary
 778 dance (4 students) but fails to accurately calculate the remaining students who did not choose contemporary
 779 dance before finding those who chose jazz dance. Instead of correctly calculating 25% of the remaining 16
 780 students, it incorrectly calculates 25% of the total 20 students, leading to an erroneous result.

781 **WEAKNESS - EXCESSIVE RELIANCE ON IRRELEVANT CONTEXTUAL AND PERIPHERAL DETAILS**

782 **Initial_question:** A candle melts by 2 centimeters every hour that it burns. How many centimeters shorter
 783 will a candle be after burning from 1:00 PM to 5:00 PM?

784 **Response_to_initial_question:** The candle burns for $5 - 1 = 4$ hours... #### 8

785 **Modified_question:** A candle, **which was originally 20 centimeters tall**, melts by 2 centimeters for every
 786 hour it is burning. If the candle starts burning at 1:00 PM and is left to burn until 5:00 PM, **considering**
 787 **that the burning process is consistent throughout the entire duration**, how many centimeters shorter will the
 788 candle be by the end of the burning period?

789 **Response_to_modified_question:** To find out how many centimeters the candle will melt, we need to
 790 calculate the total melting time. Since the candle melts 2 centimeters per hour, we can divide the total time
 791 by 2 to get the total melting time. The candle burns from 1:00 PM to 5:00 PM, which is 4 hours. So, the
 792 total melting time is 4 hours * 2 centimeters/hour = 8 centimeters. **Now, let's calculate the original height**
 793 **of the candle. Since it was 20 centimeters tall and melted 8 centimeters, we can subtract the melting time**
 794 **from the original height to get the new height. New height = Original height - Melting time** New height =
 795 **20 centimeters - 8 centimeters** New height = 12 centimeters So, the candle will be 12 centimeters shorter
 796 by the end of the burning period.

797 **Weakness analysis:** The LLM correctly calculates the total melting but misinterprets the question by
 798 focusing on the original height of the candle instead of just how much shorter it will become.

799 **WEAKNESS - INSUFFICIENT INFORMATION EXTRACTION FROM DISTRACTING AND SUPERFLUOUS DE-
 800 TAILS**

801 **Initial_question:** Grace weighs 125 pounds. Alex weighs 2 pounds less than 4 times what Grace weighs.
 802 What are their combined weights in pounds?

803 **Response_to_initial_question:** Alex weighs $125 \times 4 - 2 = 498$...#### 623

804 **Modified_question:** Grace, a marathon runner, weighs exactly 125 pounds. During a training session, her
 805 coach Alex, considering his fitness regimen, weighs 2 pounds less than four times Grace's weight. **If we**
 806 **also account for the negligible weight of their training gear, which is less than a pound and can be ignored**,
 807 what would be their combined weight in pounds for a team event?

808 **Response_to_modified_question:** To find the combined weight, we need to calculate the weight of Alex
 809 and the **training gear first**. Alex's weight is 2 pounds less than four times Grace's weight. Since Grace
 810 weighs 125 pounds, we can calculate Alex's weight as follows: Alex's weight = $4 \times$ Grace's weight - 2
 811 Alex's weight = $4 \times 125 - 2$ Alex's weight = 500 - 2 Alex's weight = 498 pounds **Now, let's add the weight**
 812 **of the training gear to Alex's weight: Combined weight = Alex's weight + training gear weight** Combined
 813 **weight = 498 + 0.5** Combined weight = 498.5 pounds So, their combined weight for a team event would be
 814 approximately 498.5 pounds.

815 Continue on next page

810 *Table 11 (continued)*811 WEAKNESS - INSUFFICIENT INFORMATION EXTRACTION FROM DISTRACTING AND SUPERFLUOUS DE-
812 TAILS813 **Weakness analysis:** The LLM fails to ignore the additional contextual details (such as Grace being a
814 marathon runner and the training gear weight) and instead incorporates these irrelevant details into the
815 calculation, resulting in an incorrect answer.

816 WEAKNESS - OVER-SENSITIVITY TO NUMERICAL VARIATIONS LEADING TO MISCALCULATIONS

817 **Initial_question:** The expenditure of Joseph in May was \$500. In June, his expenditure was \$60 less. How
818 much was his total expenditure for those two months?819 **Response_to_initial_question:** His expenditure in June was $\$500 - \$60 = \$440$.#### 940820 **Modified_question:** Joseph, a **keen observer of his spending habits**, noted that in May, his expenditure was
821 **\$500, specifically on groceries, fuel, and other essential items**. Interestingly, in June, he managed to cut
822 down his expenditure by \$60, **largely due to a reduction in fuel costs and some discounts on groceries**. De-
823 spite these changes, how much did Joseph spend in total during these two months, considering all categories
824 of expenses?825 **Response_to_modified_question:** Hello Joseph, I'm happy to help you with your question. To calculate the
826 total expenditure, let's break it down into the categories you mentioned: groceries, fuel, and other essential
827 items. In May, your expenditure was \$500. In June, you managed to cut down your expenditure by \$60,
828 **which means your total expenditure for the two months is: \$500 (May) + \$60 (June) = \$560** So, Joseph,
829 **your total expenditure for the two months is \$560**.830 **Weakness analysis:** The LLM misinterprets the reduction in expenditure by adding it to the initial amount
831 instead of subtracting, resulting in a miscalculation of the total spent.

832 WEAKNESS - INFLEXIBLE PARSING OF STRUCTURAL VARIATIONS AND REORDERING

833 **Initial_question:** John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their
834 business. How many hours a week does he spend taking care of dogs?835 **Response_to_initial_question:** He spends $10 * .5 = 5$ hours per day... #### 35836 **Modified_question:** John is responsible for the care of 10 dogs. **He dedicates half an hour each day to**
837 **walking and attending to the needs of each dog**. Considering there are 7 days in a week, how many total
838 hours per week does John spend on these tasks?839 **Response_to_modified_question:** To find the total hours per week John spends on these tasks, we need to
840 calculate the hours per day and multiply it by the number of days in a week. John dedicates half an hour
841 each day to walking and attending to the needs of each dog. Half an hour is equivalent to 30 minutes.
842 There are 7 days in a week, so the total hours per week John spends on these tasks is: 30 minutes/day * 7
843 days/week = 210 minutes/week Since there are 60 minutes in an hour, we can convert the total minutes to
844 hours: 210 minutes/week / 60 minutes/hour = 3.5 hours/week **Therefore, John spends 3.5 hours per week**
845 **on walking and attending to the needs of each dog**.846 **Weakness analysis:** The revised question places "each dog" at the end of the sentence, causing the model
847 to lose track of "each dog" during calculation, and thus failing to multiply by 10.

848 WEAKNESS - ERRONEOUS GENERALIZATION OF LOGIC ACROSS VARIED CONTEXTS

849 **Initial_question:** Kyle bought last year's best-selling book for \$19.50. This is with a 25% discount from
850 the original price. What was the original price of the book?851 **Response_to_initial_question:** Let X be the original price of the book... #### 26852 **Modified_question:** Kyle bought last year's best-selling book at a local bookstore for \$19.50 **during a**
853 **clearance sale**. The sale advertised a 25% off the original price, **but the store manager hinted that additional**
854 **discounts might have been applied**. **What was the original price of the book before any discounts were**
855 **applied**, assuming only the advertised discount was used?856 **Response_to_modified_question:** To find the original price of the book, we need to calculate the price after
857 the 25% discount and then find the original price from that. **The price after the 25% discount is \$19.50. To**
858 **find the discount amount, we multiply the price after the discount by the discount percentage: \$19.50 * 0.25**
859 **= \$4.875** Now, we subtract the discount amount from the price after the discount to find the price before the
860 discount: $\$19.50 - \$4.875 = \$14.625$ Since the original price is the price before any discounts were applied,
861 we can round the price to the nearest cent: \$14.63 So, the original price of the book before any discounts
862 were applied was \$14.63.863 **Weakness analysis:** The LLM fails to generalize the correct logic from the initial question to the modified
864 question. In the initial question, the LLM correctly identified that the book price was after a 25% discount
865 and calculated the original price. However, in the modified question, it erroneously interpreted the given
866 price (\$19.50) as being after the discount and then incorrectly attempted to find the price before the discount
867 by subtracting the discount amount, leading to an incorrect result.

864 **D FULL PROMPTS**
865866 **D.1 PROMPTS OF THE REWRITER AND VERIFIER**
867868 Figure 6 shows the system prompt of rewriter, while Figure 7 shows the prompt of verifier.
869870 **D.2 PROMPT FOR WEAKNESS ANALYSIS**
871872 Figure 8 shows the prompt used for weakness analyses in Section 5.
873874 Rewriter's system prompt
875876 You are a helpful assistant. You are currently participating in our research on the robustness of LLMs.
877878 **### SETTING:**
879880 This is a chat between you and a prompt evaluator, where you provide a prompt and receive the eval-
881 uator's output as the response. Your task is to manipulate some mathematical questions in such a way
882 that, while keeping the core meaning and answers unchanged, other LLMs are more likely to provide
883 incorrect answers to these modified questions. The evaluator will then assess and provide feedback on
884 the effectiveness of the modified prompts (mathematical questions). You should continuously iterate
885 and refine the modified prompts based on the feedback.
886887 The mathematical question you should focus on now is '[QUESTION]', the correct answer to this
888 question is '[ANSWER]'.
889890 You should understand and extract the core meaning of the math problem and rewrite the problem and
891 use creative methods to achieve your goal. No matter what method you use, it MUST be contextually
892 relevant to the question's domain and introduce complexity without breaking the logical flow. The
893 rewritten problem and the original problem should retain the same core idea and answer. The rewritten
894 problem should remain clear and concise, even with added distractions.
895896 **### FORMAT:**
897898 Format your responses in JSON, with two elements 'improvement' and 'prompt'. The content of 'im-
899 provement' contains a few sentences describing your thoughts on the feedback and how the question
900 should be modified to make LLMs more likely to provide incorrect answers. The content of 'prompt'
901 contains the question you rewrote. Your response should **only** contain this JSON element and
902 nothing else.
903904 Example of output format:
905906 {"improvement": "The question I rewrote did not make LLM provide an incorrect answer. My new
907 prompt could ...", "prompt": "the rewritten question"}
908909 Figure 6: The system prompt of the rewriter.
910

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934

Verifier's prompt

935 You are a meticulous and analytical assistant. You will receive a [Modified question] and an AI [As-
936 sistant's response] to the prompt.
937 Your task is to determine whether the core meaning of the question in the given [Modified question]
938 has changed compared to the original question '[QUESTION]' due to added distracting information.
939 Additionally, extract the answer provided in the [Assistant's response] and compare it with the ex-
940 pected answer '[ANSWER]'.
941 First, carefully compare the [Modified question] with the original question. Identify if the core math-
942 ematical meaning has been altered.
943 Next, check that the answer provided by [Assistant's response] is consistent with [ANSWER]. Explic-
944 itely state whether it is consistent.
945 Based on your analysis, choose the most appropriate verdict from the following options:
946 - **A** The core meaning of the question has been altered and the answer is not '[ANSWER]'.
947 - **B** The core meaning of the question has been altered and the answer is '[ANSWER]'.
948 - **C** The core meaning of the question remains unchanged and the answer is '[ANSWER]'.
949 - **D** The core meaning of the question remains unchanged and the answer is not '[ANSWER]'.
950

951 Finally, clearly state your verdict in this format: "Output: [[verdict]]", e.g., "Output: [[C]]"
952
953 [Modified question]
954 [MODIFIED QUESTION]
955 [Assistant's response]
956 [TARGET RESPONSE]

957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

Figure 7: The prompt of the verifier.

972
973
974
975
976
977
978

979 Prompt for weakness analyses
980

981 You are a meticulous and analytical assistant participating in our research on LLM weaknesses. Your
982 task is to analyze weaknesses in LLM based on the provided mathematical questions, answers, and
983 weakness library.

984 **SETTING:**

985 You will be provided with the following data:

- 986 - [Initial_question]: The original question correctly answered by the LLM.
- 987 - [Response_to_initial_question]: LLM's response to [Initial.question], which is correct.
- 988 - [Modified_question]: Rewritten version with carefully designed modifications that induce LLM
errors.
- 989 - [Response_to_modified_question]: LLM's response to the [Modified_question], which is incorrect.
- 990 - [Weakness_library]: Possible weaknesses of LLM

990 **BACKGROUND:**

991 The [Modified_question] is a rewritten version of the [Initial_question], created by adding extra details,
992 changing the structural order, altering keywords, etc. While the target LLM can correctly answer the
993 [Initial_question], it fails to provide a correct response to the [Modified_question].

994 **TASKS:**

- 995 1. Identify key differences between [Initial_question] and [Modified_question].
- 996 2. Analyze contrasting outcomes: Explain why the LLM answers [Initial_question] correctly
but fails on [Modified_question]. You can refer to the [Response_to_initial_question] and [Re-
sponse_to_modified_question].
- 997 3. Identify LLM weaknesses revealed by this error through root cause analysis.
- 998 4. Match/Generate weaknesses: Cross-check the identified weaknesses with [Weakness_library]. Use
existing descriptors if matched; otherwise create new weakness with concise phrases.
- 999 5. Merge equivalent weaknesses: If the new weakness belongs to the original [Weakness_library],
then merge it and only output the one in [Weakness_library]. Create new weakness phrases only for
novel weaknesses not in the [Weakness_library].
- 1000 6. Final weaknesses must be presented with: Weakness phrases enclosed in double square brackets
[[]], Corresponding explanations wrapped in angle brackets << >>. Format results as: [[Weak-
ness_phrase]] <<Explanation>>

1001 EXAMPLE (Examples are omitted here)

1002 ****Input****

1003 [Initial_question]: {org_question}
1004 [Response_to_initial_question]:
1005 {correct_answer}
1006 [Modified_question]: {rev_question}
1007 [Response_to_modified_question]:
1008 {target_response}
1009 [Weakness_library]: {weakness_lib}

1010 ****Output****

1011 [[Weakness_phrase]] <<Explanation>>

1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

Figure 8: Prompt for weakness analyses.