GSM-Noise: Exploring and Enhancing Large Language Models' Reasoning under Noisy Inputs

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated impressive reasoning capabilities, yet they often struggle when dealing with complex, ill-formed, or noisy inputs that frequently occur in interactions with real users. LLMs typically lack crucial refining capabilities needed to filter out irrelevant details, restructure key points before reasoning over the text and responding, resulting in suboptimal performance and incorrect answers. From an information theory perspective, this behavior is akin to decoding a high-entropy problem without first reducing its entropy. In this work, we first introduce GSM-Noise, a benchmark featuring grade-school math problems systematically perturbed to reflect real-world input variability. We show that the reasoning ability of open-source models (e.g., LLaMA and Qwen series) can be compromised by noise, while closed-source models are more robust. To improve LLM robustness under noisy conditions, we propose that LLMs first refine inputs — thereby reducing their entropy — before engaging in in-depth analysis. We investigate three approaches to instill this refinement capability: prompt engineering (PE), supervised finetuning (SFT), and reinforcement learning (RL). Experimental results show that input refinement leads to consistent performance gains: 2–12% with PE, 4–13% with SFT, and 3–25% with RL. These results highlight the importance of incorporating an explicit refinement phase to enhance the robustness and reliability of LLM reasoning in real-world scenarios.

1 Introduction

Recent advancements in large language models (LLMs) have demonstrated their ability to achieve, and in some cases surpass, human-level performance on a variety of benchmarks, including mathematical reasoning and code generation tasks (Austin et al., 2021; Chen et al., 2021; Cobbe et al., 2021; Hendrycks et al., 2021; Achiam et al., 2023; Dubey et al., 2024; Team et al., 2023, 2024). However, these notable successes obscure the fragility of LLM reasoning when confronted with ambiguous, noisy, or otherwise imperfect input. A growing body of research suggests that current LLMs often rely on surface-level pattern matching rather than robust logical inference (Jiang et al., 2024; Mirzadeh et al., 2024), making them sensitive to subtle variations in the input. Minor changes in statement order, irrelevant distractors, or altered entities in the input can result in dramatically inconsistent and sometimes incorrect outputs (Shi et al., 2023; Chen et al., 2024; Berglund et al., 2023; Jiang et al., 2024), thereby calling into question the reliability of these models in real-world scenarios.

This phenomenon can be understood through an information-theoretic lens: directly reasoning over a high-entropy, noisy input is akin to decoding a complex message without first simplifying it. While humans naturally reduce cognitive load by filtering out irrelevant information, restructuring key details, and refining their understanding, LLMs do not inherently perform these useful refinement steps. Instead, they attempt chain-of-thought (CoT) reasoning (Wei et al., 2022) directly on the raw input, often leading to suboptimal results.

We first demonstrate that users inevitably introduce various forms of noise into their prompts in real-world scenarios, as detailed in §3. While prior studies (Shi et al., 2023; Chen et al., 2024; Berglund et al., 2023; Jiang et al., 2024) have explored the effects of specific input variations—such as adding irrelevant context or modifying entity names—they generally focus on a single type of perturbation and thus fall short of simulating the diversity of noise found in real-world conversations. To comprehensively evaluate LLMs' reasoning ability in real-world conversations, we introduce GSM-Noise, an enhanced benchmark built upon R-GSM (Chen et al., 2024) and GSM8K (Cobbe et al., 2021). Unlike prior work, GSM-

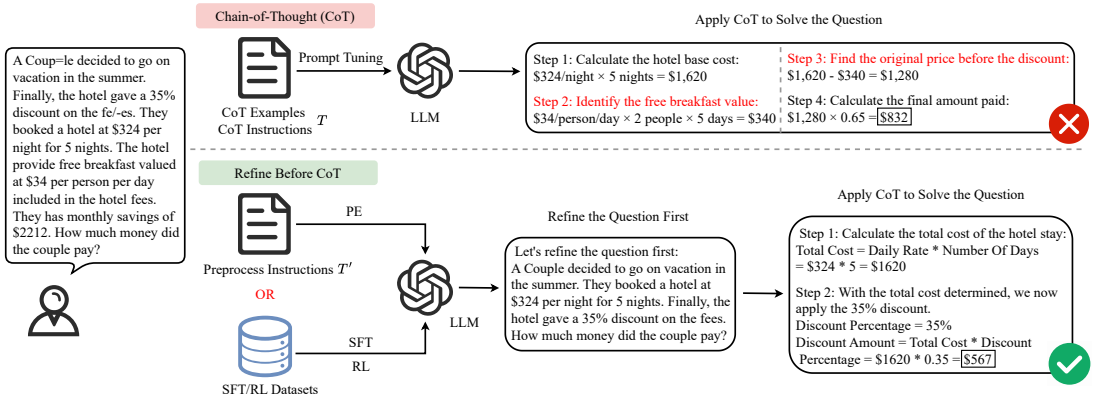

Figure 1: An example illustrating CoT and Refine Before CoT. PE, SFT, and RL denote prompt engineering, supervised fine-tuning, and reinforcement learning. Errors in CoT reasoning are shown in red.

Noise systematically applies a wide spectrum of perturbations to the input, including statement shuffling, irrelevant information injection, and symbol or grammatical errors. Specifically, we construct problem templates—each capable of generating multiple unique variants— to allow GSM-Noise more accurately capture the range of complexity and noise that LLMs may encounter in real-world settings. We test both open-source and closed models (e.g., LLaMA3 series, Qwen2.5 series, and ChatGPT series). We find that existing LLMs exhibit varying degrees of vulnerability to noisy inputs, with accuracy decreasing by 5% to 25%.

To address the vulnerability of LLMs to noisy inputs, we investigate three methods–prompt engineering (PE), supervised finetuning (SFT), and reinforcement learning (RL)–to guide LLMs in refining the input before conducting detailed reasoning, as shown in Figure 1. By reducing the entropy of the noisy input, e.g., removing irrelevant details, correcting formatting and grammar, and reordering statements, LLMs are better positioned to apply chain-of-thought (CoT) reasoning effectively. This refining process improves robustness, reduces the likelihood of confusion, and leads to more accurate solutions even under challenging input conditions.

Our results show that PE (Brown, 2020; Chowdhery et al., 2023), SFT, and RL yield accuracy improvements of 2–12%, 4–13%, and 3–25%, respectively, across various open- and closed-source LLMs. We also demonstrate that post-training methods without a refinement step fail to match the performance of our approach. Our analysis shows that refined problems generated by post-trained models exhibit significantly lower entropy (i.e., perplexity) than the original inputs. These findings highlight the importance of a preliminary refinement phase, paving the way for more robust, reliable, and human-like reasoning in LLMs.

2 Related Work

Reasoning abilities of LLMs. While LLMs have demonstrated impressive performance on a variety of benchmarks, there is still an ongoing debate about whether they truly possess logical reasoning abilities or simply rely on pattern matching. Studies have shown that LLMs often struggle when inputs are modified or perturbed, suggesting that their reasoning processes may lack robust logical inference (Jiang et al., 2024; Mirzadeh et al., 2024). To better understand the computational foundations of LLMs, researchers have investigated how transformer components map onto basic computational primitives (Weiss et al., 2021; Zhou et al., 2023). Some have found that transformers fail to generalize effectively on non-regular tasks and may require structured memory (e.g., scratchpads) for more complex reasoning (Delétang et al., 2022; Li et al., 2024). Techniques like Chain-of-Thought (CoT) (Wei et al., 2022) and auxiliary memory have improved LLM performance, but their effectiveness underscores the inherent expressiveness limitations of raw transformers. Recently, (Guo et al., 2025) and (Team et al., 2025) post-train the LLM using reinforcement learning (Kaelbling et al., 1996) (RL) to further enhance its reasoning ability. While these investigations provide valuable insights, they also highlight the persistent uncertainty: even with added memory, sophisticated prompting, and post-training, it is unclear whether LLMs can emulate the kind of stable, formal logical reasoning that humans perform naturally.

Prompt engineering and reasoning decomposition. Few-shot prompting (Brown, 2020; Chowdhery et al., 2023) is a powerful technique to en-

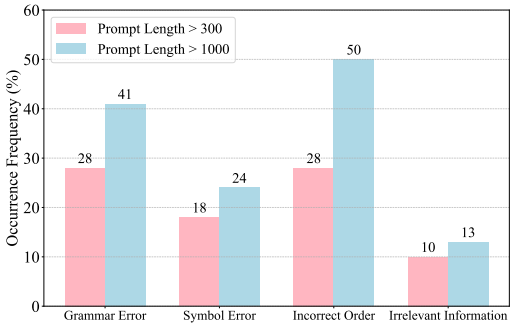

Figure 2: Frequency of common errors in different prompt lengths.

hance LLM performance by providing minimal yet informative examples. A range of methods have leveraged this concept to improve reasoning: generating intermediate reasoning steps (Ling et al., 2017; Cobbe et al., 2021; Nye et al., 2021; Wei et al., 2022; Suzgun et al., 2022; Shi et al., 2022b), decomposing complex problems into simpler subproblems (Zhou et al., 2022; Drozdov et al., 2022; Dohan et al., 2022; Khot et al., 2022; Press et al., 2022), and guiding models to produce code or logical plans (Austin et al., 2021; Chowdhery et al., 2023; Gao et al., 2023a; Chen et al., 2022). Other strategies include marginalizing over multiple reasoning paths (Wang et al., 2022a; Shi et al., 2022a) or ensembling different solutions (Wang et al., 2022b; Drozdov et al., 2022). Additionally, providing subtle hints or instructions within prompts can boost performance (Kojima et al., 2022). In this work, we extend this line of research by adding a preliminary refinement phase, prompting LLMs to refine the inputs themselves.

Natural language benchmarks with input variations. There is a line of work on adding different input variations for natural language tasks. For example, Liang et al. (2022); Ravichander et al. (2022) change the model-agnostic part of the input. Jia and Liang (2017); Shi et al. (2018); Morris et al. (2020); Wang et al. (2021) generate adversarial examples against individual models. Jia and Liang (2017); Kassner and Schütze (2019); Pandia and Ettinger (2021); Misra et al. (2022); Li et al. (2022) show that general-purpose pre-trained language models can be significantly affected by adversarial distracting sentences on factual reasoning tasks. Patel et al. (2021); Kumar et al. (2021) construct arithmetic reasoning benchmarks by paraphrasing or rewriting sentences in the base problems from clean datasets. Chen et al. (2024) changes the order in which the statements appear in the problem description for a math problem, and shows that statement order affects LLM reasoning. Besides, Mirzadeh et al. (2024) changes the entities and values in math problem descriptions, and the performance of LLMs significantly drops. They infer that current LLMs cannot perform genuine logical reasoning and pattern matching on their training data. Overall, prior works typically focus on a single type of imperfection variation and cannot control the difficulty of the dataset. Therefore, it is necessary for researchers to construct a benchmark with a wider variety of common error types and can introduce diverse variations into inputs.

3 Preliminary Experiments

To motivate our benchmark design, we conduct an analysis of real-world user inputs to examine the prevalence of common input errors. Based on prior works (Shi et al., 2023; Chen et al., 2024), we identify two frequent issues: the presence of irrelevant information and the incorrect ordering of statements. We further extend this set by incorporating additional error types commonly observed in user inputs, i.e., grammatical mistakes and symbol misuse. We use the WildChat-1M dataset (Zhao et al., 2024), which consists of user-ChatGPT conversations and reflects the distribution of real-world input scenarios. We first filter out non-English samples, retaining only English ones, which constitute 56.2% of the dataset. From this subset, we randomly select 100 user inputs, each of which exceeded 300 and 1,000 words, respectively. It's worth noting that approximately half of all user inputs exceed 300 words, while a quarter exceed 1,000 words. To assess the presence of common input errors, we prompt QwQ-32B to verify whether the user inputs contain any of the aforementioned errors. The frequencies of these common errors are shown in Figure 2. We find that longer user inputs consistently show higher error rates across all categories. This trend aligns with expectations, as longer inputs are more prone to irrelevant content and structural inconsistencies. Notably, even in the 300-word group, all error types occur with frequencies above 10%, indicating that such issues are common even in moderately long inputs. Therefore, it is necessary to explore and enhance LLMs' reasoning ability in real-world conversations.

4 The GSM-Noise Benchmark

We introduce GSM-Noise, an enhanced benchmark that adds diverse variations of noise to grade school

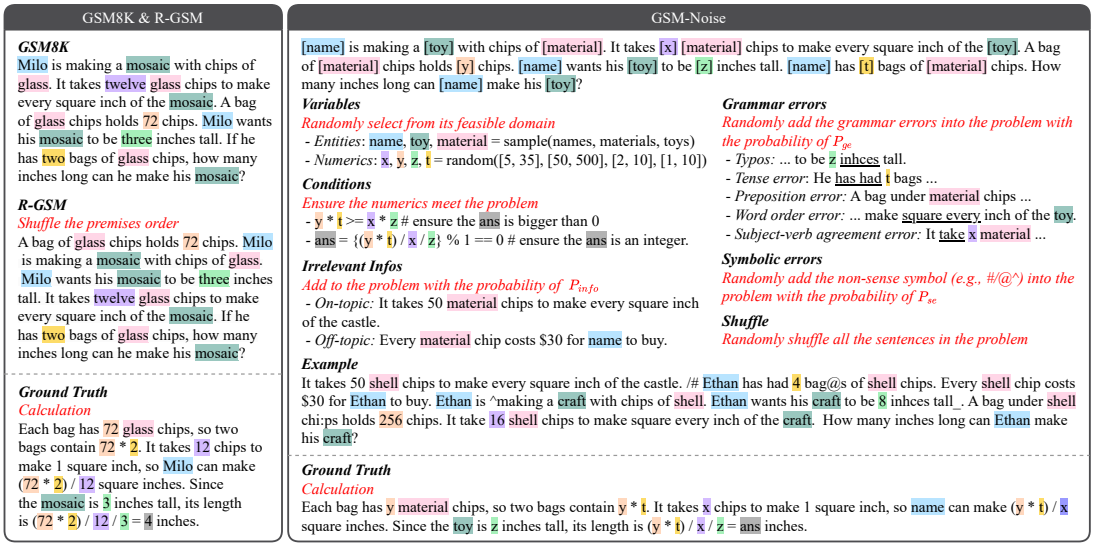

Figure 3: The problem template.

math problems and is designed to evaluate LLMs' performance in real-world input. Our benchmark is built upon the R-GSM (Chen et al., 2024) and GSM8K (Cobbe et al., 2021) and consists of 220 math problem templates. We introduce template creation and then describe how problems in GSM-Noise are generated using templates.

Template creation. For each problem in R-GSM, we manually create a parsable template, as illustrated in Figure 3, following a structured annotation process. The annotation begins with identifying variables within each problem, including entities (e.g., a person's name) or numerical values (e.g., the number of items). Then, we define the feasible domains for each variable. For an *entity variable*, the feasible domain is a set of similar items, e.g., different person names. For a *numerical variable*, we establish its feasible domain as a range (e.g., the price of a cup might range from 0 to 100) and/or necessary conditions (e.g., the number of people should be a positive integer). Next, we introduce irrelevant information to enhance the difficulty. We create a candidate set of on-topic irrelevant information and a candidate set of off-topic irrelevant information. The on-topic candidate set includes 2 to 4 statements of irrelevant information that are closely related to the problem statements, creating potential distractions without affecting the ground truth answer. The off-topic candidate set includes 1 to 2 statements of irrelevant information, introducing noise without semantic overlap.

Problem generation. The GSM-Noise benchmark is created by generating a number of problems from each template defined in the previous step. When generating a problem, we first set a value for each variable in the template. For an entity variable, we randomly select a value from its feasible domain set defined in the previous step. For a numerical variable, we randomly select a value within its feasible domain, verify if it meets its necessary conditions, and repeat the selection if it does not.

We then add **(1) irrelevant information** to the problem. Specifically, each candidate statement from the on-topic and off-topic sets is independently selected with probability P_{info}; selected statements are then appended to the problem to introduce irrelevant information. Next, we incorporate grammar and symbol errors into the generated problems. **(2) Grammar errors** are introduced through various syntactic perturbations, such as modifying subject-verb agreement, tense, prepositions, articles, word order, and character order. For each statement in the problem, we introduce a grammar error with probability P_{ge}. If a grammar error is to be introduced, we randomly select one grammar error type for the statement to simulate realistic mistakes. For **(3) symbol errors**, we randomly insert 1 to 3 meaningless symbols (e.g., @, #, &) before or after each character in each statement in the problem according to a probability P_{se}. Note that we do not add any symbol errors to *numerical variable* in the problem to avoid potential pollution to the ground truth. After this, we **(4) shuffle premise order**. We shuffle the order of all the statements for each generated problem, and then append the question statement (e.g., "What is the price after discount?") to the end.

Finally, we perform automated and manual

checks on generated problems to ensure each template's correctness. For every template, we generate 10 problems and use QwQ-32B to produce the answers. A low accuracy may suggest either flaws in the template or limitations in the model's capabilities. To rule out the former, we verify that the accuracy drop is not due to template issues. If the template is valid, then all generated Q–A pairs derived from it can be considered correct. If QwQ-32B's accuracy falls below 80%, we manually review and revise the template accordingly.

5 Refine before Analyze

In this section, we will demonstrate how to enhance the LLM's capability to refine a problem before analyzing it. We begin by revisiting the standard reasoning procedure. Given a problem x, the standard approach for generating an answer y using the autoregressive LLM π_θ is defined as $y \sim \pi_\theta(\cdot|x)$. For prompt-based reasoning techniques such as CoT (Wei et al., 2022), π_θ is first prompted to generate intermediate reasoning rationales before producing the final response $x' := T \oplus x, z \sim \pi_\theta(\cdot \mid x'), y \sim \pi_\theta(\cdot \mid x' \oplus z)$, where T represents an input prompt consisting of reasoning rationale (i.e., CoT) examples (Wei et al., 2022) or instructions (e.g., "Let's think step by step" (Kojima et al., 2022)), z denotes the reasoning rationale, and $\oplus$ is the concatenation operator. Previous work has empirically demonstrated that the CoT increases the likelihood of π_θ generating the desired answer $y*$, compared to directly sampling $y \sim \pi_\theta(\cdot \mid x)$. However, in real-world scenarios, most users are not experts or professionally trained. Their inputs are often complex, ill-formed, or noisy. Therefore, even with CoT, LLMs sometimes struggle to understand the input or be distracted by noise, as CoT does not inherently address these issues.

We attempt to reinterpret the above issue from an information-theoretic perspective (Shannon, 1948). User-provided inputs can be seen as high-entropy messages containing significant noise. CoT, in essence, attempts to directly decode these noisy, high-entropy inputs into meaningful responses. However, since it does not reduce the entropy or filter out the noise in the input, CoT can still produce low-quality or incorrect outputs. In contrast, humans naturally reduce entropy before tackling problems: they filter out irrelevant information, reorganize key points, and ensure a clear understanding of the objectives (Broadbent, 2013; MacLeod, 2007; Hirsh et al., 2012). This process increases the effective signal-to-noise ratio, making decoding (e.g., solving the problem) easier from an information-theoretic perspective. We argue that the LLM should emulate this human-like approach by first refining the inputs to simplify and clarify the input representation before engaging in actual analysis and answer generation. This "refinement" phase can help the model focus on the core problem, improve the robustness against flaws and noise in the input, and ultimately improve the correctness of its output. Building on this insight, we propose that LLMs should first refine their inputs before analyzing, outlined as:

$$x' := T' \oplus x, q \sim \pi_\theta(\cdot \mid x'), \quad (1)$$

$$z \sim \pi_\theta(\cdot \mid x' \oplus q), \quad (2)$$

$$y \sim \pi_\theta(\cdot \mid x' \oplus q \oplus z), \quad (3)$$

where T' represents the prompt consisting of refinement instructions (e.g., "Let's refine the problem first.") and previous reasoning rationale examples or instructions. q is the refined problem generated by the LLM. We provide an example of T' and q in Appendix A. To further enhance the LLM's reasoning capabilities, we employ Supervised Fine-tuning (SFT) using specially curated datasets. We employ the prompt template in Appendix B, and feed each problem x and prompt T' to QwQ-32B to generate the refined problem q, reasoning path z, and final answer y. To ensure dataset quality, we implement rejection sampling as our data selection. Specifically, we sample multiple refined problems and reasoning paths for each problem, then filter out low-quality samples where the generated answer does not match the ground truth. We then use the dataset to train smaller LLMs. Since our supervision signals come from QwQ-32B, our SFT can be viewed as distillation.

Furthermore, we use Reinforcement Learning (RL) to post-train the LLMs to improve their reasoning ability. We follow DeepSeek-R1 (Guo et al., 2025) and use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with a rule-based reward. Unlike DeepSeek-R1, which utilizes both format and accuracy reward, we only employ the accuracy reward and instruct the LLM to generate the refined problem, CoT, and the answer as shown in Appendix B. We find that the LLM still generates each component within its designated tags even without explicitly rewarding the format.

Table 1: Statistics of the GSM-Noisy benchmark datasets across different noise levels.

Dataset	P_{info}	P_{ge}	P_{se}	Shuffle?
Original	0.0	0.0	0.0	No
Level 1	0.2	0.1	0.004	No
Level 2	0.4	0.2	0.008	Yes
Level 3	0.6	0.3	0.012	Yes
Level 4	0.8	0.6	0.016	Yes
Level 5	1.0	0.8	0.02	Yes

6 Experiments Setup

In this section, we describe the dataset and models used in Section 7, and we provide more training details in Appendix C.

Datasets. We construct our datasets by sampling 10 problems from each template, yielding a total of 2,200 problems. The dataset is divided into training, validation, and test sets with a ratio of 7:1:2. To systematically evaluate model robustness under different levels of input noise, we create six datasets with five different noise levels and one without noise in our GSM-Benchmark. The statistics of six datasets are presented in Table 1. Note that both the quantity of problems per template and the noise difficulty level can be adjusted to accommodate various evaluation scenarios.

Models. We evaluate both open- and closed-source language models. Open-source models include LLaMA3.2-1B/3B, LLaMA3.1-8B/72B, Qwen2.5-1.5B/3B/7B/72B, and QwQ-32B (Dubey et al., 2024; Yang et al., 2024), assessed in main and PE experiments. Closed-source models include ChatGPT 3.5, 4o, 4o-mini, o3-mini, and Claude3.5 (Achiam et al., 2023; Jaech et al., 2024; Anthropic), evaluated under the same settings. For SFT and RL experiments, we focus on smaller models: LLaMA3.2-1B/3B, LLaMA3.1-8B, and Qwen2.5-1.5B/3B/7B.

7 Experimental Results

In this section, we present our main findings on the GSM-Noise benchmark, comparing model performance across various noise levels. Next, we detail our experiments with prompt engineering and post-training (i.e., SFT and RL). Finally, we analyze the entropy (i.e., perplexity) of the original problem and the refined problem.

Main results on the GSM-Noise benchmark. Our results in Table 2 reveal distinct patterns in how different LLMs respond to increasing levels of noise in GSM-Noise when using CoT. We observe two interesting findings:

Finding 1: Model accuracy and noise resilience increase with model scale. Both the LLaMA and Qwen series exhibit a clear positive correlation between model scale and performance robustness under noise. Smaller models in both families, such as LLaMA3.2-1B and Qwen2.5-1.5B, suffer substantial accuracy drops under high noise levels—LLaMA3.2-1B declines from 6.91% (Original) to 1.36% (Level 5), while Qwen2.5-1.5B drops from 37.50% to 20.23%. As models scale up, this degradation becomes less pronounced. For example, Qwen2.5-72B maintains 81.36% accuracy at Level 5, with only an 8.4% relative drop from its original 88.86%, while LLaMA3.1-70B retains 58.86% at Level 4 from an original 77.27% (a 24% relative drop). Interestingly, some scale inconsistencies exist—for instance, LLaMA3.2-3B slightly outperforms LLaMA3.2-8B at most noise levels—yet the overall trend still supports the conclusion that larger models are more resilient to noise. Notably, Qwen models consistently outperform LLaMA models at comparable scales across all noise levels, suggesting architectural or training differences in favor of Qwen. For closed-source models, similar scaling benefits are observed. ChatGPT 3.5 drops from 63.18% to 39.77% (a 37% relative decrease), while the more advanced ChatGPT 4o-mini and 4o demonstrate stronger resilience—only a 10.5% and 6.4% relative drop, respectively. Claude 3.5 also reaches similar levels of performance and stability.

Finding 2: Long CoT enhances both accuracy and noise resilience. QwQ-32B demonstrates remarkable consistency across all noise levels (84.60% to 85.68%), actually showing slight improvement under noise. Similarly, ChatGPT o3-mini exhibits exceptional stability among closed-source models (87.50% to 85.00%, just a 2.9% relative drop). By examining the reasoning outputs of these models, we find that long CoT models perform granular statement-level noise removal rather than the problem-level refinement proposed in our work. We show a case study in the Appendix I.1.

Experiments on prompt engineering. Building on our CoT analysis across noise levels, we evaluate the effectiveness of our prompt engineering (PE) method in comparison with CoT (Table 2; prompt template in Appendix A). Among open-source models, both the LLaMA and Qwen series generally benefit from our PE method, especially under mid-to-high noise levels. LLaMA3.2-1B, 3B, and 8B show consistent improvements, except

Table 2: Main results and PE experiments results on the GSM-Noise Benchmark.

Model	Method	Original	Level 1	Level 2	Level 3	Level 4	Level 5
LLaMA3.2-1B	CoT	6.91%	2.50%	3.18%	3.41%	2.50%	1.36%
	Ours	7.45%	3.18%	3.41%	4.67%	3.23%	1.82%
LLaMA3.2-3B	CoT	22.95%	23.41%	20.23%	17.50%	15.00%	12.50%
	Ours	24.77%	25.23%	19.77%	22.95%	19.09%	17.73%
LLaMA3.1-8B	CoT	18.92%	20.00%	21.36%	17.05%	21.59%	14.32%
	Ours	26.36%	25.23%	25.34%	18.86%	26.59%	17.50%
LLaMA3.1-70B	CoT	77.27%	68.64%	66.14%	68.41%	58.86%	68.41%
	Ours	76.59%	74.77%	74.32%	71.59%	68.41%	73.41%
Qwen2.5-1.5B	CoT	37.50%	32.50%	26.75%	26.36%	18.86%	20.23%
	Ours	36.14%	30.40%	26.12%	20.45%	21.36%	17.95%
Qwen2.5-3B	CoT	57.73%	51.82%	48.18%	40.23%	39.55%	33.64%
	Ours	58.41%	54.55%	50.00%	43.41%	39.09%	34.95%
Qwen2.5-7B	CoT	71.82%	67.05%	66.36%	61.14%	60.00%	56.14%
	Ours	71.82%	66.36%	68.64%	62.95%	62.95%	62.05%
Qwen2.5-72B	CoT	88.86%	82.27%	84.32%	79.77%	70.68%	81.36%
	Ours	84.55%	82.27%	83.86%	81.32%	81.82%	83.41%
QwQ-32B	CoT	84.60%	84.55%	81.59%	78.64%	80.68%	85.68%
	Ours	84.32%	84.32%	85.00%	81.36%	84.32%	86.14%
ChatGPT 3.5	CoT	63.18%	55.91%	51.59%	48.64%	42.05%	39.77%
	Ours	60.45%	55.91%	54.77%	50.00%	46.59%	42.50%
ChatGPT 4o	CoT	88.86%	89.32%	84.77%	82.95%	82.50%	83.18%
	Ours	87.50%	85.91%	82.27%	82.73%	82.50%	80.91%
ChatGPT 4o-mini	CoT	80.00%	78.86%	75.00%	73.41%	69.77%	71.59%
	Ours	81.59%	76.82%	75.23%	73.64%	74.55%	72.27%
ChatGPT o3-mini	CoT	87.50%	86.36%	85.45%	85.45%	85.00%	85.00%
	Ours	88.41%	88.18%	87.05%	85.91%	87.05%	85.91%
Claude 3.5	CoT	87.95%	79.55%	76.36%	75.45%	80.68%	82.27%
	Ours	83.64%	82.27%	82.95%	77.50%	80.91%	82.73%

Table 3: Post-training experiments results.

Model	Original	+SFT	+RL
LLaMA3.2-1B	1.36%	9.09%	13.00%
LLaMA3.2-3B	12.50%	34.77%	48.41%
LLaMA3.1-8B	14.32%	36.59%	45.36%
Qwen2.5-1.5B	20.23%	24.32%	23.41%
Qwen2.5-3B	33.64%	41.14%	45.00%
Qwen2.5-7B	56.14%	60.45%	65.21%

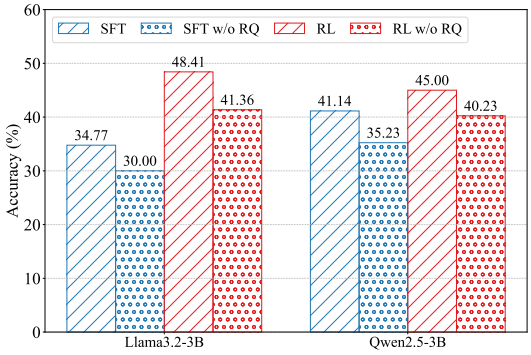

Figure 4: Comparison of post-training methods between w/ and w/o refining problem.

for a minor drop in LLaMA3.2-3B on the Level 2 dataset. LLaMA3.1-70B exhibits notable noise resilience, achieving up to 16.2% relative improvement at Level 4, despite a slight decrease on the original dataset. Similarly, Qwen2.5-3B and 7B mostly improve (except for Qwen2.5-3B at Level 4), and Qwen2.5-72B gains significantly at Level 4 (15.8% relative increase). However, Qwen2.5-1.5B consistently degrades, likely due to insufficient problem refinement—often repeating the final question statement rather than rephrasing it. QwQ-32B, already robust, sees further gains under noise with our method. Closed-source models also benefit from PE under noise. ChatGPT 3.5 and 4o-mini show relative improvements of 4.54% and 6.9%, respectively, at high noise levels. Claude 3.5 generally improves except on the noise-free dataset. Although ChatGPT 4o sees a slight decline, the ChatGPT o3-mini improves consistently, suggesting that our method complements long CoT reasoning and enhances robustness. We present few-shot learning experiments in Appendix D and a case study illustrating our prompt engineering in Appendix I.2. We also conduct a preliminary extension to factual QA reasoning in Appendix E to demonstrate the generalizability of our method.

Experiments on post-training. We evaluate the effectiveness of post-training techniques—SFT

Table 4: Comparison of entropy between original noisy input problems and refined problems.

Model	Original	RQ (SFT)	RQ (RL)
LLaMA3.2-3B	67.15	10.33	14.17
Qwen2.5-3B	62.00	10.69	36.64

followed by RL—on our most challenging Level 5 datasets. The prompt template used for post-training is shown in Appendix B. Results summarized in Table 3 show substantial performance improvements across all evaluated models. The LLaMA series demonstrates the most significant gains. LLaMA3.2-1B improves from 1.36% to 13.00% after SFT and RL, an 856% relative improvement. LLaMA3.2-3B shows a 287% relative increase (from 12.50% to 48.41%), surpassing Qwen2.5-3B after RL. While improvements in the Qwen family are more modest, they remain substantial—Qwen2.5-3B relatively improves by 33.8%, and Qwen2.5-7B achieves the highest overall performance with 65.21% accuracy. Compared to prompt engineering, post-training yields stronger gains in reasoning robustness under noisy inputs. We show a case study in Appendix I.3.

Ablation Study. To assess the importance of problem refinement, we conduct an ablation study by post-training LLaMA3.2-3B and Qwen2.5-3B without refining the input prompt (Appendix B). As shown in Figure 4, omitting problem refinement during post-training leads to a around 6% drop in accuracy, equivalent to a 16% relative decline. These findings highlight the necessity for both industry and academic practitioners to implement further post-training techniques to achieve more robust LLM reasoning capabilities in real-world applications where input quality cannot be guaranteed. We present a token usage comparison in Appendix F and provide additional experiments on out-of-domain (OOD) tasks in Appendix G to demonstrate the effectiveness and generalizability of our method. An analysis of the impact of each error type is included in Appendix H.

Experiments on entropy comparison. To assess whether post-trained models reduce the entropy of input problems through refinement, we compare the perplexity of refined problems—generated by the post-trained checkpoints of LLaMA3.2-3B and Qwen2.5-3B—with that of the original noisy inputs. We use the original (pre-trained) versions of the same models to measure perplexity. We use perplexity as a proxy for entropy, reflecting text predictability and coherence—lower values indicate more fluent, structured language. As shown in Table 4, both models produce refined problems with significantly reduced perplexity after SFT, confirming that post-training helps the model transform noisy input into more structured and predictable sequences. Interestingly, RL-trained models exhibit higher perplexity compared to their SFT counterparts. This is likely due to the absence of explicit format constraints in our RL setup, where the model occasionally refines only the final question rather than the full input, or outputs refined content outside the expected `<rq></rq>` tags. These results support the view that entropy reduction through refinement plays a key role in the accuracy gains observed in Section 7.

8 Discussion, Limitation, and Conclusion

In this work, we introduce GSM-Noise, a benchmark that systematically evaluates LLM reasoning under realistic inputs. Unlike previous work, our benchmark encompasses a wider variety of common error types and introduces diverse variations into inputs. However, GSM-Noise is limited to the mathematical domain, consists of only 220 templates, and may lack generality. In future work, we plan to extend to other domains, including code, factual QA, writing, and other open-ended tasks. Furthermore, we test a wider range of open-source models and closed-source models from 1B to 70B parameters and beyond. We observe two interesting findings: 1) Model accuracy and noise resilience increase with scale. Closed-source models outperform open-source models, likely due to this scaling advantage. 2) Long CoT models (e.g., QwQ-32B and ChatGPT o3-mini) exhibit the best accuracy and stability and achieve performance on par with results on the original dataset without noise. To improve LLM performance under noisy inputs, we propose to use PE, SFT, and RL methods. Our results demonstrate improvements of 2-12% with PE, 4-13% with SFT, and 3-25% with RL. We further show that refining the problem is a crucial step to achieve these improvements. This refinement step can be regarded as orthogonal to existing methods such as Retrieval-Augmented Generation (RAG) (Gao et al., 2023b; Lewis et al., 2020), LLM Planning (Huang et al., 2022), searching algorithm (Sel et al., 2024; Gandhi et al., 2024), and (long) CoT. We successfully combine problem refinement with (long) CoT in this work and will explore combinations with other approaches in future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. The claude 3 model family: Opus, sonnet, haiku.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the ai: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*.

Donald Eric Broadbent. 2013. *Perception and communication*. Elsevier.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and 1 others. 2022. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*.

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A Saurous, Jascha Sohl-Dickstein, and 1 others. 2022. Language model cascades. *arXiv preprint arXiv:2207.10342*.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. 2024. Stream of search (sos): Learning to search in language. *Preprint*, arXiv:2404.03683.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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*.

Jacob B Hirsh, Raymond A Mar, and Jordan B Peterson. 2012. Psychological entropy: a framework for understanding uncertainty-related anxiety. *Psychological review*, 119(2):304.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo J Taylor, and Dan Roth. 2024. A peek into token bias: Large language models are not yet genuine reasoners. *arXiv preprint arXiv:2406.11050*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Nora Kassner and Hinrich Schütze. 2019. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. *arXiv preprint arXiv:1911.03343*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

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.

Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2021. Adversarial examples for evaluating math word problem solvers. *arXiv preprint arXiv:2109.05925*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2022. Large language models with controllable working memory. *arXiv preprint arXiv:2211.05110*.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, and 1 others. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Colin M MacLeod. 2007. The concept of inhibition in cognition.

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*.

Kanishka Misra, Julia Taylor Rayz, and Allyson Ettinger. 2022. Comps: Conceptual minimal pair sentences for testing robust property knowledge and its inheritance in pre-trained language models. *arXiv preprint arXiv:2210.01963*.

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, and 1 others. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

Lalchand Pandia and Allyson Ettinger. 2021. Sorting through the noise: Testing robustness of information processing in pre-trained language models. *arXiv preprint arXiv:2109.12393*.

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

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Abhilasha Ravichander, Matt Gardner, and Ana Marasović. 2022. Condaqa: A contrastive reading comprehension dataset for reasoning about negation. *arXiv preprint arXiv:2211.00295*.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2024. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *Preprint*, arXiv:2308.10379.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I Wang. 2022a. Natural language to code translation with execution. *arXiv preprint arXiv:2204.11454*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, and 1 others. 2022b. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Haoyue Shi, Jiayuan Mao, Tete Xiao, Yuning Jiang, and Jian Sun. 2018. Learning visually-grounded semantics from contrastive adversarial samples. *arXiv preprint arXiv:1806.10348*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 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, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022b. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. 2023. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*.

A Example of T' and q

Prompt Template with Refinement Instructions

You are a helpful math assistant. The user asks a noisy math problem. You should first refine the whole math problem by eliminating irrelevant information, correcting grammar errors, removing nonsensical symbols, and then reorganizing the premises logically. You should then analyze and solve the refined math problem, provide a detailed reasoning process, and finally present the answer. The final answer must be wrapped inside a \boxed{{}} tag for clear extraction.

User: {problem}

Assistant:

Figure 5: An example of the prompt template consisting of refinement instructions T'.

Refined Question

User: The Scie~|nce Center hosted field trips Monday t_hrough Friday last\ week. On Monda|y, 48 classes visited. the same number as many v*isited on Tuesday= a?|~nd= three times as many visited on Wednesday. =S%#=tudents are preparing for the u@+pcoming sc\&ience fair. The @Science Cen!t+er w&i^ll be closed next week for renovations. M/?eanwhile, a new movie was released in theaters. Last mont~?h's f`%|ield trips were fewer due to bad we?ather. Another 30 clas@ses visited on Th!ursday and 40 visit+^_ed on Fri/day. In all, how many classes visited the Science Center last week?

Assistant: ... Last week, the Science Center hosted field trips from Monday through Friday. On Monday, 48 classes visited. On Tuesday, the same number as Monday visited. On Wednesday, three times as many visited as on Monday. On Thursday, 30 classes visited, and on Friday, 40 classes visited. How many classes visited the Science Center in total last week?
...

Figure 6: An example of a refined problem q generated by Qwen2.5-3B-SFT.

B Prompt Template for Post-training

Prompt template used for post-training (SFT and RL) w/ Refinement

A conversation between the User and Assistant. The user asks a noisy math problem. The assistant should first refine the whole math problem by eliminating irrelevant information, correcting grammar errors, removing nonsensical symbols, and then reorganizing the statements logically. It should then analyze and solve the refined math problem, provide a detailed reasoning process, and finally present the answer. The refined problem should be enclosed within <rq> </rq> tags. The reasoning process should be enclosed within <think> </think> tags, and the answer should be enclosed within <answer> </answer> tags. The final answer must be wrapped inside a \boxed{{}} tag for clear extraction.

User: {problem}

Assistant:

Figure 7: Prompt template used for post-training (SFT and RL) with the problem refinement step.

C Training Setup Details

For supervised fine-tuning, we train the LLaMA-1B, LLaMA-3B, Qwen-1B, and Qwen-3B models using the training data described in Section 6.1. We use bfloat16 precision to improve memory efficiency and

Prompt template used for post-training (SFT and RL) w/o Refinement

A conversation between the User and Assistant. The user asks a noisy math problem, and the assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process should be enclosed within <think> </think> tags, and the answer should be enclosed within <answer> </answer> tags. The final answer must be wrapped inside a \boxed{{}} tag for clear extraction.

User: {problem}

Assistant:

Figure 8: Prompt template used for post-training (SFT and RL) without the problem refinement step.

training stability. The maximum input sequence length is set to 2048 tokens. We use a per-device batch size of 4 and apply gradient accumulation over 4 steps, resulting in an effective batch size of 16. Training is conducted for 5 epochs. We adopt a cosine learning rate scheduler with a base learning rate of 1e-7 and a 10% warmup ratio. Model performance is monitored on the validation set, and the checkpoint with the best validation accuracy is used for evaluation on the test set.

For reinforcement learning, we adopt the GRPO algorithm as implemented in the VeRL framework. We use Ray for efficient distributed training. The setup uses bfloat16 precision with a per-device batch size of 4 and gradient accumulation over 16 steps, resulting in an effective batch size of 64. We set the maximum prompt length to 500 tokens and the maximum response length to 1508 tokens. The learning rate is set to 1e-6. For each question, 8 rollouts are generated in parallel. To improve generation speed, we utilize vLLM as the backend. Gradient checkpointing is enabled to reduce memory usage. We train for a total of 15 epochs. Both SFT and RL experiments are conducted on 2 NVIDIA A100 GPUs (80GB each).

D Experiments on Few-shot Prompting

In this section, we conduct additional experiments using few-shot prompting. Specifically, we apply one-shot prompts and evaluate all models introduced in Section 6. The results are shown in Table 5.

Table 5: Few-shot prompting results on the GSM-Noise Benchmark.

Model	Method	Original	Level 1	Level 2	Level 3	Level 4	Level 5
LLaMA3.2-1B	CoT	6.53%	2.73%	2.89%	3.11%	2.83%	1.75%
	Ours	7.18%	3.25%	3.07%	4.39%	2.67%	1.93%
LLaMA3.2-3B	CoT	24.59%	22.50%	21.47%	18.08%	16.33%	13.51%
	Ours	27.56%	25.67%	21.22%	20.15%	17.58%	15.92%
LLaMA3.1-8B	CoT	19.35%	21.42%	20.46%	18.57%	19.30%	15.27%
	Ours	25.60%	26.08%	25.44%	20.15%	21.36%	19.63%
LLaMA3.1-70B	CoT	79.57%	70.29%	67.33%	69.30%	57.51%	65.94%
	Ours	81.45%	77.21%	75.27%	72.36%	69.25%	73.59%
Qwen2.5-1.5B	CoT	41.58%	33.84%	27.94%	28.95%	21.03%	21.56%
	Ours	39.00%	32.67%	25.18%	23.87%	22.64%	20.50%
Qwen2.5-3B	CoT	58.49%	53.56%	50.44%	51.34%	41.67%	35.11%
	Ours	57.15%	54.80%	50.93%	50.56%	45.16%	37.97%
Qwen2.5-7B	CoT	72.49%	69.34%	67.58%	62.48%	60.97%	57.80%
	Ours	73.97%	70.00%	66.53%	61.07%	61.62%	60.25%
Qwen2.5-72B	CoT	88.16%	83.54%	84.98%	80.33%	73.12%	79.59%
	Ours	89.85%	85.11%	85.05%	82.67%	81.24%	80.01%
QwQ-32B	CoT	86.31%	85.67%	82.49%	80.56%	80.34%	83.73%
	Ours	85.64%	85.10%	86.28%	82.87%	83.37%	84.80%
ChatGPT 3.5	CoT	62.27%	57.95%	53.41%	49.55%	46.14%	42.95%
	Ours	59.32%	54.32%	56.14%	49.77%	46.14%	44.77%
ChatGPT 4o	CoT	91.36%	87.27%	85.91%	83.41%	83.64%	82.27%
	Ours	89.27%	87.05%	84.32%	81.59%	83.64%	84.77%
ChatGPT 4o-mini	CoT	78.18%	76.82%	75.23%	70.00%	70.00%	68.64%
	Ours	81.14%	76.82%	77.05%	74.32%	74.55%	70.91%
ChatGPT o3-mini	CoT	89.55%	85.68%	85.91%	86.36%	84.32%	85.23%
	Ours	90.00%	87.50%	88.18%	87.27%	85.68%	87.50%
Claude 3.5	CoT	86.36%	80.91%	75.45%	80.23%	79.55%	81.59%
	Ours	87.05%	83.41%	81.82%	80.45%	80.00%	85.00%

First, few-shot prompting improves the accuracy of both CoT and our refinement method across all models, with the exception of smaller models (i.e., those with 1B parameters), where the gains are negligible or absent. Second, our refinement consistently outperforms CoT on all open-source models. For closed-source models, our method surpasses CoT on ChatGPT 4o-mini, o3-mini, and Claude 3.5, but underperforms on ChatGPT 3.5 and ChatGPT 4o at Levels 1–3. However, our refinement still outperforms CoT at Levels 4–5 for both ChatGPT 3.5 and 4o, demonstrating its robustness and effectiveness in reducing hallucinations under high-noise scenarios.

E Experiments on Factual QA Domain

In this section, to further demonstrate our approach, we conduct a preliminary extension to factual QA reasoning. Specifically, we adapt the AdversarialQA dataset (Bartolo et al., 2020) and design 30 templates, generating 5 questions per template. We then evaluate these questions using both the open-source and closed-source models mentioned in Section 6, applying the PE method. The results are shown in Table 6.

Table 6: Main results and PE experiments results on factual QA domain.

Model	Method	Original	Level 1	Level 2	Level 3	Level 4	Level 5
LLaMA3.2-1B	CoT	17.50%	16.17%	14.67%	12.67%	10.00%	7.33%
	Ours	17.50%	15.39%	15.83%	12.00%	11.17%	6.67%
LLaMA3.2-3B	CoT	20.83%	20.00%	18.33%	14.33%	12.67%	10.00%
	Ours	22.51%	31.67%	23.33%	21.67%	20.00%	19.67%
LLaMA3.1-8B	CoT	21.17%	20.00%	23.67%	16.33%	13.50%	13.67%
	Ours	27.67%	25.33%	22.00%	20.67%	19.33%	18.17%
LLaMA3.1-70B	CoT	52.50%	50.00%	43.33%	35.00%	36.51%	33.33%
	Ours	74.17%	60.33%	55.00%	51.67%	48.67%	43.33%
Qwen2.5-1.5B	CoT	16.67%	14.33%	14.33%	12.17%	10.00%	8.33%
	Ours	17.33%	12.67%	12.00%	10.50%	9.17%	8.33%
Qwen2.5-3B	CoT	38.17%	33.33%	33.33%	28.00%	29.55%	26.67%
	Ours	42.33%	41.67%	40.50%	35.33%	35.17%	35.00%
Qwen2.5-7B	CoT	45.83%	57.67%	55.00%	41.33%	39.33%	36.00%
	Ours	52.17%	53.33%	50.67%	43.25%	40.60%	38.17%
Qwen2.5-72B	CoT	57.50%	63.33%	51.67%	55.00%	48.67%	47.33%
	Ours	69.17%	60.67%	66.25%	50.67%	58.50%	55.25%
QwQ-32B	CoT	55.00%	65.00%	53.33%	46.67%	49.50%	50.00%
	Ours	59.17%	60.00%	68.33%	58.33%	47.17%	56.67%
ChatGPT 3.5	CoT	42.50%	56.67%	48.33%	51.67%	48.33%	50.00%
	Ours	47.50%	41.67%	45.00%	50.00%	40.00%	46.67%
ChatGPT 4o	CoT	50.00%	50.00%	50.00%	50.00%	50.00%	50.00%
	Ours	50.00%	55.00%	51.67%	51.67%	55.00%	56.67%
ChatGPT 4o-mini	CoT	50.00%	51.67%	50.00%	48.33%	48.33%	55.00%
	Ours	55.83%	50.00%	48.33%	55.00%	48.33%	56.67%
ChatGPT o3-mini	CoT	62.50%	66.67%	68.33%	63.33%	63.33%	63.33%
	Ours	63.33%	70.00%	68.33%	68.33%	70.00%	60.00%
Claude 3.5	CoT	75.00%	76.67%	76.67%	73.33%	78.33%	73.33%
	Ours	66.67%	70.00%	68.33%	68.33%	68.33%	70.00%

First, our refinement generally outperforms CoT on open-source models, with the exception of smaller models (e.g., those with 1B parameters). For instance, our approach achieves up to a 20% absolute improvement on the LLaMA3.1-70B model. Second, the results are mixed for closed-source models: our refinement performs better on ChatGPT 4o, 4o-mini, and o3-mini, but underperforms on ChatGPT 3.5 and Claude 3.5. Third, in contrast to the accuracy degradation trend observed in GSM-Noise for both open-source and closed-source models, we find that the accuracy of closed-source models remains stable, or even improves, as noise levels increase. We hypothesize that this robustness stems from the fact that closed-source models are generally trained on larger pretraining corpora and with longer context lengths. As a result, they are more capable of generating answers directly from the final question, even without relying heavily on context. Unlike mathematical problems, factual QA often requires less logical reasoning, which may explain the stable performance. We leave a deeper investigation of this phenomenon for future work. Additionally, we plan to further validate our refinement approach using post-training methods; however, these experiments are currently ongoing due to their computational and time complexity.

F Experiments on Token Usage

In this section, we conduct several experiments to show the number of generation tokens. First, we evaluate token usage in prompt engineering (PE) using the Qwen series models on the Level 5 dataset. The results are shown in 7. We find that applying refinement increases token usage by approximately 50 tokens, which is expected, as the refined question typically adds that amount of content.

Table 7: Token usage comparison with and without refinement using Qwen models on the Level 5 dataset.

Model	CoT	Ours
Qwen2.5-1.5B	442.95	471.74
Qwen2.5-3B	443.16	489.70
Qwen2.5-7B	418.86	462.30
Qwen2.5-72B	379.36	447.05

Second, we further evaluate token usage in post-trained models. The results are presented in Table 8. We observe that the post-trained models exhibit similar token usage patterns to those in the PE experiments. The increase in token count is primarily attributed to the question refinement process.

Table 8: Token usage analysis in post-trained models with and without refinement.

Model	CoT		Ours		
	think%	total	refinement%	think%	total
Qwen2.5-3B-SFT	0.9351	262	0.2339	0.7251	342
Qwen2.5-3B-RL	0.7702	322	0.1920	0.7421	349
LLaMA3.2-3B-SFT	0.6651	427	0.2729	0.5706	524
LLaMA3.2-3B-RL	0.8069	439	0.2038	0.7958	517

Third, we evaluate the reasoning model (e.g., QwQ-32B) across multiple noise level datasets. The results are presented in Table 9. Interestingly, our refinement reduces the number of generated tokens to some extent, supporting the interpretation that our method functions as a meta-cognitive step—helping the model clarify its reasoning pathway before executing detailed chain-of-thought (CoT) reasoning.

Table 9: Token usage of QwQ-32B across different levels with and without refinement.

QwQ-32B	CoT	Ours
Original	1045.95	1012.98
Level 1	940.73	937.42
Level 2	950.88	964.15
Level 3	970.47	975.33
Level 4	990.23	974.65
Level 5	980.58	862.18

Fourth, we observe mixed results on closed-source models regarding token usage after applying our method. For ChatGPT 3.5, token usage increases consistently across all difficulty levels: for example, at Level 1, token usage rises from 227 (CoT) to 272 (Ours), and at Level 5, it increases from 246 to 293. A similar upward trend is observed for Claude 3.5, where token usage grows from 275 to 318 at Level 1 and from 282 to 325 at Level 5. We exclude ChatGPT o3-mini from this analysis, as it typically returns only the final answer without intermediate reasoning steps, making its token usage incomparable. However, the open-source reasoning model QwQ-32B, when combined with our method, exhibits a modest reduction in the number of generated tokens. In contrast, for ChatGPT 4o and ChatGPT 4o-mini, our question

refinement leads to a consistent reduction in token usage. For instance, ChatGPT 4o's token count at Level 3 decreases from 341 (CoT) to 333 (Ours), and at Level 5, from 343 to 323. Similarly, ChatGPT 4o-mini shows a drop from 453 to 417 at Level 5. We leave a more thorough investigation on token usage to the future work.

Table 10: Token usage across different levels for each model, with and without refined questions (rq).

Model	Method	Original	Level 1	Level 2	Level 3	Level 4	Level 5
ChatGPT 3.5	CoT	218	227	229	256	261	246
	Ours	254	272	283	284	294	293
ChatGPT 4o	CoT	490	416	483	341	406	343
	Ours	447	408	403	333	327	323
ChatGPT 4o-mini	CoT	485	422	456	423	425	453
	Ours	413	415	418	421	426	417
ChatGPT o3-mini	CoT	227	241	244	253	273	261
	Ours	289	299	300	311	316	311
Claude 3.5	CoT	267	275	277	280	286	282
	Ours	312	318	317	319	325	325

G Experiments on Out-of-Domain Tasks

In this section, we evaluate the post-trained models (e.g., LLaMA3.2-3B-SFT and LLaMA3.2-3B-RL) introduced in our paper on additional datasets, including GSM8K, AIME 2024, and GPQA. The results are summarized as follows.

Table 11: Evaluation on OOD tasks (GSM8K, AIME 2024, GPQA) using LLaMA3.2-3B SFT and RL.

Model	Method	GSM8K	AIME2024	GPQA
LLaMA3.2-3B-SFT	CoT	70.29%	3.75%	20.71%
	Ours	71.74%	3.96%	28.28%
LLaMA3.2-3B-RL	CoT	71.38%	3.56%	21.95%
	Ours	72.34%	4.31%	31.31%

We find that models incorporating our refinement method consistently outperform their non-refinement counterparts on OOD tasks. Specifically, our refinement yields an accuracy improvement of approximately 1–2% on GSM8K and AIME 2024, and a notable absolute gain of around 10% on the GPQA benchmark. These results highlight the effectiveness of our method in enhancing model generalization on OOD datasets.

H Experiments on Impact of Each Error Type

In this section, we conduct additional ablation experiments to analyze the impact of each error type. These experiments are performed on both PE and post-trained models. For the PE setting, we evaluate LLaMA3.2-3B and LLaMA3.1-72B. For post-trained models, we evaluate LLaMA3.2-3B-SFT and LLaMA3.2-3B-RL.

Impact of Grammar Errors. In this experiment, we fix the rates of all other error types to 0 and vary the grammar error rate from 0 (i.e., the original column) to 1.0. The performance of PE is shown in Table 12, while the results of post-trained methods are presented in Table 13.

The performance of PE models exhibits a gradual decline as the grammar error rate increases. For instance, LLaMA3.2-3B under our method drops from 24.77% to 21.46% at a full error rate, while the CoT baseline declines from 22.95% to 20.54%. A similar trend is observed for the larger LLaMA3.1-72B

Table 12: Impact of grammar error rate for LLaMA3.2-3B and LLaMA3.1-72B using PE.

Model	Method	Original	0.2	0.4	0.6	0.8	1.0
LLaMA3.2-3B	CoT	22.95%	23.06%	23.06%	21.50%	20.23%	20.54%
	Ours	24.77%	24.59%	23.85%	23.22%	21.70%	21.46%
LLaMA3.1-72B	CoT	77.27%	77.27%	76.54%	76.13%	75.60%	75.91%
	Ours	76.59%	77.01%	77.26%	75.05%	76.98%	76.83%

Table 13: Impact of grammar error rate for LLaMA3.2-3B-SFT and LLaMA3.2-3B-RL.

Model	Original	0.2	0.4	0.6	0.8	1.0
LLaMA3.2-3B-SFT	49.54%	49.87%	46.10%	43.02%	44.23%	44.64%
LLaMA3.2-3B-RL	55.36%	54.05%	50.18%	49.55%	49.55%	49.18%

model, although it demonstrates greater robustness to grammar errors; its performance under our method remains relatively stable, even achieving slight fluctuations (e.g., 77.26% at 0.4 error rate vs. 76.59% at 0). This suggests that larger models may exhibit increased tolerance to isolated syntactic distortions. In post-trained models (Table 13), both LLaMA3.2-3B-SFT and LLaMA3.2-3B-RL show noticeable performance drops as grammar errors increase, with the RL variant being slightly more robust overall.

Impact of irrelevant information. In this experiment, we fix the rates of all other error types to 0 and vary the irrelevant information rate from 0 (i.e., original column) to 1.0. The performance of PE is shown in Table 14, while the results of post-trained methods are presented in Table 15.

Table 14: Impact of irrelevant information error rate for LLaMA3.2-3B and LLaMA3.1-72B using PE.

Model	Method	Original	0.2	0.4	0.6	0.8	1.0
LLaMA3.2-3B	CoT	22.95%	23.06%	23.06%	21.50%	18.23%	15.54%
	Ours	24.77%	24.59%	23.85%	23.22%	21.70%	19.46%
LLaMA3.1-72B	CoT	77.27%	77.27%	77.73%	71.36%	71.82%	68.64%
	Ours	76.59%	77.03%	77.46%	73.82%	72.18%	70.26%

Table 15: Impact of irrelevant information error rate for LLaMA3.2-3B-SFT and LLaMA3.2-3B-RL.

Model	Original	0.2	0.4	0.6	0.8	1.0
LLaMA3.2-3B-SFT	49.54%	47.35%	46.26%	46.78%	43.33%	42.94%
LLaMA3.2-3B-RL	55.36%	51.36%	50.05%	49.09%	47.55%	47.29%

The presence of irrelevant information substantially degrades performance for both PE models. For LLaMA3.2-3B, performance under our method drops from 24.77% to 19.46% as the irrelevant information rate increases from 0 to 1.0, while the CoT baseline exhibits an even steeper decline from 22.95% to 15.54%. The larger LLaMA3.1-72B model is more robust, but still experiences a performance drop: under our method, accuracy decreases from 76.59% to 70.26%, and under CoT, from 77.27% to 68.64%. This suggests that even large models struggle with distractive content, though their degradation is less severe. In the post-trained models, both SFT and RL variants of LLaMA3.2-3B also show consistent performance degradation. The RL model drops from 55.36% to 47.29%, while the SFT model declines from 49.54% to 42.94%. These findings indicate that irrelevant information is one of the most disruptive error types, highlighting the importance of input clarity and focus for reliable reasoning performance.

Impact of symbol error. In this experiment, we fix the rates of all other error types to 0 and vary the

irrelevant information rate from 0 (i.e., original column) to 0.040. The performance of PE is shown in Table 16, while the results of post-trained methods are presented in Table 17.

Table 16: Impact of symbol error rate for LLaMA3.2-3B and LLaMA3.1-72B using PE.

Model	Method	Original	0.008	0.016	0.024	0.032	0.040
LLaMA3.2-3B	CoT	22.95%	22.02%	19.67%	18.64%	15.21%	11.26%
	Ours	24.77%	25.79%	22.00%	20.67%	17.56%	14.79%
LLaMA3.1-72B	CoT	77.27%	78.15%	76.49%	73.18%	70.33%	66.16%
	Ours	76.59%	76.20%	75.87%	73.45%	71.56%	68.30%

Table 17: Impact of symbol error rate for LLaMA3.2-3B-SFT and LLaMA3.2-3B-RL.

Model	Original	0.008	0.016	0.024	0.032	0.040
LLaMA3.2-3B-SFT	49.54%	50.24%	49.76%	47.64%	45.91%	42.57%
LLaMA3.2-3B-RL	55.36%	54.89%	53.32%	51.00%	49.67%	47.16%

We find that even small amounts of symbol corruption lead to substantial performance degradation across all PE models. For LLaMA3.2-3B, our method's performance drops sharply from 24.77% to 14.79% as the symbol error rate increases to 0.040, while the CoT baseline falls even more drastically from 22.95% to 11.26%. A similar trend is observed in LLaMA3.1-72B, though the degradation is somewhat less severe—its accuracy under our method decreases from 76.59% to 68.30%. These results suggest that symbol-level noise, such as altered operators or variables, is particularly disruptive to reasoning, likely due to its interference with mathematical semantics and logic parsing. In post-trained models, we observe comparable patterns: LLaMA3.2-3B-SFT declines from 49.54% to 42.57%, and the RL variant drops from 55.36% to 47.16%. Compared to grammar errors and irrelevant information, symbol errors consistently lead to the steepest performance decline across both PE and post-trained settings. This underscores the sensitivity of LLMs to even minor perturbations in symbolic expressions, highlighting the need for enhanced robustness to syntactic and semantic noise.

Table 18: Impact of input shuffle on LLaMA3.2-3B and LLaMA3.1-72B using PE.

Model	Method	Original	w/ shuffle
LLaMA3.2-3B	CoT	22.95%	18.67%
	Ours	24.77%	21.26%
LLaMA3.1-72B	CoT	77.27%	75.46%
	Ours	76.59%	75.98%

Impact of shuffling. In this experiment, we fix the rates of all other error types to 0 and compare the performance with and without shuffling. The performance of PE is shown in Table 18, while the results of post-trained methods are presented in Table 19.

Table 19: Impact of input shuffle on LLaMA3.2-3B-SFT and LLaMA3.2-3B-RL.

Model	Original	w/ shuffle
LLaMA3.2-3B-SFT	49.54%	45.68%
LLaMA3.2-3B-RL	55.36%	52.97%

Shuffling leads to a modest performance drop in PE models. For LLaMA3.2-3B, our method sees a decrease from 24.77% to 21.26%, while the CoT baseline drops from 22.95% to 18.67%. The larger

LLaMA3.1-72B model demonstrates greater resilience, with less than 2% degradation under both methods. Similarly, in post-trained models (Table 19), LLaMA3.2-3B-SFT and RL show only slight declines (e.g., 49.54% to 45.68% for SFT, and 55.36% to 52.97% for RL). When viewed alongside grammar errors, irrelevant information, and symbol errors, shuffling shows a modest disruptive perturbation. This may be because shuffled inputs retain all original content and, in many cases, allow the model to recover meaning through global reasoning or attention mechanisms.

In conclusion, in the case of LLaMA3.2-3B, we observe that irrelevant information and symbol errors have the most significant negative impact, while grammar errors and shuffling lead to more modest performance degradation. LLaMA3.1-72B exhibits a similar trend, demonstrating greater robustness overall but following the same relative ranking of error severity. Across both PE and post-trained models, symbol errors consistently cause the steepest performance drop, likely due to their disruption of core logical and mathematical structures. Irrelevant information also degrades performance substantially, particularly in smaller models, by introducing semantic noise that distracts from the reasoning task. In contrast, shuffling results in more gradual declines, and grammar errors prove to be the least harmful, suggesting that models can often recover from syntactic disfluency or reordered content. These findings highlight the varying sensitivities of reasoning models to different types of noise and underscore the importance of designing robust models capable of handling symbolic and semantic perturbations. We leave a more detailed investigation to future work.

I Case Study

In this section, we present example responses from QwQ-32B, Qwen2.5-3B with prompt engineering, and Qwen2.5-3B after post-training (i.e., Qwen2.5-3B-RL).

I.1 Example Responses of QwQ-32B

We present the essential portions of the QwQ-32B response with and without reformulating the problem, as the experimental setups are comprehensively detailed in §6. The original problem is colored in blue. The key part of the response where the model parses the original problem and does refinement is highlighted in violet.

USER: (Omit prompt instruction here) Nadine andw̄ere Cindy given the same amount of allowance by their mother . Nadine and Cindy bes both in grade 5 . The stor?!e was offeri*ng a di@**scount on candles that day . Nadine and Cindy thought about buying a cake wiht a cost? //of 8but decided not to . The two girls d!ecide to combi/ne their allowance to surprise their| father on hi s birth/#_day . They buy a book whi+%ch costs $ 5 . After buyi&ng +boo/k and c andles , the remianing money was used to bũy 4 packets of cookies for $ 11 each . It aws a sunny day when Nadine a@-nd Cindy went to the store . Their favorite subject i+!s . scien +ce They also bought 48 ca‘ndle+s which were s=old fro $ 1.0 for 4 candles . How much did Nadine and Cindy’s mother give each one of them?

QWQ-32B: Okay, let’s see. I need to figure out how much allowance Nadine and Cindy each got from their mom. The problem says they were given the same amount, so whatever the total is, we can divide by two to get each person’s allowance.
First, let me parse all the information given here. The problem is a bit jumbled with some weird symbols and formatting issues, but I’ll try to extract the key points step by step.
...
Wait, the problem says "After buying the book and candles, the remaining money was used to buy 4 packets of cookies for $11 each." Wait, but the original thought was to buy a cake but they didn’t. So maybe the candles they bought are separate?
...
But to find the cost of candles, the problem says: "They also bought 48 candles which were sold for $1.0 for 4 candles."
...
Wait, but the problem says "After buying the book and candles, the remaining money was used to buy 4 packets of cookies for $11 each." So that implies that the cookies were bought with the remaining money after the book and candles. Therefore, the total money they had must be equal to the sum of the book, candles, and cookies. So total money is indeed 5 +12 +44 = 61. Since they each got the same amount, each got 61/2 = $30.50.
...
Answer:

$$\boxed{30.50}$$

Now, we show the essential portions of QwQ-32B’s response with problem refinement.

USER: (Omit prompt instruction here) Nadine andw̄ere Cindy given the same amount of allowance by their mother . Nadine and Cindy bes both in grade 5 . The stor?!e was offeri*ng a di@**scount on candles that day . Nadine and Cindy thought about buying a cake wiht a cost? //of 8but decided not to . The two girls d!ecide to combi/ne their allowance to surprise their| father on hi s birth/#_day . They buy a book whi+%ch costs $ 5 . After buyi&ng +boo/k and c andles , the remianing money was used to bũy 4 packets of cookies for $ 11 each . It aws a sunny day when Nadine a@-nd Cindy went to the store . Their favorite subject i+!s . scien +ce They also bought 48 ca‘ndle+s which were s=old fro $ 1.0 for 4 candles . How much did Nadine and Cindy’s mother give each one of them?

QWQ-32B: Okay, let me try to figure this out. The problem is a bit messy with all the symbols and extra info, so first I need to clean it up. Let me read through it again and pick out the relevant parts.

So, Nadine and Cindy got the same allowance from their mom. They both are in 5th grade. The store had a discount on candles, but they decided not to buy a cake that cost $8. They used their combined allowance to buy a book for $5 and some candles. Then they bought 4 packets of cookies at $11 each. Also, they bought 48 candles which were $1 for 4 candles. The question is how much each got from their mom.

Hmm, okay. Let me list the key points:

- Both have the same allowance.
- They combined their money to buy gifts for their dad's birthday.
- They bought a book costing $5.
- 4 packets of cookies at $11 each. So 4 x 11 = $44 for cookies.
- 48 candles bought at $1 for 4 candles. So how much did the candles cost? Let's see: 48 candles divided by 4 per dollar is 12, so 12 dollars for candles.
- They decided not to buy the cake that was $8, so that's not part of the expenses.
- After buying the book, candles, and cookies, the remaining money was used for the cookies?

Wait, no. Wait, the problem says "After buying the book and candles, the remaining money was used to buy 4 packets of cookies for $11 each." Wait, maybe I misread that part. Let me check again.

...

Which is the same as adding all expenses. So total expenses are indeed 5 + 12 + 44 = 61. Therefore, their combined allowance is $61. Since they each got the same amount, each got 61/2 = $30.50.

Answer:

$$\boxed{30.50}$$

We observe that QwQ-32B exhibits distinct problem-solving strategies depending on whether problem refinement is applied. Without refinement, the model adopts a statement-level approach—meticulously correcting grammatical and symbolic errors line by line while conducting a detailed step-by-step analysis. It engages in self-reflection and verification, ultimately reorganizing the entire problem structure to arrive at the correct solution. In contrast, when refinement is performed upfront, QwQ-32B shifts to a problem-level strategy, leading to a more streamlined reasoning process and significantly reduced token usage. This contrast suggests that early refinement may offload low-level correction effort and facilitate more efficient reasoning. We plan to further investigate whether combining our problem refinement with long CoT can synergistically improve both accuracy and efficiency.

I.2 Example Responses of Qwen2.5-3B

We present the Qwen2.5-3B response with and without problem refinement, as the experimental setups are comprehensively detailed in §6. The refined The original problem is colored in blue. The key part of the response where the model parses the original problem and does refinement is highlighted in violet. Incorrect reasoning steps in the response are highlighted in red.

USER: (Omit prompt instruction here) There bes 40 students in Nina 's class% . Nin+#˜a loves to play the piano*# . Each students in N#ina 's friend class started with 5 penci?/ls . Each ?s$tud!ents started last year with 8 penclis ?. Each student started the year with 6 pencils . Nina plans to give each st#/˜udent 4 more pencils next year . After two months , 1/5 of the total pˆencils in class were used /. Nina has a pet ham&ster . Each student al&so received 3 notebooks at the beginning the of year . At the end of year , only 1/2 of˜ the remaining pencils were left . How many pencils were left?

QWEN2.5-3B: To solve this problem, we need to carefully follow the steps and calculations provided. Let's break it down step by step.

1. Determine the initial number of pencils each student had:
- Each student started with 8 pencils.
- There are 40 students in the class.
- Therefore, the total number of pencils initially is:

$$8 \text{ pencils/student} \times 40 \text{ students} = 320 \text{ pencils}$$

2. Calculate the number of pencils given to each student next year:
- Each student will receive 4 more pencils.
- Therefore, the total number of pencils each student will have next year is:

$$8 \text{ pencils} + 4 \text{ pencils} = 12 \text{ pencils}$$

3. Calculate the total number of pencils after the first year:
- Each student will have 12 pencils.
- There are 40 students.
- Therefore, the total number of pencils after the first year is:

$$12 \text{ pencils/student} \times 40 \text{ students} = 480 \text{ pencils}$$

4. Calculate the number of pencils used after two months:
- $\frac{1}{5}$ of the total pencils were used.
- Therefore, the number of pencils used is:

$$\frac{1}{5} \times 480 = 96 \text{ pencils}$$

5. Calculate the number of pencils remaining after two months:
- The total number of pencils after the first year is 480.
- 96 pencils were used.
- Therefore, the number of pencils remaining is:

$$480 - 96 = 384 \text{ pencils}$$

6. Calculate the number of pencils remaining at the end of the year:
- At the end of the year, only $\frac{1}{2}$ of the remaining pencils were left.
- Therefore, the number of pencils remaining at the end of the year is:

$$\frac{1}{2} \times 384 = 192 \text{ pencils}$$

Answer:

$$\boxed{192}$$

Now, we show the Qwen2.5-3B's response with problem refinement.

USER: (Omit prompt instruction here) Nadine andw̄ere Cindy given the same amount of allowance by their mother . Nadine and Cindy bes both in grade 5 . The stor?!e was offeri*ng a di@**scount on candles that day . Nadine and Cindy thought about buying a cake wiht a cost? //of 8but decided not to . The two girls d!ecide to combi/ne their allowance to surprise their| father on hi s birth/#_day . They buy a book whi+%ch costs $ 5 . After buyi&ng +boo/k and c andles , the remianing money was used to bũy 4 packets of cookies for $ 11 each . It aws a sunny day when Nadine a@-nd Cindy went to the store . Their favorite subject i+!s . scien +ce They also bought 48 ca'ndle+s which were s=old fro $ 1.0 for 4 candles . How much did Nadine and Cindy's mother give each one of them?

QWEN2.5-3B: Let's refine the math problem logically and eliminate irrelevant information:

Refined Problem:

There are 40 students in Nina's class.
Each student in Nina's class started with 6 pencils.
Nina plans to give each student 4 more pencils next year.
After two months, $\frac{1}{5}$ of the total pencils in class were used.
At the end of the year, only $\frac{1}{2}$ of the remaining pencils were left.
We need to find out how many pencils were left at the end of the year.

Solution:

First, calculate the total number of pencils at the beginning of the year:

$$6 \text{ pencils/student} \times 40 \text{ students} = 240 \text{ pencils}$$

After two months, $\frac{1}{5}$ of the pencils were used:

$$\frac{1}{5} \times 240 = 48 \text{ pencils}$$

Pencils remaining after two months:

$$240 - 48 = 192 \text{ pencils}$$

At the end of the year, only $\frac{1}{2}$ of the remaining pencils were left:

$$\frac{1}{2} \times 192 = 96 \text{ pencils}$$

Answer:

$$\boxed{96}$$

The results indicate that without problem refinement, Qwen2.5-3B is prone to distraction from irrelevant information. As shown in the highlighted example, the model is misled by mentions such as "the pencil each student had last year," leading to incorrect calculations. When problem refinement is applied, the model—despite not filtering out all irrelevant content—maintains focus on the core information and successfully computes the correct answer. This suggests that LLMs possess a degree of innate problem refinement ability acquired during pre-training, even in the absence of explicit post-training for this skill.

I.3 Example Responses of Qwen2.5-3B-RL

We present the response of Qwen2.5-3B with and without problem refinement. The original problem is colored in blue. The key part of the response where the model parses the original problem and does refinement is highlighted in violet. Incorrect reasoning steps in the response are highlighted in red.

USER: (Omit prompt instruction here) A bumper car rink has 18 pink . cars The ri+-nk also some has green caˆ@rs . green cars are the oldest ones in the‘ rink . The rink ahs 6 fewer orang&e cars than-ˆ they have pink cars . It was raining on the day the count was made. The r‘=%ink o+wner#¡ also own##s arcade center nearby . The rink has 4 times the number of red cars as they have orange cars . The pink c/?ars$ are most popular among kids . A new shipment of 11 gray cars will arrive next week. If the rink has 92 cars in total now, how many green cars do they have?

QWEN2.5-3B: To solve the problem, let's denote the number of green cars as G, the number of orange cars as O, and the number of red cars as R.

From the problem, we have the following information:

1. The number of pink cars is 18.
2. The number of orange cars is 6 fewer than the number of pink cars.
3. The number of red cars is 4 times the number of orange cars.

4. The total number of cars in the rink is 92.
5. A new shipment of 11 gray cars will arrive next week.

Let's translate this information into equations:

- $O = 18 - 6 = 12$ (orange cars)
- $R = 4 \times O = 4 \times 12 = 48$ (red cars)

The total number of cars is:

$$18 \text{ (pink)} + 12 \text{ (orange)} + 48 \text{ (red)} + G \text{ (green)} + 11 \text{ (gray)} = 92$$

Now, solve for G:

$$90 + G = 92 \Rightarrow G = 92 - 90 = 2$$

Answer:

$$\boxed{2}$$

Next, we show the response of Qwen2.5-3B trained by SFT with problem refinement.

USER: (Omit prompt instruction here) A bumper car rink has 18 pink . cars The ri+-nk also some has green caˆ@rs . green cars are the oldest ones in the‘ rink . The rink ahs 6 fewer orang&e cars than-ˆ they have pink cars . It was raining on the day the count was made. The r‘=%ink o+wner#¡ also own##s arcade center nearby . The rink has 4 times the number of red cars as they have orange cars . The pink c/?ars$ are most popular among kids . A new shipment of 11 gray cars will arrive next week. If the rink has 92 cars in total now, how many green cars do they have?

QWEN2.5-3B: Let's refine the math problem first

Refined Problem:

A bumper car rink has 18 pink cars. The number of orange cars is 6 fewer than the number of pink cars. The number of red cars is 4 times the number of orange cars. The total number of cars in the rink is 92. How many green cars are there?

Solution:

1. **Define Variables:** Let $P = 18$ (pink cars), Let O = number of orange cars, Let R = number of red cars, Let G = number of green cars.
2. **Given Information:**

$$O = P - 6 = 18 - 6 = 12$$

$$R = 4 \times O = 4 \times 12 = 48$$

$$\text{Total cars: } P + O + R + G = 92$$

3. **Substitute and Solve:**

$$18 + 12 + 48 + G = 92$$

$$\Rightarrow 78 + G = 92 \Rightarrow G = 92 - 78 = 14$$

Answer:

$$\boxed{14}$$

Now, we show the response of Qwen2.5-3B trained by RL with problem refinement.

USER: (Omit prompt instruction here) A bumper car rink has 18 pink . cars The ri+-nk also some has green caˆ@rs . green cars are the oldest ones in the‘ rink . The rink ahs 6 fewer orang&e cars than-ˆ they have pink cars . It was raining on the day the count was made. The r‘=%ink o+wner#¡ also own##s arcade center nearby . The rink has 4 times the number of red cars as they have orange cars . The pink c/?ars$ are most popular among kids . A new shipment of 11 gray cars will arrive next week. If the rink has 92 cars in total now, how many green cars do they have?

QWEN2.5-3B: Let's refine the math problem first

Refined Problem:

A bumper car rink has 18 pink cars. The rink also has some green cars. The rink has 6 fewer orange cars than the number of pink cars. It has 4 times as many red cars as orange cars. A new shipment of 11 gray cars will arrive next week. If the rink has 92 cars in total **now**, how many green cars are there?

Solution:

Let the variables be defined as:

- P = number of pink cars = 18
- O = number of orange cars = $P - 6 = 18 - 6 = 12$
- R = number of red cars = $4 \times O = 4 \times 12 = 48$
- G = number of green cars (to be determined)

The total number of cars currently is given as:

$$P + O + R + G = 92$$

Substitute the known values:

$$18 + 12 + 48 + G = 92$$

$$\Rightarrow 78 + G = 92 \Rightarrow G = 92 - 78 = 14$$

Answer:

$$\boxed{14}$$

As highlighted in red, without problem refinement, Qwen2.5-3B erroneously incorporates irrelevant information—specifically, the new shipment of 11 gray cars—into the total car count, leading to an incorrect answer. After post-training, however, the model is able to identify and exclude all irrelevant details, restructuring the problem statement in a more logical and concise manner—unlike the PE method, which falls short in this aspect. These results demonstrate that post-training methods significantly enhance the ability to refine problems, thereby improving performance on noisy inputs.