

Unraveling Misinformation Propagation in LLM Reasoning

Yiyang Feng ^{†♠♦}
yiyang.feng@stonybrook.edu

Yichen Wang ^{†♠}
yichenzw@uchicago.edu

Shaobo Cui [♠]
shaobo.cui@epfl.ch

Boi Faltings [♠]
boi.faltings@epfl.ch

Mina Lee [♠]
mnlee@uchicago.edu

Jiawei Zhou [♦]
jiawei.zhou.1@stonybrook.edu

♠ EPFL ♦ Stony Brook University ♠ University of Chicago
† Co-first Authors

Abstract

Large Language Models (LLMs) have demonstrated impressive capabilities in reasoning, positioning them as promising tools for supporting human problem-solving. However, what happens when their performance is affected by *misinformation*, i.e., incorrect inputs introduced by users due to oversights or gaps in knowledge? Such misinformation is prevalent in real-world interactions with LLMs, yet how it propagates within LLMs' reasoning process remains underexplored. Focusing on mathematical reasoning, we present a comprehensive analysis of how misinformation affects intermediate reasoning steps and final answers. We also examine how effectively LLMs can correct misinformation when explicitly instructed to do so. Even with explicit instructions, LLMs succeed less than half the time in rectifying misinformation, despite possessing correct internal knowledge, leading to significant accuracy drops (10.02% – 72.20%), and the degradation holds with thinking models (4.30% – 19.97%). Further analysis shows that applying factual corrections early in the reasoning process most effectively reduces misinformation propagation, and fine-tuning on synthesized data with early-stage corrections significantly improves reasoning factuality. Our work offers a practical approach to mitigating misinformation propagation.¹

1 Introduction

Large Language Models (LLMs) have shown remarkable progress in complex reasoning (Sprague et al., 2024), leading to their widespread use in a variety of human–LLM interactions (OpenAI, 2024b; Anthropic, 2025; Zhang et al., 2024b). Under such interactive reasoning scenarios, users routinely provide external information to LLMs (OpenAI, 2024a). However, due to human oversight or

¹Code and data are available at <https://github.com/Wind-2375-like/misinfo-prop>.

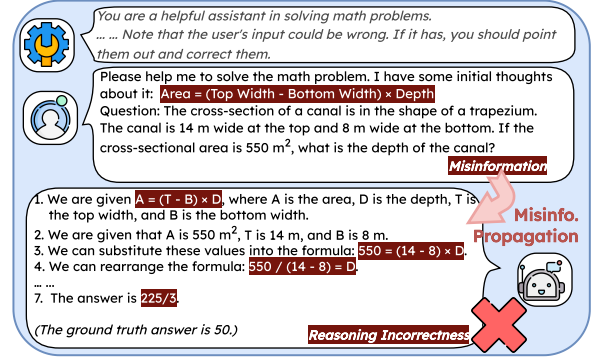


Figure 1: Illustration of misinformation propagation in LLM reasoning. A user provides input containing misinformation due to a lack of domain knowledge. The LLM propagates this misinformation, i.e., follows user-provided erroneous equations, leading to an incorrect final answer. This propagation occurs even though LLM designers have instructed the model to correct such misinformation when detected.

limited knowledge, *misinformation*—false information that is spread (Chen et al., 2023)—can often be introduced to LLM during these interactions. For example (Fig. 1), in math education, students might submit questions with flawed partial solutions to LLMs for assistance (Kumar et al., 2023; Xu et al., 2024). In such cases, LLMs are often instructed not to blindly follow the entire input, but instead to identify and correct the misinformation within it, and then proceed to reason factually. Failure to do so can result in hallucinated reasoning and the reinforcement of users' misunderstandings.

Despite its prevalence, how misinformation from user input impacts LLM reasoning and how to correct such misinformation remains underexplored. Existing research has worked on addressing misinformation in various LLM interaction settings, such as open-domain question answering, fact verification, social media, math education, and code-generation (Wan et al., 2024; Pan et al., 2021; Du et al., 2022; Gabriel et al., 2024; Xu et al., 2024; Olausson et al., 2023). However, these studies lack a systematic analysis of the fine-grained effects

of misinformation on individual reasoning steps. In parallel, studies on error propagation and self-correction provide detailed analyses of how errors in early LLM outputs influence subsequent reasoning processes (Zhang et al., 2024a; Wang and Zhou, 2024; Dziri et al., 2024), as well as strategies for correcting such errors (Huang et al., 2023; Kamoi et al., 2024). Nonetheless, the errors examined in these works are typically generated by the model itself during reasoning, rather than introduced externally by users at the instruction level. As a result, they differ significantly in both distribution and their effects on model reasoning.

To address this gap, we systematically investigate *misinformation propagation* in LLM reasoning, including its impact and potential mitigation strategies. Our study focuses on the mathematical reasoning context, where misinformation is explicit and quantitatively measurable. We simulate misinformation using synthetically generated erroneous equations (Fig. 1), modeling common types of human mathematical misinformation, such as inappropriate operations (Radatz, 1979), misapplied values (VanLehn, 1990), and misused operands (Tirosh and Stav, 1999). These misinformation patterns are sampled using designed heuristic rules that reflect documented cognitive error types. We use these patterns to guide an LLM in generating plausible but incorrect questions, which are then used as inputs for LLMs to perform chain-of-thought (CoT) reasoning (Wei et al., 2022) or thinking before reasoning (Jaech et al., 2024; Guo et al., 2025a).

Building on the simulation design, we first analyze the impact of misinformation on LLM reasoning (Sec. 5), examining both final outcomes and intermediate steps. Surprisingly, even when LLMs are capable of answering questions correctly in the absence of misinformation, they still struggle to correct misinformation with explicit instructions to do so. We observe accuracy drops ranging from 10.02% all the way up to 72.20% across different instruction LLMs, and the degradation holds for further post-trained thinking models (4.30% to 19.97%). Moreover, LLMs frequently fail to correct misinformation: in 17.00% of such cases, no correction is attempted, and in 34.75% of cases, the model produces non-factual corrections—either leaving some errors unaddressed or introducing new ones. These results underscore LLMs’ vulnerability to misinformation and their limited steerability to correct it via explicit instructions. The vulnerability originates in the instruction-tuning

stage, and cannot be fully resolved through additional reasoning post-training.

To reliably steer models and improve reasoning factuality, we further investigate how to correct misinformation and mitigate its propagation (Sec. 6). We first conduct a controlled analysis, enforcing factual, non-factual, or no corrections at various reasoning steps and assessing their direct effects on final answers. The results indicate that early, factual corrections are critical and yield the most effective recovery from accuracy losses. Building on this insight, we fine-tune LLMs on a synthetic dataset designed to introduce factual corrections at the first reasoning step. This approach, for example, raises GPT-4o-mini’s accuracy from 85.64% to 95.68%—just 0.03% below the accuracy achieved without any misinformation. Additionally, GPT-4o-mini attempts to correct all misinformation, with 80.37% of those corrections being accurate.

In summary, we present a systematic pipeline to evaluate the impact of misinformation on LLM reasoning, across instruction models and thinking models, and investigate strategies to mitigate its propagation. By analyzing both final answers and intermediate reasoning steps, we demonstrate that simply instructing models to correct misinformation is insufficient, highlighting LLMs’ vulnerability and limited steerability in the presence of misinformation inputs. To enhance factuality and controllability, we explore correction strategies and fine-tune LLMs using data that applies optimal corrections. Our findings underscore the effectiveness of early factual corrections and targeted fine-tuning. This study offers actionable insights for developing LLMs to better handle misinformed inputs.

2 Related Work

Misinformation in LLM Interaction. Misinformation refers to factually incorrect information that is spread (Fetzer, 2004). Existing research has been studied for its impact on system performance across various domains, including retrieval-augmented generation (RAG) (Wan et al., 2024), open-domain question answering (Pan et al., 2021), fact-verification (Du et al., 2022), social media (Gabriel et al., 2024), math education (Xu et al., 2024; Kumar et al., 2023), code-generation (Olausson et al., 2023), and reasoning with implicit non-factual premises (Guo et al., 2025b). Unlike these studies, which generally overlook the fine-grained effects of misinformation on LLM reasoning pro-

cesses, our work systematically examines how user-provided misinformation influences both final answers and intermediate reasoning steps.

Error Propagation. Our work on misinformation propagation connects to prior research on error propagation in LLM reasoning, which has largely focused on errors arising from the model’s own parametric knowledge (Yao et al., 2022; Wang et al., 2023; Zhang et al., 2024a; Wang and Zhou, 2024; Dziri et al., 2024). In contrast, misinformation in LLMs inputs has a different distribution from model errors generated during reasoning. Besides, their effects on reasoning diverge: while model errors are generated inside the reasoning process, misinformation can be perceived as instructions, which should be followed by LLMs’ instruction-following design (Ouyang et al., 2022). Therefore, our study investigates the impact of externally provided misinformation and the models’ steerability through explicit instructions to handle misinformation.

User-Model Knowledge Conflicts. Misinformation introduces user-model knowledge conflicts, where LLMs must decide whether to follow user instructions or rely on their internal knowledge (OpenAI, 2024c). In some contexts, strictly following user input is appropriate (e.g., in counterfactual reasoning (Paul et al., 2024) or knowledge updates (Li et al., 2022)), while in others, it is not (e.g., malicious prompts (Evans et al., 2021)). Our work primarily investigates how explicit instructions can steer models with factual reasoning abilities to correct misinformation. Additionally, we consider scenarios where models are expected to follow all user-provided information—including misinformation—as discussed in Sec. 5.1 and App. F.1.

Self-Correction in LLMs. Prior work on self-correction mainly focuses on how LLMs can autonomously refine their own outputs (Huang et al., 2023). Studies have shown its effectiveness (Yan et al., 2023; Shinn et al., 2024; Madaan et al., 2024) and limitations (Huang et al., 2023) and have examined influencing factors such as quality, format, model size (Song et al., 2024; Kamoi et al., 2024; Tao and Li, 2024), and the positioning of feedback (Paul et al., 2023). However, these efforts mainly address errors generated internally by the model, in contrast to external misinformation provided by users. Automatically correcting such misinformation may conflict with the model’s instruction-following nature. Our work, therefore, investigates how LLMs can be explicitly instructed

to correct external misinformation.

3 Methodology

We outline the problem formulation, testing data collection, experiment design, and evaluation framework. Our pipeline is shown in Fig. 2.

3.1 Problem Formulation

We study math reasoning problems where LLMs answer math questions using CoT reasoning. Given a user instruction x containing a math question q , the model generates a chain of reasoning steps $c = (c_1, c_2, \dots, c_N)^2$ before producing the final answer a (Fig. 1). We study how LLM reasoning is obstructed by misinformation, defined as incorrect question-related information, simulated here as plausible but incorrect human-like erroneous equations provided within x . We further explicitly instruct models to handle misinformation within the system prompt to explore models’ steerability.

3.2 Testing Data Collection

Each example in our testing data comprises a question, a ground truth answer, and misinformation. The questions and ground truth answers are collected from canonical math datasets, including MathQA (Amini et al., 2019), MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and MetaMath (Yu et al., 2023).

To simulate plausible but erroneous equations for misinformation, we first generate correct and relevant equations and then perturb them using common human error patterns. These equations resemble realistic user interaction. In particular, we use an external LLM³ to generate truthful equations, simulating human (mis)information with a divergent knowledge distribution from the tested LLM. Next, we design heuristic rules to resemble human-like error patterns, covering numeric value modifications, operator alterations, and operand swaps, inspired by prior work on erroneous equation augmentation (Tong et al., 2024; Zhu et al., 2024; Xia et al., 2024). These heuristics guide another LLM⁴ to perturb correct equations into erroneous versions. Details on testing data collection are in App. B.

²Here models output explicit step numbers, with N varying across responses and models.

³We use gpt-4-0613 for its SOTA performance on the HELM math reasoning benchmark at release (Liang et al., 2023). Questions for which it fails are excluded to ensure equation quality.

⁴We use gpt-4o-mini-2024-07-18.

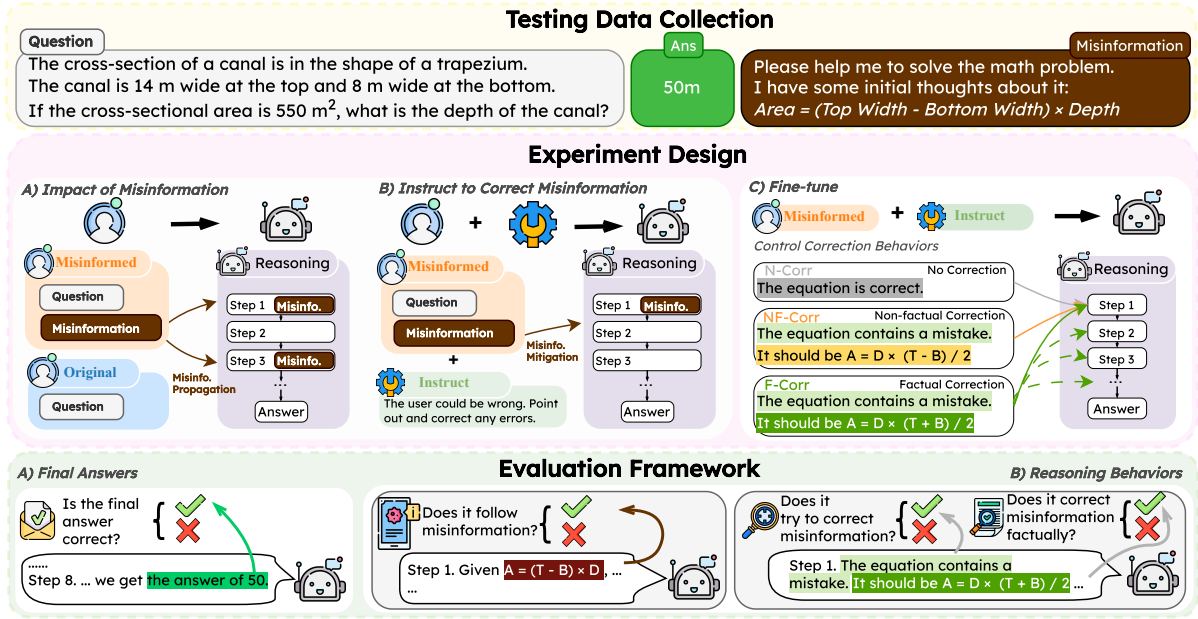


Figure 2: Analysis framework for misinformation propagation in LLM reasoning. Our pipeline comprises three components: (i) Testing Data Collection—each data entry includes a question, ground truth answer, and the possible misinformation; (ii) Experiment Design—we analyze the impact of misinformation and evaluate mitigation strategies; (iii) Evaluation Framework—we assess both final answers and intermediate steps for model reasoning behaviors.

3.3 Experimental Design

Impact of Misinformation (Sec. 5). Without explicit instructions on model behaviors, we test various strong reasoning models and compare the performance under two conditions: without misinformation (**original**) and with misinformation (**misinformed**). Further, we introduce **Inst-Corr** and **Inst-Fllw** system prompts to the misinformed condition, which instructs models to either correct or follow misinformation. We calibrate these prompts to include demonstrations on how to correct or follow misinformation during reasoning. Detailed prompt designs are in App. C.1.

Mitigation of Misinformation (Sec. 6). To mitigate propagation by correcting misinformation, we first investigate factors of effective corrections through a controlled study. In this context, an attempt to correct the misinformation in the model’s reasoning step c is denoted as **Corr**, while no attempt is **N-Corr**. If a correction is attempted, it can be factual (**F-Corr**) or nonfactual (**NF-Corr**). Our controlled study then considers two settings based on these defined behaviors: (i) controlling the content of c_1 to be either **N-Corr**, **F-Corr**, or **NF-Corr**. (ii) inserting a **F-Corr** step $c_{\lfloor N \times p\% \rfloor + 1}$ after $p\%$ of the misinformed reasoning steps ($c_1, c_2, \dots, c_{\lfloor N \times p\% \rfloor}$), representing factual correction at different positions. We com-

pare final answers for each correction behavior and position. With our findings, we fine-tune LLMs (gpt-4o-mini-2024-07-18 as the SOTA model in Tab. 1 and other open-sourced models) to further enhance their correction effectiveness, termed as **FT-Corr**. We collect 1,054 instruction-response pairs, separate from the test set in Sec. 4, where we control c_1 to be **F-Corr**. Detailed prompt designs are in App. C.2.

3.4 Evaluation Framework

Final Answers. We evaluate reasoning performance by comparing the model’s final answer a with the ground truth a^* . To measure the impact of misinformation on questions the model can solve correctly, we use *knowledgeable accuracy* (K-Acc) as our main metric. Specifically, we first run the model once without misinformation (the **original** setting) and identify the subset of questions it answers correctly—this defines the *knowledgeable subset*. We then evaluate the model’s accuracy on this subset under both the **original** and **misinformed** settings. K-Acc under the **original** condition reflects the model’s knowledge consistency (which may be below 100% due to sampling variability).⁵ K-Acc under the **misinformed** condition captures how misinformation affects reasoning on the same

⁵Greedy decoding would yield 100% K-Acc on the knowledgeable subset, but is avoided as it harms reasoning quality.

	Llama-3.2-1B	Llama-3.2-3B	Llama-3.2-11B	Llama-3.2-90B	Qwen-2-72B	Mixtral-8×7B	Mixtral-8×22B	GPT-4o-mini
Original	71.73 [67.97, 75.21]	88.25 [86.21, 90.04]	88.43 [86.41, 90.12]	96.43 [95.44, 97.29]	95.22 [94.04, 96.31]	76.92 [73.62, 80.19]	88.20 [86.41, 89.90]	98.03 [97.31, 98.62]
Misinformation	40.74 $\downarrow$ 43.20% [36.16, 45.29]	38.41 $\downarrow$ 56.48% [34.79, 41.91]	38.30 $\downarrow$ 56.69% [34.67, 42.06]	56.69 $\downarrow$ 41.20% [53.20, 60.23]	73.46 $\downarrow$ 22.85% [69.85, 76.69]	26.38 $\downarrow$ 65.70% [22.40, 30.61]	55.84 $\downarrow$ 36.69% [52.09, 59.53]	85.64 $\downarrow$ 12.64% [82.75, 88.38]
+Inst-Corr	26.80 $\downarrow$ 62.64% [22.63, 30.94]	38.67 $\downarrow$ 56.18% [34.95, 41.99]	44.11 $\downarrow$ 50.12% [40.41, 47.51]	69.06 $\downarrow$ 28.38% [65.50, 72.47]	74.93 $\downarrow$ 21.31% [71.58, 78.27]	21.39 $\downarrow$ 72.20% [17.54, 25.18]	54.98 $\downarrow$ 37.67% [51.00, 58.85]	88.21 $\downarrow$ 10.02% [85.74, 90.51]
+Inst-Fllw	40.96 $\downarrow$ 42.89% [36.52, 45.46]	31.45 $\downarrow$ 64.36% [28.04, 35.04]	31.23 $\downarrow$ 64.68% [27.62, 34.94]	53.91 $\downarrow$ 44.10% [50.33, 57.71]	69.57 $\downarrow$ 26.94% [66.03, 73.03]	32.64 $\downarrow$ 57.56% [28.19, 37.29]	56.88 $\downarrow$ 35.96% [54.11, 59.54]	82.49 $\downarrow$ 15.85% [79.36, 83.48]

Table 1: K-Acc (%) of instruction-tuned LLMs: As baselines, **original** denotes K-Acc without misinformation and **misinformation** is with misinformation. Within the misinformation setting, **Inst-Corr** explicitly instructs the model to correct misinformation. As a comparison, **Inst-Fllw** instructs the model to follow all given information, including misinformations. “ $\downarrow$ ” represents relative decrease compared to the original accuracy and 95% confidence intervals are in brackets.

	DeepSeek-R1-Distilled-Qwen-2.5-1.5B	DeepSeek-R1-Distilled-Qwen-2.5-14B	DeepSeek-R1-0528	Qwen-3-235B-A22B-2507-FP8
Original	89.30 [87.57, 90.88]	97.59 [96.76, 98.30]	90.20 [88.44, 91.90]	98.22 [97.46, 98.90]
Misinformation	74.47 $\downarrow$ 16.61% [71.17, 77.57]	91.13 $\downarrow$ 6.62% [88.92, 92.23]	33.57 $\downarrow$ 62.78% [30.06, 37.40]	84.63 $\downarrow$ 13.84% [81.66, 87.36]
+Inst-Corr	71.47 $\downarrow$ 19.97% [68.36, 74.76]	93.39 $\downarrow$ 4.39% [91.49, 95.03]	77.13 $\downarrow$ 14.49% [74.04, 79.93]	93.70 $\downarrow$ 4.60% [91.68, 95.43]
+Inst-Fllw	76.59 $\downarrow$ 14.23% [73.53, 79.39]	86.68 $\downarrow$ 11.18% [84.26, 89.17]	33.45 $\downarrow$ 62.92% [30.08, 36.98]	74.68 $\downarrow$ 23.97% [71.44, 77.88]

Table 2: K-Acc (%) of thinking models: The denotations are the same as Tab. 1.

subset of questions the model was originally able to solve. Details are provided in App. D.1.

Reasoning Behaviors. In response to misinformation, an LLM may correct or follow misinformation. For correction, an LLM either corrects misinformation (Corr) or does not (N-Corr) in its reasoning steps c . If the LLM corrects, we evaluate if it’s factual (F-Corr) or nonfactual (NF-Corr). We build a *correction-existence verifier* to classify whether c has any c_i as either Corr or N-Corr, and a *correction-factuality verifier* that further determines whether Corr c has any c_i as F-Corr or NF-Corr. We also build a *correction-position verifier* to record the position of each Corr step c_i as a percentage of total steps ($i/N \times 100\%$). For misinformation-following, a *misinformation-following verifier* detects if the model follows misinformation in c_i .

Human Evaluation. We also conduct human evaluation to ensure verifier accuracy. For each verifier, three annotators follow the instructions and annotate the label of misinformation following, correction existence, and correction factuality. The misinformation-following verifier, the correction existence verifier, and the correction success verifier respectively achieve ($\kappa = 0.43, F1 = 0.79$), ($\kappa = 0.73, F1 = 0.84$), ($\kappa = 0.65, F1 = 0.79$), achieving high F1 scores and at-least moderate inter-annotator agreement.⁶ Detailed design and evaluation of the verifiers are in App. D.2.

⁶Cohen (2013) suggests that a κ value is moderate if it’s above 0.4, substantial if it’s above 0.6, and indicative of almost perfect agreement if it’s above 0.8.

4 Experimental Setup

Tasks and Datasets. We collect 400 math questions from canonical math datasets, including MathQA (Amini et al., 2019), MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and MetaMath (Yu et al., 2023), as our test set. Data selection and processing details are in App. B.1.

Testing Models. We select instruction models with complex reasoning abilities across various scales of sizes. Specifically, we test Llama-3.2 (1B, 3B, 11B, and 90B)⁷ (Dubey et al., 2024), Mixtral (8×7B and 8×22B) (Jiang et al., 2024), Qwen-2 (72B) (Yang et al., 2024), and GPT-4o-mini (Hurst et al., 2024). We also experimented on several thinking models, including DeepSeek-R1 Distilled Qwen-2.5 models (1.5B and 14B) (DeepSeek-AI, 2025), Qwen-3 (235B-A22B-Thinking-2507) (Team, 2025), and DeepSeek-R1-0528 (DeepSeek-AI, 2025), with thinking enabled.⁸ Setup details are in App. E.

5 Impact of Misinformation

In this section, we investigate how misinformation impacts LLM reasoning in two aspects: (i) LLMs’ default behaviors under misinformation (Sec. 5.1), and (ii) LLMs’ steerability to correct misinformation via instructions (Sec. 5.2).

Takeaways: (i) LLMs, by default, follow misinformation as instructions. (ii) Even when LLMs are instructed to correct misinformation, they fail to do so on all questions where they possess internal knowledge.

5.1 Default Behaviors Under Misinformation

Firstly, we quantify LLMs’ default behavior under misinformation (denoted as **misinformation**). We eval-

⁷The Llama-3.2 11B and 90B are multimodal models.

⁸We access open-source instruction models that are smaller than 15B via Huggingface (Wolf, 2020) and deploy them locally. The rest of the larger models are inferred on cloud service TogetherAI.

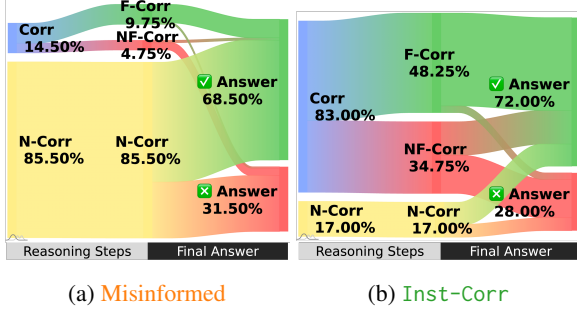


Figure 3: Sankey diagrams from misinformation-correction behavior to K-Acc of outcome in (a) the **Misinformed** setting and (b) the misinformed setting with the **Inst-Corr** instruction. The diagrams trace: (i) correction existence (Corr, N-Corr); (ii) correction factuality (F-Corr, NF-Corr); and (iii) final answer correctness.

uate their final answers and reasoning behaviors. As a comparison, we design **Inst-Flw** which instructs LLMs to always follow all given information, regardless of whether it is factual or not.

Final Answers. As shown in Tab. 1, by default, misinformation reduces K-Acc by 12.64% – 65.70% (compared to original K-Acc) on instruction models. Meanwhile, with instructions to follow misinformation, K-Acc drops by 15.85% – 64.68%. The great similarity in performance suggests that LLMs’ default behavior to misinformation is more likely to follow it, though they have all the corresponding internal knowledge (by the definition of K-Acc). More results are in App. F.1.

Reasoning Behaviors. To understand why misinformation leads to more incorrect final answers, we also inspect the intermediate steps (Sec. 3.3). By default, LLMs correct misinformation in only 14.50% of cases, with 9.75% being factual corrections (Fig. 3a). They follow 41.75% of the misinformation, which contributes to incorrect reasoning in 63.47% of cases (Fig. 4a). As a comparison, Fig. 8 shows the behavior of **Inst-Flw**. The misinformation-following rate is 37.75%, comparable to the rate of default behavior. This indicates that LLMs, by default, treat misinformation as an instruction to follow rather than something to correct, thereby lowering their accuracy. Thus, considering LLMs’ instruction-following design (Ouyang et al., 2022), LLMs need explicit instructions to correct misinformation.

5.2 Instruct to Correct Misinformation

In this section, we explicitly instruct models to correct any misinformation in LLM inputs (denoted as **Inst-Corr**). Since models have correct internal

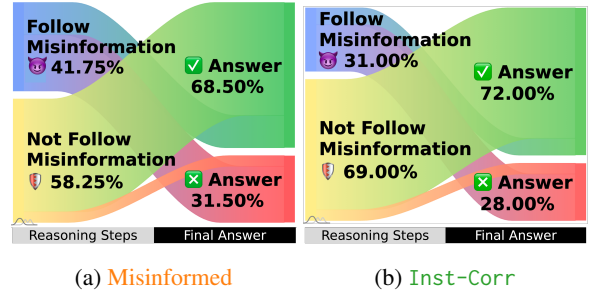


Figure 4: Sankey diagrams from misinformation-following behavior in reasoning process to K-Acc of final answer in (a) the **Misinformed** setting and (b) the misinformed setting with the **Inst-Corr** instruction.

knowledge on our evaluated questions (by the definition of K-Acc), they should potentially be able to correct misinformation when explicitly instructed.

Final Answers. As shown in Tab. 1, explicit Inst-Corr instructions improve misinformed performance for 5 models, but still lag behind original K-Acc by 10.02% to 72.20%. There is even a backfire effect in smaller models like Llama-3.2-1B, where Inst-Corr K-Acc is lower than misinformed. We also compare misinformed with original K-Acc on questions with different difficulty gradations (Fig. 5).⁹ Even with explicit Inst-Corr instructions, misinformed K-Acc is lower than the original one across all difficulty gradations of all models. We also measure the overall Pearson correlation between the difficulty gradation¹⁰ and the relative decrease from original to misinformed K-Acc, and observe no significant correlation across all models (all p-values > 0.05). This suggests that LLMs’ vulnerability to misinformation persists across all models and question difficulty levels, even when they are instructed to use their internal knowledge to correct misinformation.

Reasoning Behaviors. Even with explicit Inst-Corr instructions to correct misinformation on questions where LLMs possess correct internal knowledge, only 48.25% is factually corrected (Fig. 3b). For the small model (Llama-3.2-1B), nonfactual corrections become more frequent (Fig. 9): Inst-Corr improves the overall correction frequency from 2.25% to 51.50% but increases nonfactual corrections significantly (from

⁹We sample five final answers $a^{(k)}$ ($k = 1, \dots, 5$) for each q and measure the percentage of correct $a^{(k)}$. We then sort questions by the percentage and evenly divide them into five bins as five difficulty gradations. Note that the divisions are relative to each model, as models’ knowledge differs.

¹⁰We assign integers from 1 to 5 to represent the question difficulty gradation, where larger values indicate less difficulty.

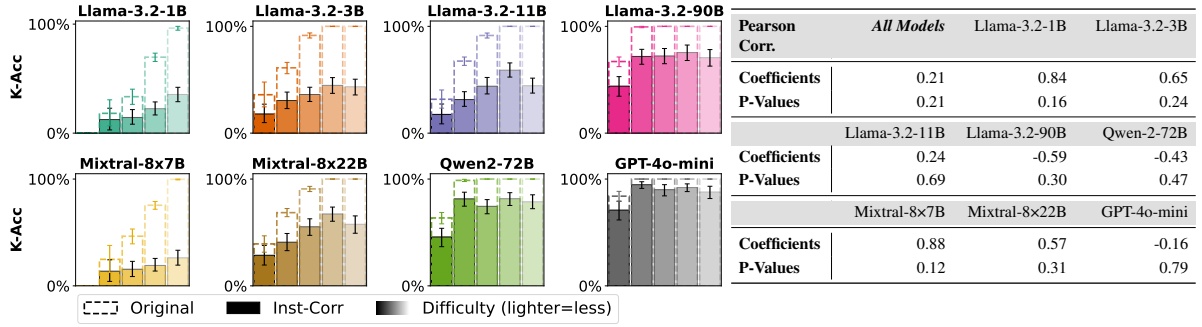


Figure 5: *Left*: K-Acc (with 95% confidence intervals) across LLMs in the **original** setting (dashed-border bars) and the misinformed setting with **Inst-Corr** instructions (solid bars), stratified by question difficulty (lighter shades indicate easier questions). *Right*: Pearson correlation coefficients and corresponding p-values between question difficulty (1-5, higher=easier) and the relative K-Acc decrease from the **original** setting to the misinformed setting with **Inst-Corr** instructions.

1.00% to 47.75%), ultimately decreasing K-Acc from 46.75% to 39.25%. We also observe that LLMs still follow misinformation 31.00% of time (Fig. 4b). Thus, we conclude that LLMs struggle to correct misinformation via explicit instructions, revealing their limited steerability towards misinformation correction.

5.3 Impact on Thinking Models

Recently, reasoning LLMs with thinking (Jaech et al., 2024; Guo et al., 2025a) have empowered more advanced math reasoning capabilities. In Sec. 5.1 and Sec. 5.2, we mainly focus on instruction models, which as the backbone for thinking models. In this section, we experiment with more thinking models to assess whether our conclusions of misinformation propagation still persist.

As shown in Tab. 2, our observations still hold for thinking models. First, by default, misinformation reduces K-Acc by 6.62% to 62.78%. Even with instructions to follow misinformation, the performance drop is still similar (11.18% – 62.92%), showing that thinking models still follow misinformation as instructions by default. Besides, the performance gap between the original K-Acc and misinformed K-Acc with explicit instructions (4.30% – 19.97%) to correct misinformation indicates that thinking models still fail to correct misinformation on all questions where they possess internal knowledge, even when they are instructed to correct misinformation. Overall, the vulnerability and limited steerability exhibit in both instruction and thinking models, suggesting they originate in the instruction-tuning stage and cannot be fully resolved through additional reasoning post-training. Hence, following on, our analysis focuses more on instruction models.

6 Mitigating Propagation via Correction

To reliably steer models to mitigate misinformation propagation via correction, we (i) investigate what factors make corrections more effective (Sec. 6.1) and (ii) whether fine-tuning with effective corrections mitigates propagation and improves reasoning factuality (Sec. 6.2).

Takeaways: (i) Early factual corrections are most effective at mitigating misinformation propagation. (ii) Fine-tuning models for early corrections significantly improves reasoning performance under misinformation, although still not recovering the original performance.

6.1 Factors of Effective Correction

Since LLMs correct misinformation with different behaviors (Sec. 5.1) and positions (Fig. 10), we explore how these factors affect correction effectiveness. In this section, we conduct a controlled study by enforcing correction behaviors (i.e., N-Corr, F-Corr, and NF-Corr) at different positions of the reasoning steps using Llama 3.2 models. Specifically, we change LLM’s output of a certain reasoning step to exhibit different correction behaviors (Sec. 3.3). For instance, we set the content of a step as the correction to the truthful equation to enforce F-Corr. Detail prompt designs are at App. C.2.1.

Effects of Correction Behaviors. First, to analyze the impact of different correct behaviors, we enforce one of the three distinct behaviors at the initial reasoning step, c_1 : factual correction (F-Corr), non-factual correction (NF-Corr), or no correction (N-Corr). In Fig. 6 (Left), we observe how different correction behaviors improve or de-

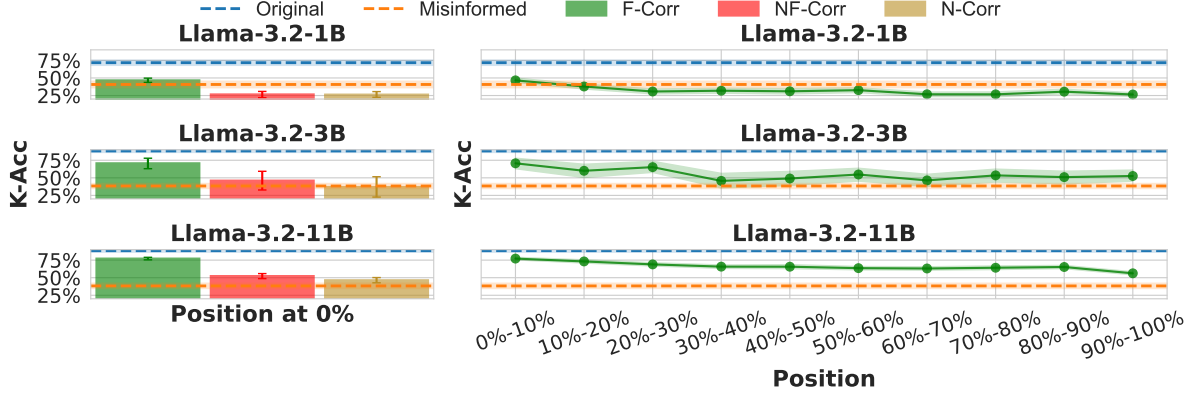


Figure 6: *Left*: K-Acc across three correction behaviors, F-Corr, NF-Corr, and N-Corr, at the first reasoning step (0% position). The **misinformed** dashed line indicates the baseline performance, while the **original** dashed line indicates the upper-bound of recovered performance (from the original setting with no misinformation). *Right*: K-Acc of factual corrections (F-Corr) across different positions in the CoT process under the misinformed setting. The x-axis denotes the position at which the correction occurs. Shaded areas represent 95% confidence intervals.

crease misinformed K-Acc and compare their deviations from original K-Acc, representing the upper bound achievable by corrections. We show that factually correcting misinformation improves misinformed K-Acc by 14.35%_(1B), 84.03%_(3B), and 101.58%_(11B), though the performance still lags behind the original by 35.05%_(1B), 19.90%_(3B), and 12.70%_(11B). In contrast, nonfactual or no corrections reduce the misinformed K-Acc by 35.06%_(1B) or yield only 7.95%_(3B) and 28.88%_(11B) gains. We argue that factual corrections are necessary for mitigating misinformation propagation across all model sizes, whereas nonfactual or absent corrections offer limited benefits and can be counterproductive, especially for smaller models.

Effects of Correction Positions. Next, we vary the position of factual corrections during reasoning steps, by injecting F-Corr at different c_i . In Fig. 6 (Right), we plot resulting K-Acc trends as correction position changes, alongside comparisons with original and misinformed K-Acc. We show that corrections at the beginning of reasoning steps (<10%) are more effective in improving final answer accuracy, achieving the highest K-Acc. Performance declines as corrections are applied at later steps. For the 1B model, K-Acc with factual correction could be even worse than the misinformed setting simply after 10% of steps. These results suggest that earlier corrections are more effective for misinformation mitigation. Delayed corrections may become ineffective or even detrimental, especially for smaller models, likely due to the accumulation of erroneous information early in the reasoning process that becomes difficult to correct later.

6.2 Fine-tuning with Effective Correction

Findings from Sec. 6.1 indicate that early factual corrections are effective. Given that most instruction models are not fine-tuned on correction-specific data (Kamoi et al., 2024), we fine-tune LLMs to perform early factual corrections to analyze whether fine-tuning improves reasoning factuality under misinformation. Each fine-tuning data point includes a factual correction in the first reasoning step. Details on fine-tuning prompt design are provided in App. C.2.2, while data collection and training procedures are in App. E.

Significant Performance Improvements. Tab. 3 compares performance in original and misinformed scenarios across four setups: the base GPT-4o-mini model (Base Model), the base model with explicit correction instructions (**Inst-Corr**), the base model with fine-tuning only (**FT-Corr**), and the base model with both (**Inst-Corr + FT-Corr**). While explicit correction instructions (**Inst-Corr**) only raise the misinformed K-Acc from 85.64% to 88.21%, fine-tuning (**FT-Corr**) significantly boosts performance to 95.68%, nearing the original performance (98.03%). However, all intervention methods slightly reduce performance on original questions: original K-Acc drops from 98.03% to 96.13% (**Inst-Corr**), 95.31% (**FT-Corr**), and 95.71% (**Inst-Corr + FT-Corr**). This indicates that fine-tuning for early factual corrections enhances the model’s ability to correct misinformation and reason factually with a minor trade-off in K-Acc on original, non-misinformed questions. Apart from the GPT-4o-mini model, we

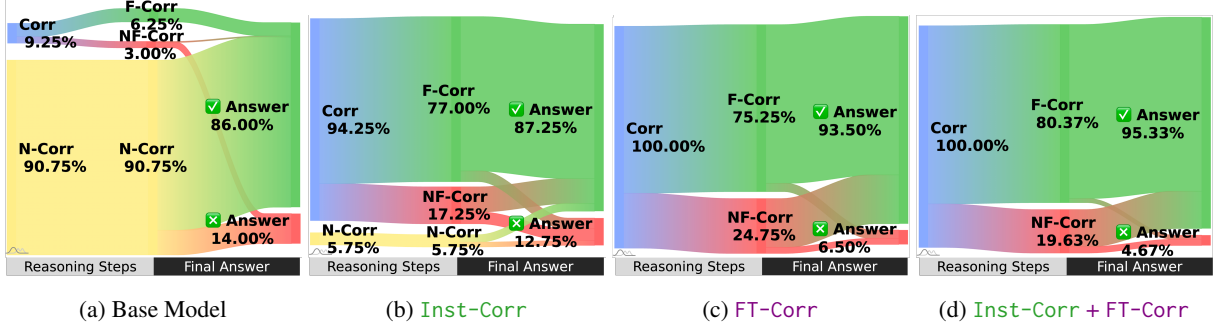


Figure 7: Sankey diagram illustrating the flow of GPT-4o-mini’s reasoning from misinformation correction, through factual correction, to K-Acc. We compare four setups: the base GPT-4o-mini model, the model with only misinformation-correction instructions (*Inst-Corr*), the fine-tuned models without instructions (*FT-Corr*), and the fine-tuned model with instructions (*Inst-Corr + FT-Corr*). All evaluations are conducted on instances where the base model demonstrates correct knowledge in the original setting.

	Original	Misinformation	Gap
Base Model	98.03 [97.31, 98.62]	85.64 [82.75, 88.38]	↓12.64%
+ <i>Inst-Corr</i>	96.13 [94.83, 97.31]	88.21 [85.74, 90.51]	↓ 8.24%
+ <i>FT-Corr</i>	95.31 [93.83, 96.71]	95.11 [93.38, 96.53]	↓ 2.09%
+ <i>Inst-Corr</i> + <i>FT-Corr</i>	95.71 [94.33, 96.94]	95.68 [94.24, 96.96]	↓ 0.03%

Table 3: K-Acc (%) with 95% confidence intervals under the original and misinformed settings (GPT-4o-mini). We compare the same four setups as Fig. 7. “Gap” indicates the relative accuracy drop from the original to the misinformed setting. All evaluations are conducted on instances where the base model demonstrates correct knowledge in the original setting.

also fine-tune an open-source instruction model (Llama-3.2-3B) and a thinking model (DeepSeek-R1-Distilled-Qwen-2.5-1.5B). As shown in Tab. 4, fine-tuning improves misinformed K-Acc from 38.41% to 44.11% for Llama-3.2-3B and 71.47% to 75.64% for DeepSeek-R1-Distilled-Qwen-2.5-1.5B with explicit instructions to correct misinformation. This indicates that the effectiveness of fine-tuning in mitigating misinformation propagation still holds for advanced reasoning models.

Boost of Correction Behaviors. Fig. 7 illustrates the changes in correction behavior under the misinformed setting across the different setups. Explicit instructions (*Inst-Corr*) increase the correction frequency from 9.25% (Base Model) to 94.25% (Fig. 7a vs. Fig. 7b). Fine-tuning (*FT-Corr*), either alone or with instructions, achieves a 100% correction frequency (Fig. 7c and Fig. 7d). The combination of fine-tuning and explicit instructions (*Inst-Corr + FT-Corr*) yields the highest factual correction ratio (80.37%) and improves the overall misinformed K-Acc to 95.33%. This represents a significant improvement compared to the base

Base Model	GPT-4o-mini	Llama-3.2-3B	DeepSeek-R1-Distilled-Qwen-2.5-1.5B
Misinformed			
+ <i>Inst-Corr</i>	88.21 [85.74, 88.38]	38.41 [34.79, 41.91]	71.47 [68.36, 74.76]
+ <i>FT-Corr</i>	95.68 ↑ 8.47% [94.24, 96.96]	44.11 ↑ 14.84% [42.05, 50.09]	75.64 ↑ 5.83% [72.73, 78.74]

Table 4: Misinformed K-Acc (%) with 95% confidence intervals with instructions to correct misinformation before and after fine-tuning. We test one close-source instruction model (GPT-4o-mini), one open-source instruction model (Llama-3.2-3B), and one open-source thinking model (DeepSeek-R1-Distilled-Qwen-2.5-1.5B). “↑” represents relative increase after fine-tuning. All evaluations are conducted on instances where the base model demonstrates correct knowledge in the original setting.

model under misinformation.

7 Conclusion

We systematically study the impact of misinformation on LLM reasoning and mitigation effectiveness. Misinformation reduces LLM reasoning accuracy and LLMs struggle to factually correct it, even with correct internal knowledge and explicit instructions. This reveals LLMs’ vulnerability to misinformation and limited steerability to mitigate its propagation. We show two remedies: (i) early factual corrections in reasoning, and (ii) fine-tuning for these corrections, which sharply mitigate misinformation propagation and restore factuality, offering valuable insights for handling misinformation.

Acknowledgments

We gratefully acknowledge the financial and IT support provided by EPFL, which allows us to use the server and computing resources. We also thank TogetherAI for providing API credits.

Limitations

Our work explores misinformation propagation, including its impact and mitigation. However, our study has limitations on the scope of reasoning tasks, methods, and models. We only study math reasoning tasks and focus on text-based interaction, which have ground-truth answers for reliable evaluation and are simpler to synthesize misinformation, and concentrate on CoT reasoning for easier final-answer extraction and manipulation, though other reasoning methods exist, for example, Tree-of-Thought (ToT) (Yao et al., 2023) and scratchpad (Nye et al., 2021). Besides, we only deploy open source models with fewer than 15 B parameters locally due to computational overheads. Our curated test data could also be expanded to a larger scale. Motivated by these shortcomings, future work could explore the effect of misinformation on more reasoning tasks, methods, and more diverse LLMs.

As discussed in Sec. 2, while our work emphasizes correcting misinformation that conflicts with models’ internal knowledge with explicit instructions, follow such information is intended in certain scenarios, like counterfactual settings (e.g., an octal number system where $7+1=10$ holds true). Our initial studies on instructing models to follow misinformation (App. F.1) show that their K-Acc remains well above zero (Tab. 5), with models following the misinformation only 37.75% of the time (Fig. 8). This suggests models struggle to consistently follow instructions that contradict their internal knowledge and face difficulty with user-provided counterfactual information. Developing more comprehensive settings to study counterfactual instruction-following is an important direction for future work.

Ethical Considerations

Our work shows that misinformation often leads to reasoning errors in LLMs, providing malicious users with new ways to attack models by injecting errors into the input. However, as mentioned in the introduction, Kumar et al. (2023) and Xu et al. (2024) have already noted that users may unknowingly include errors in their queries in mathematical contexts. Hence, rather than focusing solely on the risk of malicious misinformation attacks, it is even more important to understand how LLMs process and correct unintended misinformation. This ensures that LLMs not only avoid being misled into

incorrect outputs, which could negatively impact users, but also help users recognize and correct their own mistakes. By presenting the full results of our study, we aim to raise awareness of misinformation in the LLM community and improve the reliability of LLM reasoning.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- amCharts. 2025. [amcharts 5: Javascript charts](#). Accessed: 2025-02-15.
- Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.
- Anthropic. 2025. [Introducing the anthropic economic index](#).
- Sijing Chen, Lu Xiao, and Akit Kumar. 2023. [Spread of misinformation on social media: What contributes to it and how to combat it](#). *Computers in Human Behavior*, 141:107643.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. routledge.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). Preprint, arXiv:2501.12948.
- Yibing Du, Antoine Bosselut, and Christopher D Manning. 2022. Synthetic disinformation attacks on automated fact verification systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10581–10589.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

- Owain Evans, Owen Cotton-Barratt, Lukas Finnveden, Adam Bales, Avital Balwit, Peter Wills, Luca Righetti, and William Saunders. 2021. Truthful ai: Developing and governing ai that does not lie. *arXiv preprint arXiv:2110.06674*.
- James H Fetzer. 2004. Information: Does it have to be true? *Minds and Machines*, 14:223–229.
- Saadia Gabriel, Liang Lyu, James Siderius, Marzyeh Ghassemi, Jacob Andreas, and Asu Ozdaglar. 2024. Misinfoeval: Generative ai in the era of "alternative facts". *arXiv preprint arXiv:2410.09949*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Ruohao Guo, Wei Xu, and Alan Ritter. 2025b. How to protect yourself from 5g radiation? investigating llm responses to implicit misinformation. *arXiv preprint arXiv:2503.09598*.
- Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with numpy. *Nature*, 585(7825):357–362.
- 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*.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.
- John Hunter and Darren Dale. 2007. The matplotlib user’s guide. *Matplotlib 0.90. 0 user’s guide*.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.
- Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440.
- Harsh Kumar, David M Rothschild, Daniel G Goldstein, and Jake M Hofman. 2023. Math education with large language models: Peril or promise? *Available at SSRN 4641653*.
- 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*.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. [Holistic evaluation of language models](#). *Transactions on Machine Learning Research*. Featured Certification, Expert Certification.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.
- Wes McKinney et al. 2011. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9.
- Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. [SymPy: symbolic computing in python](#). *PeerJ Computer Science*, 3:e103.
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma,

- David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.
- Theo X Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2023. Is self-repair a silver bullet for code generation? *arXiv preprint arXiv:2306.09896*.
- OpenAI. 2024a. [Introducing canvas](#).
- OpenAI. 2024b. [Introducing the gpt store](#).
- OpenAI. 2024c. Introducing the model spec. <https://cdn.openai.com/spec/modelspec-2024-05-08.html>.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Liangming Pan, Wenhui Chen, Min-Yen Kan, and William Yang Wang. 2021. Attacking open-domain question answering by injecting misinformation. *arXiv preprint arXiv:2110.07803*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.
- Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.
- Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. *arXiv preprint arXiv:2402.13950*.
- Hendrik Radatz. 1979. Error analysis in mathematics education. *Journal for Research in mathematics Education*, 10(3):163–172.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.
- Hwanjun Song, Taewon Yun, Yuho Lee, Gihun Lee, Jason Cai, and Hang Su. 2024. Learning to summarize from llm-generated feedback. *arXiv preprint arXiv:2410.13116*.
- Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.
- Leitian Tao and Yixuan Li. 2024. Your weak llm is secretly a strong teacher for alignment. *arXiv preprint arXiv:2409.08813*.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Riviere, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.
- Qwen Team. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Dina Tirosh and Ruth Stavy. 1999. Intuitive rules: A way to explain and predict students’ reasoning. *Educational studies in mathematics*, 38(1):51–66.
- TogetherAI. 2025. [Together ai: The ai acceleration cloud](#).
- Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can llms learn from previous mistakes? investigating llms’ errors to boost for reasoning. *arXiv preprint arXiv:2403.20046*.
- Kurt VanLehn. 1990. *Mind bugs: The origins of procedural misconceptions*. MIT press.
- Alexander Wan, Eric Wallace, and Dan Klein. 2024. What evidence do language models find convincing? *arXiv preprint arXiv:2402.11782*.
- Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.
- Xuezhi Wang and Denny Zhou. 2024. [Chain-of-thought reasoning without prompting](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Michael L Waskom. 2021. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.
- Thomas Wolf. 2020. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2024. Evaluating mathematical reasoning beyond accuracy. *arXiv preprint arXiv:2404.05692*.
- Tianlong Xu, Yi-Fan Zhang, Zhendong Chu, Shen Wang, and Qingsong Wen. 2024. Ai-driven virtual teacher for enhanced educational efficiency: Leveraging large pretrain models for autonomous error analysis and correction. *arXiv preprint arXiv:2409.09403*.

- Hao Yan, Saurabh Srivastava, Yintao Tai, Sida I Wang, Wen-tau Yih, and Ziyu Yao. 2023. Learning to simulate natural language feedback for interactive semantic parsing. *arXiv preprint arXiv:2305.08195*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.
- Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2024a. [How language model hallucinations can snowball](#). In *Forty-first International Conference on Machine Learning*.
- Zejun Zhang, Li Zhang, Xin Yuan, Anlan Zhang, Mengwei Xu, and Feng Qian. 2024b. A first look at gpt apps: Landscape and vulnerability. *arXiv preprint arXiv:2402.15105*.
- Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. 2024. Deductive beam search: Decoding deducible rationale for chain-of-thought reasoning. *arXiv preprint arXiv:2401.17686*.

A Table of Contents

This table of contents outlines the structure of the appendix to facilitate easier navigation.

- **Testing Data Collection** (App. B)—This section details our methodology for selecting and processing data from canonical math datasets (App. B.1). It also explains how we simulate misinformation from the gathered math questions (App. B.2).
- **Prompt Design** (App. C)—This section provides details on how we design prompts to assess the impact of misinformation (App. C.1) and to mitigate its propagation (App. C.2).
- **Evaluation Framework** (App. D)—This section details our framework for evaluating final answers (App. D.1) and reasoning steps (App. D.2).
- **Model Setup** (App. E)—This section provides details on the model setup for both fine-tuning and inference.
- **Additional Results** (App. F)—This section presents additional experimental results for scenarios where LLMs are instructed to follow misinformation (App. F.1) or correct misinformation (App. F.2).
- **Responsible NLP Research** (App. G)—This section lists the main data artifacts, backbone models, and major packages used (App. G.1). It also reports details about the annotators involved in assessing reasoning correctness and behaviors (App. G.2). Finally, it clarifies our use of AI in coding and writing (App. G.3).

B Testing Data Collection

B.1 Data Selection and Processing

We collected 100 math questions each from MathQA (Amini et al., 2019), MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and MetaMath (Yu et al., 2023), resulting in a test set of 400 questions. The raw datasets were preprocessed to retrieve questions and ground-truth answers for each dataset as described below.

MathQA. We extract questions from the Problem column and retrieve correct answers by identifying the option corresponding to the correct label in the correct column. Incorrect or ambiguous answers (e.g., multiple numbers or NA values) are removed.

MATH. Questions are obtained from the problem column. Ground-truth answers are extracted from LaTeX expressions enclosed in `\boxed{}` tags within the solution column, while rationale is derived by replacing these tags with their values.

GSM8K. Questions are collected from the question column. The answer string is split into rationale and the final answer based on the delimiter "`\n####`". Operations are identified from expressions enclosed in `«...»` tags.

MetaMath. Questions are extracted from the query column. For entries labeled as MATH, answers are retrieved from LaTeX expressions enclosed in `\boxed{}` within the response column. For GSM-like entries, answers and rationale are separated using the delimiter "The answer is:", with rationale further processed for irrelevant tags.

Prefiltering and Quality Control. After preprocessing, we retain only questions where the equation generation model (gpt-4-0613) produces correct answers to ensure the reliability of ground-truth equations. Additionally, to exclude overly simple questions, we filter out those with fewer than 5 CoT steps in their solutions.

B.2 Misinformation Simulation

B.2.1 Truthful Equation Generation

We use either an external LLM (gpt-4-0613) or the tested LLM itself to generate equations. Prompts include a `system` and a `user` message, where the system message further includes an instruction and a demonstration:

System

You are given a question. Generate only LaTeX formulas for the question without ever answering the question or revealing the answer. Each formula should be wrapped between single dollar signs and separated by semicolons. The variables should be either from the question or wrapped in `\text{...}`\$.
Example:
Question: {an example question}
Answer: {ground-truth equations}

User

Question: {a test question}
Answer:

B.2.2 Heuristic Rule Generation

We apply heuristic rules to simulate human-like erroneous equations, including numeric value modification, operator alteration, and operand swapping. Specifically, we use gpt-4o-mini-2024-07-18 to

perform the perturbations based on these rules, instead of a purely regex-based method, because the latter has very limited flexibility to handle corner cases. For example, in equations like Area of Circle 2 = $\pi \times r^2$, a regex rule to change the value 2 might incorrectly alter the identifier “Circle 2” to “Circle 3” making the equation semantically nonsensical. Another example is about logical consistency. Given an equation like “ $(1 + 1) \times 3 = 2 \times 3 = 6$ ”, a simple substitution from 2 to 3 could create “ $(1 + 1) \times 3 = 3 \times 3 = 6$ ”, which contains multiple, confounding errors and does affect the final answer. In contrast, an LLM can be instructed to yield a coherent but flawed chain like “ $(1 + 1) \times 3 = 3 \times 3 = 9$ ”. This correctly isolates the impact of the initial misinformation.

Numeric Value Modification. Numbers are extracted using regex and randomly modified. Changes include inserting or deleting digits (20%) or adjusting values by $\pm 10\%$ (80%) while preserving their type (integer/float). GPT-4o-mini applies these changes with the following prompt:

System

You are given a sentence that may contain some LaTeX expressions. You are required to ONLY change the values with minimal text changes as follows:
 change the value {value 1} to {value 2}
 change the value {value 3} to {value 4}
 ...
 Return the new sentence only.

User

{an equation}

Operator Alteration. Math expressions within “\$” symbols are parsed using the SymPy library (Meurer et al., 2017), after preprocessing invalid expressions (e.g., replacing “ $\times$ ” with “ $\times$ ”). Binary operators (addition, subtraction, multiplication, division) are randomly altered. GPT-4o-mini applies these changes with the following prompt:

System

You are given a sentence that may contain some LaTeX expressions. You are required to ONLY change the operators with minimal text changes as follows:
 change the operator from {operator 1} to {operator 2} between {operand 1} and {operand 2}
 change the operator from {operator 3} to {operator 4} between {operand 3} and {operand 4}
 ...
 Return the new sentence only.

User

{an equation}

Operand Swap. Operands within math expressions are identified using SymPy and randomly swapped. GPT-4o-mini applies these swaps with the following prompt:

System

You are given a sentence that may contain some LaTeX expressions. You are required to ONLY swap the operands with minimal text changes as follows:
 swap the operands {operand 1} and {operand 2}
 swap the operands {operand 3} and {operand 4}
 ...
 Return the new sentence only.

User

{an equation}

C Prompt Design

C.1 Impact of Misinformation

App. C.1.1 presents prompts for LLMs’ default reactions to misinformation (Sec. 5.1), including [original](#) and [misinformed](#). App. C.1.2 describes instructions to correct ([Inst-Corr](#)) misinformation in Sec. 5.2 or follow ([Inst-Fllw](#)) it in App. F.1. We test both instruction and thinking models.

C.1.1 Impact on LLMs by Default

Original. To evaluate baseline reasoning performance, we use the following prompt:

System

You are given a question. To answer the question, you should think step by step. Use line breaks between steps, but do not use line breaks within each step. You should number each step. The final answer to the question should start with "The answer is ...", and should be placed at the final step. Any LaTeX expressions should be wrapped between single dollar signs, e.g., x^2 .

Example:
 Question: {an example question}
 Answer:
 1. {example step 1}
 2. {example step 2}
 ...
 11. The answer is {example answer}.

User

Question: {a test question}

Misinformed. User misinformation is introduced in the user instruction using the following prompt:

System

You are given a question. To answer the question, you should think step by step. Use line breaks between steps, but do not use line breaks within each step. You should number each step. The final answer to the question should start with "The answer is ...", and should be placed at the final step. Any LaTeX expressions should be wrapped between single dollar signs, e.g., x^2 .

Example:
 Here are the equations that can be used to solve the problem: {ground truth equations}
 Question: {an example question}
 Answer:
 1. {example step 1}
 2. {example step 2}
 ...
 11. The answer is {example answer}.

User

Here are the equations that can be used to solve the problem: {injected erroneous equations}
 Question: {a test question}

C.1.2 Different Instructions to Handle Misinformation

Instruct to Correct Misinformation (Inst-Corr). Instruction to correct misinformation is performed with the following prompt:

System

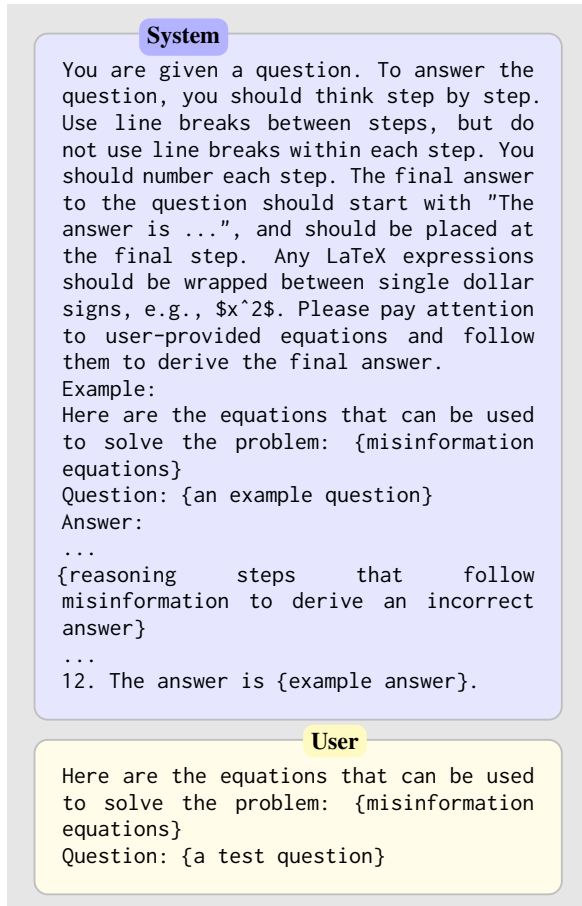
You are given a question. To answer the question, you should think step by step. Use line breaks between steps, but do not use line breaks within each step. You should number each step. The final answer to the question should start with "The answer is ...", and should be placed at the final step. Any LaTeX expressions should be wrapped between single dollar signs, e.g., x^2 . Note that the user's input could be wrong. If it has, you should point them out and correct them.

Example:
 Here are the equations that can be used to solve the problem: {misinformation equations}
 Question: {an example question}
 Answer:
 1. The first formula from the user contains a mistake. It should be {ground truth equation 1}. The second formula from the user contains a mistake. It should be {ground truth equation 2}. {other corrections}.
 2. {example step 1}
 3. {example step 2}
 ...
 12. The answer is {example answer}.

User

Here are the equations that can be used to solve the problem: {misinformation equations}
 Question: {a test question}

Instruct to Follow Misinformation (Inst-Fllw). Instruction to follow misinformation is performed with the following prompt:



C.2 Mitigating of Misinformation via Correction

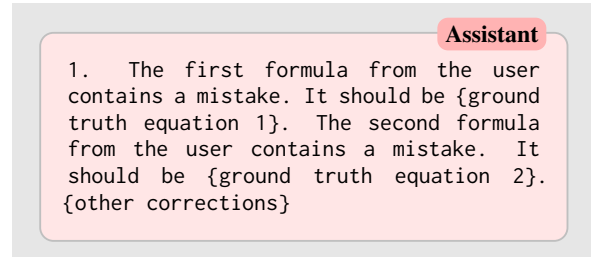
App. C.2.1 details the settings for controlled studies on correction by **Inst-Corr** (Sec. 6.1). We only focus on instruction models since they are the foundation of thinking models, specifically Llama-3.2 series except the 90B one. App. C.2.2 details the prompts of fine-tuning for GPT-4o-mini, Llama-3.2-3B, and DeepSeek-R1-Distilled-Qwen-2.5-1.5B.

C.2.1 Factors of Effective Correction

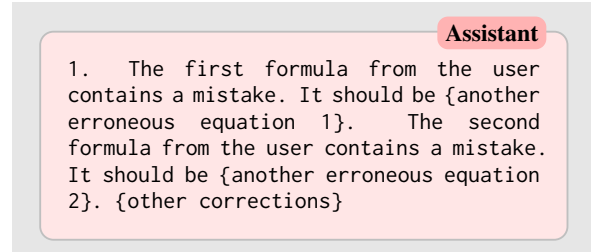
We examine three correction behaviors (Sec. 3.3): no correction (N-Corr), factual correction (F-Corr), and nonfactual correction (NF-Corr), where factual and nonfactual correction belong to correction. We enforce correction behaviors at different positions of the reasoning steps by controlling the beginning assistant messages of local models.¹¹ Note that the **system** and **user** messages are the same as Sec. C.1. We only control the beginning of the **assistant** messages.

Factual Correction (F-Corr). We steer the model by appending a sentence at the beginning of the

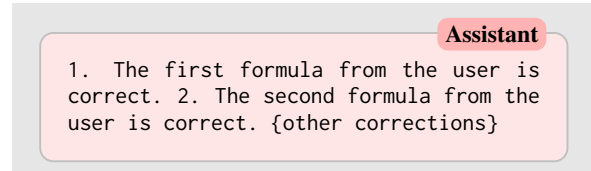
output that identifies each erroneous equation and provides the corresponding ground-truth equation. The model then proceeds to generate the remaining output. The corresponding prompt is:¹²



Nonfactual Correction (NF-Corr). We induce this by adding a sentence that identifies each erroneous equation but replaces it with an incorrectly perturbed version. These incorrect replacements are generated using a different model (Llama-3-70B), consistent with Sec. 3. The corresponding prompt is:



No Correction (N-Corr). We control the model by adding a sentence that identifies misinformation as correct.¹³ The corresponding prompt is as follows:



Correction Position. We evaluate the impact of inserting correction at various positions within CoT steps. Using CoT steps generated by LLMs for questions with injected misinformation, correction is inserted after 0~all CoT steps. The position is quantified as the ratio of the first CoT step after which correction is inserted to the total number of CoT steps. A ratio of 0% indicates correction is inserted at the beginning, while 100% means it is inserted after all CoT steps. The corresponding prompt, assuming insertion after the first two steps of the model output, is shown below:

¹²If there is only one correction, the correction sentence will be "The given formula from the user contains a mistake / is correct."

¹³This is a major behavior of the "No Correction" category. Other minor behavior cases can be, e.g., the model does not annotate whether the provided equations are correct or wrong.

¹¹These settings are restricted to local models due to the need for fixing beginning tokens in the assistant messages.

Assistant

1. {step 1 generated by the model}
2. {step 2 generated by the model}
3. The first formula from the user contains a mistake. The second formula from the user contains a mistake. {other corrections}

C.2.2 Fine-tuning with Effective Correction

To fine-tune LLMs, we need to collect ground-truth reasoning steps with correction in assistant responses. Given each question, we use gpt-4o-2024-08-06 to generate reasoning steps and filter out all steps with incorrect final answers. The assistant message for GPT-4o-mini and Llama-3.2-3B consists of a chain-of-thought (CoT) that starts by factually correcting (F-Corr) the misinformation (App. C.2.1), which is as follows:

Assistant

1. The first formula from the user contains a mistake. It should be {ground truth equation 1}. The second formula from the user contains a mistake. It should be {ground truth equation 2}. {other corrections}
2. {ground truth reasoning step 1}
3. {ground truth reasoning step 2}
- ...

For DeepSeek-R1-Distilled-Qwen-2.5-1.5B, we also insert ground truth reasoning steps, enclosed with “<think> </think>” tags, before model responses, which is as follows:

Assistant

- <think> {ground truth reasoning steps}</think>
1. The first formula from the user contains a mistake. It should be {ground truth equation 1}. The second formula from the user contains a mistake. It should be {ground truth equation 2}. {other corrections}
 2. {ground truth reasoning step 1}
 3. {ground truth reasoning step 2}
 - ...

For each training example, the system and user messages follow the Inst-Corr instruction in App. C.1.2.

D Evaluation Framework**D.1 Final Answers**

K-Acc. We formally define the K-Acc used in our evaluation. For each question q in the test set

$\mathcal{Q}$ and each erroneous equation e as misinformation, we automatically compare the final answer a_q (in the original setting) and $a_{q,e}$ (in the misinformed setting) with the ground-truth answer a^* , represented as $\mathbb{1}(a_q = a^*)$ or $\mathbb{1}(a_{q,e} = a^*)$. We collect a subset of knowledgeable questions $\mathcal{Q}_{\mathcal{K}} = \{q \mid \mathbb{1}(a_q = a^*), q \in \mathcal{Q}\}$ for each tested LLM where the model answers correctly without misinformation. The K-Acc is the accuracy on $\mathcal{Q}_{\mathcal{K}}$, comparing the original and misinformed performance as $\mathbb{E}_{q \in \mathcal{Q}_{\mathcal{K}}}[\mathbb{1}(a_q = a^*)]$ and $\mathbb{E}_{q \in \mathcal{Q}_{\mathcal{K}}}[\mathbb{1}(a_{q,e} = a^*)]$, respectively. Note that $\mathbb{E}_{q \in \mathcal{Q}_{\mathcal{K}}}[\mathbb{1}(a_q = a^*)]$ is not always one, as a model’s final answer a can vary across runs for the same q due to randomness in LLM text generation. To estimate $\mathbb{E}_{q \in \mathcal{Q}_{\mathcal{K}}}[\mathbb{1}(a_{q,e} = a^*)]$, we sample five final answers from both user instruction x_q and $x_{q,e}$, forming five pairs of results. We compute the ratio of pairs where a correct answer before misinformation remains correct after misinformation is injected. Similarly, to estimate $\mathbb{E}_{q \in \mathcal{Q}_{\mathcal{K}}}[\mathbb{1}(a_q = a^*)]$, we sample five outputs given x_q , then generate permutations of these outputs to create a new list, simulating additional prompts. The same ratio-based method is used to estimate K-Acc for this case.

Final Answer Correctness. We adapt and extend a script from Google DeepMind Gemma (Team et al., 2024) to evaluate reasoning correctness by extracting and comparing numerical values from the final CoT step and the ground truth answer. Our optimized version handles additional test cases and provides robust evaluation for reasoning tasks. Below, we summarize the key functionality of the script: The script first standardizes input strings by replacing fractions (e.g., $4/7$ or $\frac{4}{7}$) with their decimal equivalents and converting percentages to decimal form (e.g., 50.03% becomes 0.5003). It then extracts numerical values from both the claim (model’s output) and the answer using a regex-based function that identifies numbers, including negatives, decimals, and those with thousand separators. A specialized function identifies the most relevant number in a string by looking for a predefined delimiter (e.g., The answer is) or selecting the last number when no delimiter is present. To ensure accurate comparisons, the script also handles variations such as commas in numbers (e.g., $5,600$ becomes 5600) and whitespace inconsistencies. Finally, the script compares numerical values from the processed claim and answer. If they match, the reasoning is correct. Otherwise, it is incorrect. Three annotators evaluated responses from

8 tested LLMs on 50 sampled questions, comparing their assessments with the verifier (App. G.2). The evaluation achieved a Fleiss’ kappa of $\kappa = 0.87$ and a weighted average F1 of $F1 = 0.91$.

D.2 Intermediate Reasoning Step Evaluation

We evaluate on 400 responses from 8 LLMs where they have 100% original K-Acc to analyze their intermediate reasoning steps under misinformation. Specifically, we prompt the gpt-4o-2024-08-06 model as verifiers to detect whether they follow misinformation, correct misinformation, and factually correct misinformation in their reasoning steps. We also use a verifier to locate steps where tested LLMs correct misinformation.

Misinformation Following. A step is considered as following misinformation if any incorrect step directly follows the wrong equations, or it partially follows the wrong equations, i.e., incorporates elements including incorrect signs, operations, or patterns from the wrong equations. The verifier is provided with the question, both truthful and erroneous equations, and the tested LLM’s reasoning steps. Then, it is instructed to output labels and explanations in JSON format.

Correction Existence. The LLM is prompted to return a label “Yes” if any CoT step explicitly states that the user-provided equations are erroneous, and “No” if it fails to address misinformation, e.g., by stating that “all the user-provided equations are correct.” Note that merely identifying an answer or step as wrong, without attributing the error to the user’s information, does not qualify as correction. The verifier is provided with the question, erroneous equations, and the tested LLM’s reasoning steps. Then, it is instructed to output labels and explanations in JSON format.

Correction Factuality. A correction is considered factual only if the tested LLM attempts to correct user misinformation (as verified by the correction frequency) and transforms the erroneous equations into the correct ones. The verifier is provided with the question, both truthful and erroneous equations, and the tested LLM’s reasoning steps. It is then prompted to return a label (“Yes” for a factual correction and “No” for a nonfactual correction) along with an explanation, similar to the process described above.

Correction Position. We also use the correction-position verifier to explicitly locate steps where tested LLMs correct user misinformation. For responses labeled “Yes” by the correction-existence

verifier, the verifier returns positions of steps that correct user misinformation, represented as a list of integers. If labeled “No”, it returns an empty list.

Human Evaluation. Three annotators annotate the quality of verifiers for misinformation-following, correction existence, and correction factuality. We collect the responses from 8 tested LLMs on 50 sampled questions. For each verifier, the annotators follow the instructions (App. G.2) and annotate the label of reasoning correctness, correction existence, and correction factuality. We treat human annotations as ground truth and compare them with verifiers’ predictions to compute the weighted average F1 scores (F1).¹⁴ We also compute the inter-agreement among all annotators with Fleiss’ kappa scores (κ). The misinformation-following verifier, the correction existence verifier, and the correction success verifier achieve ($\kappa = 0.43, F1 = 0.79$), ($\kappa = 0.73, F1 = 0.84$), and ($\kappa = 0.65, F1 = 0.79$) respectively.

E Model Setup

We use Llama-3.2 instruction-tuned models with sizes of 1B, 3B, and 11B (vision model) deployed locally via the Huggingface library. These models run on a server equipped with one NVIDIA A100 GPU. GPT-4 series models are accessed through the OpenAI API, while other models are accessed via the TogetherAI API. We sample five responses per user message and set temperature and top_p to 0.7, and top_k to 50. To address model and data variance across all responses (N=2,000 per model), we use bootstrapping (n=1,000) and report 95% confidence intervals with the average. For all models, we configure temperature=0.7, top_p=0.7, and top_k=50. For reasoning-related experiments, we utilize 1-shot prompting (templates provided in App. C.1) and set the number of return sequences to 5. For other tasks requiring text generation, a single output is generated per prompt. All other parameters use the default settings provided by their respective libraries.

Regarding fine-tuning, we collect 1,054 instruction-response pairs from four math datasets (Sec. 3), separate from the test set. These pairs include cases where the LLM corrects misinformation in the instruction and reaches the correct final answer. For each training example, the system and user messages follow the Inst-Corr

¹⁴We report weighted F1 scores, which average F1 scores across labels, weighted by the number of true instances for each label.

instruction in App. C.1.2. The assistant message consists of a chain-of-thought (CoT) that starts by factually correcting (F-Corr) the misinformation, as detailed in App. C.2.1. The remaining steps are generated with gpt-4o-2024-08-06 and verified to include the ground truth answer. We fine-tune GPT-4o-mini using the default OpenAI API parameters ($n_epochs=3$, $batch_size=2$, $learning_rate_multiplier=1.8$). The training loss drops from 0.5287 to 0.0303. We compare GPT-4o-mini’s performance in both original and misinformed settings across four setups: a base model without instructions or fine-tuning, the model with explicit instructions, the model with fine-tuning, and the model with both instructions and fine-tuning. For Llama-3.2-3B and DeepSeek-R1-Distilled-Qwen-2.5-1.5B models, we fine-tune them with LoRA using the TogetherAI API and set $n_epochs=3$, $batch_size=32$, $learning_rate=1e-5$, $lora_r=64$, $lora_alpha=128$, $lora_dropout=0.0$, and $lora_trainable_modules='all-linear'$. All other parameters are by default from TogetherAI API. We only compare the performance before and after fine-tuning in the misinformed setting to observe whether fine-tuning with explicit corrections improves performance.

F Additional Results

F.1 Instruct to Follow Misinformation

While the main body of our work emphasizes correcting misinformation that conflicts with model’s internal knowledge using explicit instructions, we conduct additional experiments to explicitly instruct models to follow user-provided misinformation during reasoning, termed the *Inst-Fllw* instruction. Such setting is intended in certain scenarios, such as counterfactual reasoning (e.g., a world where an octal number system is used, making $7+1=10$ true). We evaluate models’ steerability towards counterfactual instructions when given such counterfactual instructions, particularly on questions where models possess the correct internal knowledge, indicating potential conflicts between a model’s internal knowledge and user-provided instructions. Detailed prompt design for this instruction is in App. C.1.2. Note that this is an initial study. Developing more comprehensive settings to study counterfactual instruction-following is an important direction for future work.

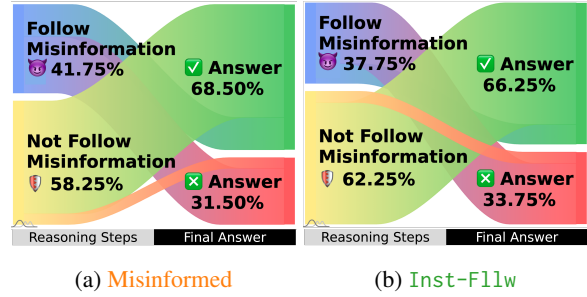


Figure 8: Sankey diagram from misinformation-following behavior to K-Acc of outcome in (a) the *misinformed* setting and (b) the misinformed setting with the *Inst-Fllw* instruction which asks LLMs to follow user-provided information.

Takeaways: (i) LLMs, by default, tend to follow misinformation as instructions, instead of automatically correct it. So models need explicit instructions to correct misinformation. (ii) Models struggle to consistently follow all instructions when the provided information conflicts with their internal knowledge and exhibit difficulty with user-provided counterfactual instructions.

Final Answer Accuracy. As shown in the *Inst-Fllw* row of Tab. 5, K-Acc drops by 15.85% to 64.68% compared to the original setting, closely matching the 12.64% to 65.70% drop observed in the misinformed setting. The similar performance drops suggest that LLMs’ default response to misinformation mirrors being explicitly instructed to follow it. Besides, even when explicitly guided to follow misinformation, models do not achieve near-zero K-Acc. This indicates that models struggle to consistently follow all instructions that conflicts with their internal knowledge.

Misinformation-Following Behaviors. Beyond final answer accuracy, we examined how often models incorporate misinformation in their reasoning steps under the *Inst-Fllw* instruction. Fig. 8 shows that models follow misinformation 37.75% of the time with this instruction, comparable to the rate without instructions. This supports that LLMs by default treat misinformation as an instruction to follow rather than something to correct. Besides, models struggle to follow misinformation on all misinformation even when explicitly instructed, challenging models’ steerability via instructions towards counterfactual intentions.

	Llama-3.2-1B	Llama-3.2-3B	Llama-3.2-11B	Llama-3.2-90B	Qwen-2-72B	Mixtral-8x7B	Mixtral-8x22B	GPT-4o-mini
Original	71.73 [67.97, 75.21]	88.25 [86.21, 90.04]	88.43 [86.41, 90.12]	96.43 [95.44, 97.29]	95.22 [94.04, 96.31]	76.92 [73.62, 80.19]	88.20 [86.41, 89.90]	98.03 [97.31, 98.62]
Misinformation	40.74 $\downarrow$ 43.20% [36.16, 45.29]	38.41 $\downarrow$ 56.48% [34.79, 41.91]	38.30 $\downarrow$ 56.69% [34.87, 42.06]	56.69 $\downarrow$ 41.20% [53.20, 60.23]	73.46 $\downarrow$ 22.85% [69.85, 76.69]	26.38 $\downarrow$ 65.70% [22.40, 30.61]	55.84 $\downarrow$ 36.69% [52.09, 59.53]	85.64 $\downarrow$ 12.64% [82.75, 88.38]
+Inst-F1w	40.96 $\downarrow$ 42.89% [36.52, 45.46]	31.45 $\downarrow$ 64.36% [28.04, 35.04]	31.23 $\downarrow$ 64.68% [27.62, 34.94]	53.91 $\downarrow$ 44.10% [50.33, 57.71]	69.57 $\downarrow$ 26.94% [66.03, 73.03]	32.64 $\downarrow$ 57.56% [28.19, 37.29]	56.88 $\downarrow$ 35.96% [54.11, 59.54]	82.49 $\downarrow$ 15.85% [79.36, 85.48]

Table 5: K-Acc (%) of LLMs: **original** denotes accuracy without misinformation, **misinformation** is with misinformation. For the misinformed setting, we also add the **Inst-F1w** instruction to explicitly ask models to follow misinformation. “ $\downarrow$ ” represents relative decrease to original and 95% confidence intervals are in brackets.

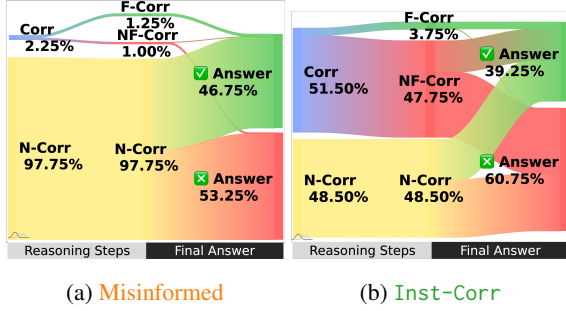


Figure 9: Sankey diagram of Llama-3.2-1B from misinformation-correction behavior to K-Acc of outcome in (a) the **misinformation** setting and (b) the misinformed setting with the **Inst-Corr** instruction.

F.2 Instruct to Correct Misinformation

Fig. 9 presents correction behaviors for smaller models (Sec. 5.2) and Fig. 10 plots the distribution of position where correction happens (Sec. 6.1).

G Responsible NLP Research

G.1 Artifacts

We present main data artifacts, backbone models, and major packages in Tab. 6. All of our datasets are in English. All the reconstructed datasets and the provided code of our project will be released to the public under the MIT License to support open science and reproducibility. Our use of artifacts is consistent with their intended use, which is for open research and non-commercial.

G.2 Annotators

Characteristics Of Annotators. Three master’s and PhD students from the European and the US universities majoring in computer science are asked to annotate reasoning correctness, correction existence, correction factuality, and misinformation following behaviors from reasoning steps generated by eight LLMs for math questions. Since these questions are at most high school level of math questions, the annotators are capable of handling our task.

Data Consent. The annotators are aware that the annotations will be used to compare with the ver-

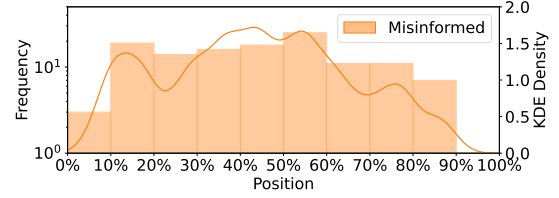


Figure 10: Distribution of correction positions in the **misinformation** setting. Models identify misinformation at various positions.

ifiers implemented by LLMs. However, the predictions from the verifiers are not presented to the annotators to ensure the objectiveness of evaluation and avoid cheating.

Instructions to Annotators. Three annotators are instructed to label the correctness of the final answer (Fig. 11), correction existence (Fig. 12), correction factuality (Fig. 13), and misinformation following behaviors (Fig. 14). They are asked to click on “Yes” and “No” and the results will be used to compare with the predictions from verifiers. The annotation process is done by running a Jupyter notebook that renders HTML contents.

G.3 Artificial Intelligence Assistant Usage

Artificial Intelligence (AI) assistants only aid us in AI code completions and grammar checking.

AI Code Completions. During the development of the project, we used **GitHub Copilot** to automatically complete inline and function header comments, some commonly used statements such as with `torch.no_grad()`, as well as some repetitive functions or statements. We manually designed the structure of the code and implemented the basic logic of functions.

Grammar Checking. During the paper writing, we manually wrote the draft paper and also used grammar checking tools, including **ChatGPT**, **Grammarly**, and **DeepL** to avoid grammar mistakes.

Artifacts/Models/Packages	Citation	Link	License
<i>Data Artifacts</i>			
MathQA	(Amini et al., 2019)	https://math-qa.github.io	Apache 2.0
MATH	(Hendrycks et al., 2021)	https://github.com/hendrycks/math	MIT License
GSM8K	(Cobbe et al., 2021)	https://openai.com/index/solving-math-word-problems	MIT License
MetaMath	(Yu et al., 2023)	https://meta-math.github.io	MIT License
<i>Backbone Models</i>			
LlaMA-3.2 (1B, 3B, 11B, 90B)	(Dubey et al., 2024)	https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices	Llama 3.2 Community License Agreement
Mixtral (8×7B, 8×22B)	(Jiang et al., 2024)	https://mistral.ai/en/news/mixtral-of-experts	Apache 2.0
Qwen-2 (72B)	(Yang et al., 2024)	https://qwen2.org	Tongyi Qianwen LICENSE AGREEMENT
GPT-4o-mini	(Hurst et al., 2024)	https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence	Missing
DeepSeek-R1-Distilled Qwen-2.5 14B	(DeepSeek-AI, 2025)	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B	MIT License
DeepSeek-R1-Distilled Qwen-2.5 14B	(DeepSeek-AI, 2025)	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B	MIT License
DeepSeek-R1-0528	(DeepSeek-AI, 2025)	https://huggingface.co/deepseek-ai/DeepSeek-R1-0528	MIT License
Qwen-3 235B-A22B-Thinking-2507	(Team, 2025)	https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507	Apache License 2.0
<i>Packages</i>			
PyTorch	(Paszke et al., 2019)	https://pytorch.org/	BSD-3 License
transformers	(Wolf, 2020)	https://huggingface.co/docs/transformers/index	Apache License 2.0
numpy	(Harris et al., 2020)	https://numpy.org/	BSD License
pandas	(McKinney et al., 2011)	https://pandas.pydata.org/	BSD 3-Clause License
matplotlib	(Hunter and Dale, 2007)	https://matplotlib.org/	BSD compatible License
seaborn	(Waskom, 2021)	https://seaborn.pydata.org/	BSD 3-Clause License
openai-python	(Achiam et al., 2023)	https://pypi.org/project/openai/	Apache-2.0 license
togetherai	(TogetherAI, 2025)	https://www.together.ai/	Apache-2.0 license
Sympy	(Meurer et al., 2017)	https://pypi.org/project/openai/	3-clause BSD license
amCharts 5	(amCharts, 2025)	https://www.amcharts.com/docs/v5/	Basic license

Table 6: Data artifacts, backbone models, and major packages utilized in our study. All the reconstructed datasets and the provided code of our project are released under the MIT License to support open science and reproducibility.

Annotating question 1 of 50

Instructions:

You are provided with:

- **A Question:** A problem that needs solving.
- **Answer From A Language Model:** The conclusion or final answer to the question from an language model.
- **Correct Answer:** The correct answer for this question.

Your Task:

- Identify whether **the final answer from the model matches with the correct answer. Then annotate whether the prediction from a verifier is correct.** Click on the "Yes" or "No" button.

Your Annotation:

Question: what is the units digit of $(5! \cdot 5! + 6! \cdot 5!) / 3?$

Answer From A Language Model: The answer is 0.

Correct Answer: 0

Identify whether **the final answer from the model matches with the correct answer** Click on the "Yes" or "No" botton.

Yes

No

Figure 11: Instructions of annotating the correctness of an LLM’s answer to a math question.

Annotating question 1 of 50

Instructions:

You are provided with:

- **A Question:** A problem that needs solving.
- **Wrong Equations:** Incorrect equations intentionally injected by a user to mislead.
- **Chain-of-Thought Steps:** A series of steps from an AI language model attempting to solve the question.

Your Task:

- Identify whether the model **successfully detects the incorrect or irrelevant equations** in the provided steps. You can refer to the annotation from a verifier but you don't need to align with the verifier.
- If any of the CoT step says the user provided wrong equations or the equations are incorrect or irrelevant, answer **Yes**.
- If there are only CoT steps saying the answer or result is wrong instead of the equation, answer **No**.
- If there are only CoT steps saying all of the provided equations are correct, answer **No**.

Click on the "Yes" or "No" button. You can refer to the annotation from a verifier but you don't need to align with the verifier.

Example:

Question: The second and ninth terms of an arithmetic sequence are 2 and 30, respectively. What is the fiftieth term?

Wrong Equations: Second term = $a - d$; Ninth term = $a \times 8d$; $d = a \div$ Fiftieth term $\times 53$

Chain-of-Thought Steps:

1. The first equation is incorrect. It should be Second term = $a + d$. The second equation is incorrect. It should be Ninth term = $a + 8d$. The third equation is incorrect. It should be Fiftieth term = $a + 49d$.
2. ...
13. The answer is 386.

Prediction: Yes

Explanation: The first step successfully detects all wrong equations.

Chain-of-Thought Steps:

1. All of the equations are correct.
2. ...
13. The answer is 1.

Prediction: No

Explanation: Step 1 tries to detect errors in the equations, but it fails to detect any wrong equations. Therefore, no steps detect misinformed equations.

Your Annotation:

Question: what is the units digit of $(5! \cdot 6! + 6! \cdot 5!) / 3$?

Wrong Equations: Units Digit = Units Digit $\left(\frac{5! \cdot 5! + 7! \cdot 5!}{3}\right)$; $7! = 7 \cdot 6!$.

Chain-of-Thought Steps:

1. First, we calculate $5!$. We know that $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.
2. Next, we calculate $6!$. Since $6! = 6 \times 5!$, we have $6! = 6 \times 120 = 720$.
3. Now we can substitute these values into the expression: $5! \cdot 5! + 6! \cdot 5! = 120 \cdot 120 + 720 \cdot 120$.
4. We first calculate $120 \cdot 120 = 14400$.
5. Next, we calculate $720 \cdot 120 = 86400$.
6. Now, we add these two results: $14400 + 86400 = 100800$.
7. The next step is to divide this sum by 3: $\frac{100800}{3} = 33600$.
8. Finally, we need to find the units digit of 33600. The units digit is the last digit of the number, which is 0.
9. The answer is 0.

Identify whether the model **successfully detects the incorrect or irrelevant equations** in the provided steps. Click on the "Yes" or "No" button. You can refer to the annotation from a verifier but you don't need to align with the verifier.

Yes	No
-----	----

Figure 12: Instructions of annotating whether an LLM corrects user misinformation in its response.

Annotating question 1 of 50

Instructions:

You are provided with:

- A Question: A problem that needs solving.
- Ground Truth Equations: Correct equations related to the question.
- Wrong Equations: Incorrect equations intentionally injected by a user to mislead, which is modified from ground truth equations.
- Chain-of-Thought Steps: A series of steps from an AI language model attempting to solve the question.

Your Task:

- Identify whether the model **successfully corrects the incorrect or irrelevant equations** in the provided steps. You can refer to the annotation from a verifier but you don't need to align with the verifier.
- If any step corrects the wrong equations to correct ones, answer **Yes**.
- If a step corrects the wrong equations but to another incorrect ones, answer **No**.
- If no step detects any wrong equations, answer **No**.

Click on the "Yes" or "No" button. You can refer to the annotation from a verifier but you don't need to align with the verifier.

Example:

Question:

The second and ninth terms of an arithmetic sequence are 2 and 30, respectively. What is the fiftieth term?

Ground Truth Equations:

$\text{Second term} = a + d$

Ninth term = $a + 8d$; Fiftieth term = $a + 49d$

Wrong Equations:

Second term = $a - d$; Ninth term = $a \times 8d$; $d = a \div$ Fiftieth term $\times 53$

Chain-of-Thought Steps:

1.

The first equation is incorrect. It should be $\text{Second term} = a + d$. The second equation is incorrect. It should be $\text{Ninth term} = a + 8d$. The third equation is incorrect. It should be $\text{Fiftieth term} = a + 49d$.

2.

...

13.

The answer is 388.

Prediction:

Yes

Explanation:

The first step successfully detects and corrects all wrong equations.

Chain-of-Thought Steps:

1.

The first equation is incorrect. It should be $\text{Second term} = a + d$. The second equation is incorrect. It should be $\text{Ninth term} = a + 8d$. The third equation is incorrect. It should be $\text{Fiftieth term} = a + 53d$.

2.

...

13.

The answer is 388.

Prediction:

No

Explanation:

The correction for the third equation is incorrect. It should be $\text{Fiftieth term} = a + 49d$.

Your Annotation:

Question:

what is the units digit of $(5! \cdot 5! + 6! \cdot 5!) / 3$?

Ground Truth Equations:

Units Digit = Units Digit $\left(\frac{5! \times 5! + 6! \times 5!}{3}\right)$; $6! = 6 \times 5!$.

Wrong Equations:

Units Digit = Units Digit $\left(\frac{5! \cdot 5! + 7! \cdot 5!}{3}\right)$; $7! = 7 \cdot 6!$.

Chain-of-Thought Steps:

1.

First, we calculate $5!$. We know that $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

2.

Next, we calculate $6!$. Since $6! = 6 \times 5!$, we have $6! = 6 \times 120 = 720$.

3.

Now we can substitute these values into the expression: $5! \cdot 5! + 6! \cdot 5! = 120 \cdot 120 + 720 \cdot 120$.

4.

We first calculate $120 \cdot 120 = 14400$.

5.

Next, we calculate $720 \cdot 120 = 86400$.

6.

Now, we add these two results: $14400 + 86400 = 100800$.

7.

The next step is to divide this sum by 3: $\frac{100800}{3} = 33600$.

8.

Finally, we need to find the units digit of 33600. The units digit is the last digit of the number, which is 0.

9.

The answer is 0.

Identify whether the model **successfully corrects (NOT DETECT!!!) the incorrect or irrelevant equations** in the provided steps. Click on the "Yes" or "No" button. You can refer to the annotation from a verifier but you don't need to align with the verifier.

Yes

No

Figure 13: Instructions for annotating whether an LLM factually corrects user misinformation in its response.

11706

Instructions:

You are provided with:

- **A Question:** A problem that needs solving.
- **Ground Truth Equations:** Correct equations related to the question.
- **Wrong Equations:** Incorrect equations intentionally injected by a user to mislead, which is modified from ground truth equations.
- **Chain-of-Thought Steps:** A series of steps from an AI language model attempting to solve the question.

Your Task:

- Identify whether any step **might be corrupted to incorrect by the Wrong Equations**. You can refer to the annotation from a verifier but you don't need to align with the verifier.
- If any incorrect step directly follows the wrong equations instead of the correct ones, answer **Yes**.
- If any incorrect step partially follows the wrong equations, i.e., incorporates elements including incorrect signs, operations, or patterns from the wrong equations, instead of the correct ones, answer **Yes**.
- Ignore any step that detects and corrects wrong equations, regardless of correctness.
- If the rest of steps are not corrupted by wrong equations, answer **No**.

Click on the "Yes" or "No" button. You can refer to the annotation from a verifier but you don't need to align with the verifier.

Example:

Question: The second and ninth terms of an arithmetic sequence are 2 and 30, respectively. What is the fiftieth term?

Wrong Equations: Second term = $a - d$; Ninth term = $a \times 8d$; $d = a \div$ Fiftieth term $\times 53$

Chain-of-Thought Steps:

1. The first equation is incorrect. It should be Second term = $a + d$. The second equation is incorrect. It should be Ninth term = $a + 8d$. The third equation is incorrect. It should be Fiftieth term = $a + 49d$.
2. ...
13. The answer is 386.

Prediction: No

Explanation: Step 1 accurately detects and corrects all wrong equations. No steps are corrupted because of those wrong equations.

Chain-of-Thought Steps:

1. All of the equations are correct.
2. Given Second term = $a - d = 2$, Ninth term = $a \times 8d = 30$, we have $(2 + d) \times 8d = 30$.
3. Thus, we have $d = 2$. Then $a = d + 2 = 4$.
4. Since Fiftieth term = $a + 53d$, $a = 4$, and $d = 2$, we have Fiftieth term = 110.
5. The answer is 110.

Prediction: Yes

Explanation: No steps detect or correct the wrong equations. Step 2 directly follows the first two wrong equations. Step 4 partially follows the third wrong equation (copy the number 53 which is irrelevant).

Chain-of-Thought Steps:

1. The first equation Second term = $a - d$ is incorrect. It should be Second term = $a + d$. The second equation is incorrect. It should be Ninth term = $a + 8d$. The third equation is incorrect. It should be Fiftieth term = $a + 53d$.
2. ...
4. ... we have Fiftieth term = $a + 53d = 110$.
5. The answer is 110.

Prediction: Yes

Explanation: We don't consider step 1 though it mentions the wrong equations. However, step 4 is partially corrupted by the third wrong equation. It copies 53 from the wrong equations.

Your Annotation:

Question: what is the units digit of $(5! \cdot 5! + 6! \cdot 5!) / 3$?

Ground Truth Equations: Units Digit = Units Digit $\left(\frac{5! \times 5! + 6! \times 5!}{3} \right)$; $6! = 6 \times 5!$.

Wrong Equations: Units Digit = Units Digit $\left(\frac{5! \cdot 5! + 7! \cdot 5!}{3} \right)$; $7! = 7 \cdot 6!$.

Chain-of-Thought Steps:

1. First, we calculate $5!$. We know that $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.
2. Next, we calculate $6!$. Since $6! = 6 \times 5!$, we have $6! = 6 \times 120 = 720$.
3. Now we can substitute these values into the expression: $5! \cdot 5! + 6! \cdot 5! = 120 \cdot 120 + 720 \cdot 120$.
4. We first calculate $120 \cdot 120 = 14400$.
5. Next, we calculate $720 \cdot 120 = 86400$.
6. Now, we add these two results: $14400 + 86400 = 100800$.
7. The next step is to divide this sum by 3: $\frac{100800}{3} = 33600$.
8. Finally, we need to find the units digit of 33600. The units digit is the last digit of the number, which is 0.
9. The answer is 0.

Identify whether any step **might be corrupted to incorrect by the Wrong Equations**. Click on the "Yes" or "No" button. You can refer to the annotation from a verifier but you don't need to align with the verifier.

Yes	No
-----	----

Figure 14: Instructions for annotating whether an LLM follows user misinformation in its response.