Inference-Time Decontamination: Reusing Leaked Benchmarks for Large Language Model Evaluation

Qin Zhu[1,2], Qinyuan Cheng[1,2], Runyu Peng[1,2], Xiaonan Li[1,2], Tengxiao Liu[1,2], Ru Peng[3], Xipeng Qiu[1,2*], Xuanjing Huang [1,2],

[1]School of Computer Science, Fudan University,
[2]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University,
[3]College of Computer Science and Technology, Zhejiang University,

Correspondence: {zhuq22,rypeng22}@m.fudan.edu.cn
{chengqy21,lixn20,xpqiu,xjhuang}@fudan.edu.cn rupeng@zju.edu.cn

Abstract

The training process of large language models (LLMs) often involves varying degrees of test data contamination (Yang et al., 2023b). Although current LLMs are achieving increasingly better performance on various benchmarks, their performance in practical applications does not always match their benchmark results. Leakage of benchmarks can prevent the accurate assessment of LLMs' true performance. However, constructing new benchmarks is costly, labor-intensive and still carries the risk of leakage. Therefore, in this paper, we ask the question "**Can we reuse these leaked benchmarks for LLM evaluation?**" We propose **I**nference-**T**ime **D**econtamination (ITD) to address this issue by detecting and rewriting leaked samples without altering their difficulties. ITD can mitigate performance inflation caused by memorizing leaked benchmarks. Our proof-of-concept experiments demonstrate that ITD reduces inflated accuracy by 22.9% on GSM8K and 19.0% on MMLU. On MMLU, using Inference-time Decontamination can lead to a decrease in the results of Phi3 and Mistral by 6.7% and 3.6% respectively. We hope that ITD can provide more truthful evaluation results for large language models.

1 Introduction

The emergence of large language models (LLMs) (Brown et al., 2020b; Touvron et al., 2023a; Zeng et al., 2023; Yang et al., 2023a; Cai et al., 2024; OpenAI, 2022; Taori et al., 2023; Chiang et al., 2023; Sun et al., 2024; Anthropic, 2023) has made the effectiveness of model capability evaluation crucial. Not only does it assist in ranking models, but it also helps in distinguishing valuable work and effective strategies for model improvement. Current LLMs are achieving increasingly better performance on various benchmarks. However, their

[*] Corresponding author.

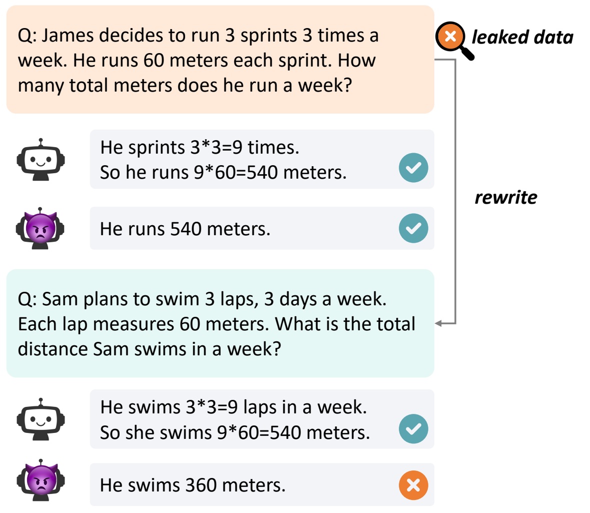

Q: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?
leaked data
He sprints 3*3=9 times.
So he runs 9*60=540 meters.
He runs 540 meters.
rewrite
Q: Sam plans to swim 3 laps, 3 days a week. Each lap measures 60 meters. What is the total distance Sam swims in a week?
He swims 3*3=9 laps in a week.
So she swims 9*60=540 meters.
He swims 360 meters.

Figure 1: Illustration of the function of Inference-Time Decontamination, aiming to discern whether a model passes the test by memorizing contaminated data. 👿 and 🤖 means the LLM delibterately memorizes and deos not memorize this case.

performance in practical applications does not always match their benchmark results (Huang et al., 2023). This suggests that the superior performance of LLMs on benchmarks might be due to intentional or inadvertent data contamination (Golchin and Surdeanu, 2023b; Li and Flanigan, 2024; Yang et al., 2023c). LLMs are potential cheaters.

The impact of potential data contamination on model evaluation encourages researchers to establish new benchmarks for a more accurate assessment of model performance (White et al., 2024; Li et al., 2024; Zhang et al., 2024). However, any benchmark faces risk of leakage once it is publicly available. Many benchmarks are fully open, leading to varying degrees of data leakage that affect the authenticity and fairness of model evaluations. As models are trained on increasingly large datasets, it becomes likely that benchmark-related data contaminates the training sets, causing LLMs to inadvertently cheat. Additionally, creating new benchmarks is both labor-intensive and costly, making it a less favorable solution for addressing test data contamination.

Can we reuse high quality leaked benchmarks for LLM evaluation? As shown in Figure 1, when a test sample is used for model training, there are two possibilities: 1. The model may learn relevant knowledge and skills, such as using chain-of-thought reasoning. In such cases, even if we make modifications to the question, the model can still provide correct answers. 2. The model might simply memorize the correct answer and directly copy them, rather than learn the skill. In this case, if we alter the background of the questions, without changing the essence of what is being tested (such as a mathematical formula), the model may fail to provide correct answers. For the first scenario, the model has achieved generalization, and such leaked data can no longer be used. For the second scenario, there is a possibility that such leaked data could be revived.

In this paper, we propose **I**nference-**T**ime **D**econtamination (ITD) to mitigate the inflation of evaluation results caused by models simply memorizing answers. ITD maximizes the value of existing high quality benchmarks avoids the substantial cost of constructing new benchmarks (Wu et al., 2023). Specifically, we first use a detector to screen for potentially leaked samples and then rewrite these samples, attempting to mitigate the impact of memorizing answers, without changing the sample's difficulty. For two types of tasks, we propose two rewriting methods. For knowledge-related benchmarks like MMLU (Hendrycks et al., 2021), we keep the knowledge points tested by the original sample unchanged and rewrite the phrasing of the questions. For math benchmarks related to model reasoning abilities like GSM8K (Cobbe et al., 2021), we maintain the specific numbers and calculations involved in the original data unchanged, but rewrite the background of the questions.

We conduct experiments on two fundamental benchmarks, GSM8K and MMLU. To validate the feasibility of Inference-Time Decontamination, we first perform *proof-of-concept experiments*. We intentionally leak half of the test data to train a model, and then test it with Inference-Time Decontamination (ITD). We find that after using ITD, the model's accuracy on GSM8K and MMLU decreased by 22.9% and 19.0%, respectively. We also study the effectiveness of ITD *in real evaluation environments* on popular large language models, Phi-3 and Mistral. After the application of ITD, Phi-3 showed reductions of 5.3% on GSM8K and 6.7% on MMLU, while Mistral experienced smaller reductions of 0.5% on GSM8K and 3.6% on MMLU. It indicates that the extent of adjustments by ITD for the models is as expected, achieving the goal of mitigating performance inflation caused by memorizing benchmarks and providing more valuable and reliable evaluation results. Our core contributions are:

1. We propose **I**nference-**T**ime **D**econtamination (ITD) to mitigate the inflation of evaluation results caused by data contamination.

2. We conduct proof-of-concept experiments that demonstrates ITD can effectively mitigate the biased performance resulting from models memorizing benchmarks.

3. We test ITD on two commonly used LLMs and find that their performance on both MMLU and GSM8K decrease to varying degrees.

4. We release a rewritten GSM8K dataset and a rewritten MMLU dataset sampled by categories to facilitate future evaluation work. [1].

2 Related Work

Contamination Detection Traditional contamination detection methods directly calculate the overlap between pre-training data and evaluation datasets, including n-gram analysis (Touvron et al., 2023b; OpenAI, 2023; Team et al., 2023; Bai et al., 2023) and BM25 (Jiang et al., 2024) for indexing and matching. However, as pre-training data grows exponentially, even simple n-gram statistics become extremely resource-intensive. Yang et al. (2023c); Gunasekar et al. (2023) find n-gram detection unreliable due to unintentional contamination risks. More importantly, training corpora for mainstream LLMs are mostly inaccessible, so recent research has turned to focus on: i)-exploiting the distributional differences between the benchmark training set and the test set to evaluated (Xu et al., 2024). ii)-Evaluate sample-level contamination by providing text segments and black-box access to the LLM (Shi et al., 2023). Other work evaluates contamination through LLM-generated

[1]We will release our data and code at `https://github.com/8188zq/Inference-Time-Decontamination`.

content, limited by the LLM's comprehension abilities to instruction (Deng et al., 2023; Golchin and Surdeanu, 2023a). Some studies test if models can coherently continue a given sample part (Golchin and Surdeanu, 2023b). Contamination detection remains a critical concern that should be addressed in benchmarks rather than affecting a fair assessment of the model's capabilities.

Decontamination Decontamination involves avoiding or mitigating the negative effects of contamination. Typically, decontamination applied in the training phase, model developers using various methods to remove these overlap between pre-training data and evaluation data (OpenAI, 2023; Touvron et al., 2023b; Radford et al., 2019; Brown et al., 2020a; Chowdhery et al., 2023). Besides, new datasets can also be created to avoid contamination The LatestEval (Li et al., 2024) avoids model contamination by strictly adhering to a temporal sequence, using texts published within a recent time window to construct new question-answer sets from the latest Wikipedia data. However, this method is also transient; once the new dataset is released, it is exposed to leakage risks, necessitating constant updates and making old benchmark results obsolete. Similarly, Scale AI creates a new dataset, GSM1K (Zhang et al., 2024), ensuring comparability on critical metrics such as human solve rates, number of solving steps, and answer magnitude. They prevent data leakage by not releasing the dataset. Livebench (White et al., 2024) also tries to limit potential contamination by releasing new questions monthly. However, these methods all require a significant amount of additional overhead.

3 Method

3.1 Problem Formulation

In this paper, we focus on the inference-time decontamination problem. Given a language model f_θ and an out-of-distribution evaluation dataset $E = \{x_i\}_{i\in[m]}$ with potential contamination, our objective is to evaluate a model's performance without access to pre-training data. Define a contamination indicator function $c : E \rightarrow \{0, 1\}$, where $c(x) = 1$ if the sample x appears in model pre-training, otherwise $c(x) = 0$. The function c is unknown yet possible to approximate. Thus, our work is to develop an contamination indicator $\hat{c}$ to adjust the evaluation dataset E and compute the final evaluation result $\hat{M}$:

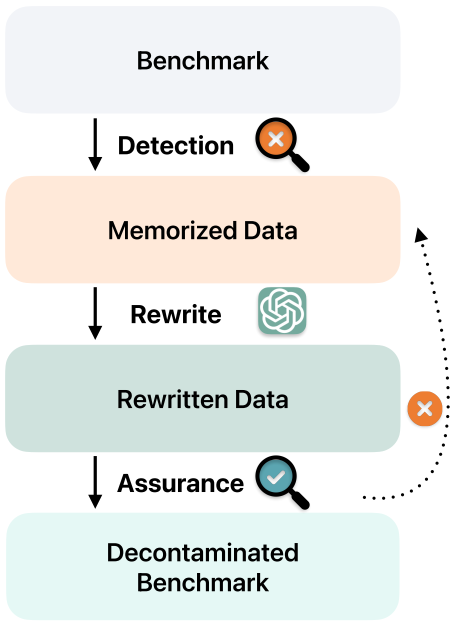


Figure 2: Overview of inference-time decontamination.

$$\hat{M} = M(f_\theta(E'), \hat{c}), \tag{1}$$

where E' is the adjusted evaluation dataset.

3.2 Inference-Time Decontamination

We show the overview of our framework in Figure 2, which consists three stages called **Detection**, **Rewrite** and **Assurance**.

Detection First, we conduct contamination detection on each evaluation sample based on the LLM to be evaluated. This allows the original dataset to be splitted into two parts: uncontaminated and potentially contaminated. Notably, according to our task definition, the split should ideally be based on whether the model's response is entirely from memorization. However, since memorized data is a subset of seen data, we follow Shi et al. (2023) to use the training data detection method MinKProb to substitute $\hat{c}$. The detecting method is an approximation of a contamination indicator function $c : E \rightarrow \{0, 1\}$. Since memorized data is a subset of pre-training data, the training data detection method is an alternative of $\hat{c}$.

The goal of MinKProb is to determine whether a text X appears in the pre-training data of an LLM. MinKProb obtains the probability $P(x_i)$ for each token x_i in the text X and selects the K tokens with the lowest probabilities $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ and calculates the average of these probabilities:

$$\text{MinKProb}(X) = \frac{1}{K} \sum_{j=1}^{K} P(x_{i_j}), \quad (2)$$

where $P(x_{i_j})$ is the probability of the j-th lowest probability token x_{i_j}, with a threshold ϵ to determine whether X appears as pre-training data.

Rewrite A suitable rewriting method involves rewriting by skilled individuals, based on the original questions according to some clear rules and instructs, similar to the construction method of GSM1k (Zhang et al., 2024). However, this expensive approach creates a new dataset rather than sticking to the original one, which violates the principle of revival. We propose an automated generation method to rewrite the potentially contaminated parts identified in the detection stage while not altering the level of challenge. We explore rewriting methods for two of the most typical and popular tasks: mathematical reasoning and knowledge-based datasets.

For mathematical reasoning problems, such as GSM8K, we redesign problem scenarios based on the original problem's computational logic and answer structure. This ensures that while maintaining the difficulty and answers, the problems become more diverse. For example, application contexts in the original questions, such as eggs, can be changed to candies, and the involved characters can be replaced with different roles. However, all numbers involved and the mathematical steps in the answers remain consistent. This approach ensures both the consistency of the content being assessed and the difficulty. The GSM1k dataset states that their problem-solving steps and distributions are similar to those of GSM8K to ensure consistent difficulty, and our verification shows that the step distribution and numbers are completely identical.

For knowledge-based problems, such as MMLU, we found that rewriting the questions requires a vast knowledge base, and any change in the background requires verification of correctness. Therefore, we choose to perform synonymous rewrites of the questions and options without changing core proper nouns and any numbers. We avoid using uncommon words that could increase the difficulty of understanding, thereby ensuring the consistency of the knowledge points and difficulty.

The prompts used, examples of rewritten questions on both GSM8K and MMLU can be found at the appendix B.

Dataset	Llama2	Mistral	Phi-3
GSM8K	0.56	0.32	0.47
MMLU	0.28	0.23	0.25

Table 1: Threshold value ϵ for different models on GSM8K and MMLU datasets.

Assurance We re-detect the modified parts. This is necessary because some models may have undergone extensive in-domain training data, causing the rewritten results to still be within the model's memory. Therefore, we iterate through the first detection step and the second rewriting step until the rewritten content passes the detection or reaches the maximum number of iterations. We also conducted human evaluations to assure the quality of the decontaminated data.

4 Experiment

4.1 Setup

Dataset We conducted experiments on two top influential benchmarks corresponding to two types of problem datasets: *knowledge-based dataset*, MMLU (Hendrycks et al., 2021) and *mathematical reasoning* GSM8K (Cobbe et al., 2021). The evaluation set for GSM8K and MMLU contains 1,319 and 14,042 data points, respectively. However, it should be noted that in our proof-of-concept experiment, we trained Llama2 to achieve intentional data leakage, resulting in *a high contamination rate* for the trained Llama-contaminated. This led to significant costs during the rewrite stage of ITD due to the API calls. Therefore, we randomly sampled from the Llama2_contaminated training data for MMLU according to the 17 official categories, with 50 samples randomly sampled from each category, resulting in a total of 850 samples, called *MMLU**. For GSM8K, we followed the recommended prompt (*8-shot*) for evaluation as suggested by Chain-of-thought (Wei et al., 2022). For MMLU, we used the official prompt (*5-shot*) provided by MMLU for model evaluation. Both used a greedy generation strategy.

Model We conducted evaluations and studies on three popular models: Llama2-7b-base (Touvron et al., 2023b), Mistral-7b-base (Jiang et al., 2023), and Phi-3-mini-128k-instruct (Abdin et al., 2024).

ITD-Detecting settings We use two detectors in our experiments, MinKprob as mentioned in section 3.2 and *All* which refers to a detector that flags all inputs as leaked. In the implementation of the detector MinKprob, we need to determine two hyper-parameters: K and ϵ. They are assessed using the evolutionary metrics described in section 3.2. By exhaustively searching for the maximum difference in MinKprob before and after rewriting, we determined $K = 20$. By exhaustively searching for the highest classification accuracy on the constructed seen and unseen sets, We determined ϵ as shown in Table 1. The experimental details can be found in the Appendix A.

When calculating the average probability, the input setting used is *0-shot* to avoid interference from the official prompts provided by the evaluation set or Chain-of-Thought (CoT) (Wei et al., 2022). This is because these prompts are likely to be leaked (included in other data released by the benchmark), while the questions are not leaked. Including these prompts could result in many questions being falsely identified as potentially contamination, affecting the accuracy of the detector.

ITD-Rewriting settings We designed two different rewriting methods to address the two types of evaluation sets. The generation model used is GPT-4 ("gpt-4-0613") (OpenAI, 2023), with the temperature set to 1, and two examples provided. The examples are shown in the appendix. The maximum number of rewrites is 3 times. Except for the first rewrite, each rewrite is based on the previous rewrite result.

Since we will evaluate multiple models multiple times, we constructed a cache dataset to reuse the rewrite results, allowing us to select which round to use as needed. This speeds up the evaluation process, controls the randomness of rewrites, and facilitates comparisons between models. We released these datasets, which include over 4,000 entries for GSM8K and more than 2,500 entries for MMLU, to facilitate future evaluations and provide samples for assessing rewrite quality.

Evaluation Metrics For the evaluation results of both datasets, we used accuracy to measure the model's ability to provide correct answers, facilitating comparison between models. When analyzing the metrics of the same model before and after rewriting the test samples, we used rate of change (ROC) to measure it:

$$\text{ROC} = \left(\frac{V_t - V_{t-1}}{V_{t-1}}\right) \times 100\%. \qquad (3)$$

We use ϵ to represent the threshold for the MinKProb detect method. A sample's MinKProb exceeding ϵ indicates that it is classified as contaminated data.

4.2 Proof-of-concept Experiment

We show the results of the proof-of-concept experiments in Table 2. This experiment is conducted on LLama2-contaminated, the model we obtain after training based on Llama2-7b-base. The maximum number of rewriting steps is 3.

The purpose of this training is to find a model that meets the requirement of having both seen and unseen data in a high-quality dataset, where we can clearly and accurately distinguish between the two. Currently, no contamination detection method can achieve this. Therefore, we chose the base model of Llama2 and artificially exposed a part of the test set data to it through training. Specifically, we divide the pre-set seen and unseen sets according to the average accuracy of Llama2-7b-base on the two datasets, ensuring that each subset had the same accuracy distribution as the original dataset, specifically 0.126 on GSM8K and 0.459 on MMLU. For MMLU, we use the typical multiple-choice prompt `"<Question>\nA.<Choice_A>\nB.<Choice_B>\nC.<Choice_C>\nD.<Choice_D>"`. For GSM8K, we use the chain-of-thought prompt `"Question: <question>\nAnswer: Let's think step by step.\n<answer>"` (Liu et al., 2023). We then use the seen set as training data and conducted training for 1 epoch relied on Fastchat (Zheng et al., 2023). In particular, we utilize the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 2e-5. We traine Llama2 for 3 epochs and other models for 2 epochs, with a batch size of 8 and a warm-up ratio of 0.03. We conducte all experiments with A100 GPUs.

The experimental results show that *Llama2-contaminated* achieves significant improvements on both MMLU and GSM8K after being artificially exposed to the data. But after using ITD, the model's accuracy on GSM8K and MMLU decreased by 22.9% and 19.0%, respectively. This proves the effectiveness of our proposed ITD. The models indeed exhibit the phenomenon of artificially inflated evaluation scores by relying solely

Dataset	Detector	Seen		ITD		ROC	Unseen	
		Acc.	Leaked Rate	Acc.	Leaked Rate		Acc.	Leaked Rate
GSM8K	MinKProb	40.1	62.7%	30.9	0.3%	22.9%	18.6	1.2%
	All	40.1	-	28.8	-	28.2%	18.6	-
MMLU*	MinKProb	87.5	79.4%	70.9	21.2%	19.0%	53.6	29.8%
	All	87.5	-	61.3	-	29.9%	53.6	-

Table 2: Results of the proof-of-concept experiment on GSM8K and MMLU. Tested models indeed exhibit the phenomenon of artificially inflated evaluation scores by relying solely on memorized leaked data. ITD successfully mitigates performance inflation caused by memorizing benchmarks. "All" refers to a detector that flags all inputs as leaked. MMLU$\star$ denotes a sampled dataset instead of the whole MMLU dataset.

on memorized leaked data. ITD successfully mitigates performance inflation caused by memorizing benchmarks.

We also observe a slight improvement in the model's performance on the unseen set compared to the original values. This indicates that in-domain training does indeed help generalize some capabilities,such as using chain-of-thought reasoning, aiding the model in answering specific types of questions.

The experiments also compare the results under two detection schemes. Using MinKProb as the detector, we are able to detect 62.7% and 79.4% of contamination in the seen set, with corrections of 22.9% and 19.0% in the evaluation metrics after rewriting, respectively. However, in fact, this part of the tested data belongs to the data we have intentionally leaked. An excellent detector should ideally detect 100% contamination, but clearly, MinKprob only detected 62.7% and 82.1%, respectively. This indicates that although this detection method is feasible and one of the most popular solutions, it still suffers from significant accuracy loss. This demonstrates that there is still considerable room for research in this area.

In contrast, using an extremely strict detection scheme that can detect 100% contamination at the sacrifice of increasing the overhead by approximately 300%, we achieve corrections of 28.2% and 29.9% in the metrics after rewriting, validating the effectiveness of our setting in addressing questions that models answer correctly solely based on memorization.

Moreover, the second scheme of All detection illustrates the upper limit of our rewriting setting and suggests that there is still significant room for improvement in the chosen detection scheme. However, in practical scenarios, we cannot know the distribution of the seen set in advance, and most evaluations are unlikely to have a 100% contamination level. Due to the uncertainty of the data's contamination degree and the multiplied additional overhead, we cannot simply use this extremely strict detection scheme. A reliable detection method remains necessary and effective.

Additionally, an interesting finding is that the leaked rate of llama2-7b-base on the unseen datasets differs between the two datasets: only 1% on GSM8K, but nearly 30% on MMLU. This suggests that llama2 is almost uncontaminated on GSM8K, while there may be a certain degree of contamination on MMLU.

4.3 Real Model Experiment

In this section, we evaluate the effectiveness of our setting in a real evaluation environment by testing Mistral-7b-base and Phi-3-mini-128k-instruct on two datasets without any knowledge of the training data. The results are shown in Table 3. We employ two types of detection methods: one using MinKprob and the other using an extremely strict detection scheme that assumes all inputs are leaked. We find that in most cases, with the use of ITD, the model's evaluation scores decreased, indicating that the questions originally answered correctly by relying on memorized benchmarks returned to their normal level after the rewrite. In comparison, Phi3-mini exhibits a higher level of contamination and underwent greater modification. However, as with our reasoning for conducting the proof-of-concept experiment, we cannot be entirely certain about the nature of the contamination without having the exact training data for the model to compare against. Therefore, it is impossible to provide an accurate assessment of the additional overhead and accuracy.

Dataset	Model	Detector	Origin		ITD		ROC
			Acc.	Leaked Rate	Acc.	Leaked Rate	
GSM8K	Phi3-mini	MinKProb	79.8	52.8%	75.6	14.9%	5.3%
		All	79.8	-	73.0	-	8.5%
	Mistral-7b	MinKProb	41.7	0.1%	41.4	0%	0.5%
		All	41.7	-	39.7	-	4.8%
MMLU*	Phi3-mini	MinKProb	73.3	53.2%	68.4	15.9%	6.7%
		All	73.3	-	61.8	-	15.7%
	Mistral-7b	MinKProb	76.8	53.4%	74.0	13.1%	3.6%
		All	76.8	-	66.7	-	13.2%

Table 3: Results of real model experiment on GSM8K and MMLU datasets. In real evaluation scenarios, models still exhibit the phenomenon of artificially inflating scores by relying solely on memorized leaked data. ITD can still mitigate performance inflation caused by memorizing benchmarks. "All" refers to a detector that flags all inputs as leaked. MMLU$\star$ denotes a sampled dataset instead of the whole MMLU dataset.

Nevertheless, given MinKProb's high rate of missed detections in the proof-of-concept experiment, we are concerned about its accuracy in determining contamination, leading to the detection of relatively low contamination rates, especially with mistral only showing 0.1% on GSM8K. According to MinKprob's results, mistral exhibits almost no contamination on GSM8K, thus there is very little data rewriting, and the degree of correction is evidently low. In contrast, the second detection method , at the cost of significant additional overhead, results in more substantial corrections, providing us with a reference for the upper limit of inference-decontamination and also validating that our setting remains effective in a real-world evaluation environment.

As discussed in Section 3.2, the presence of a detector is meaningful for two reasons: first, it reduces additional costs, especially as the number of rewrites increases and the contamination level is not particularly high. Second, some models have undergone extensive in-domain training, which might lead to situations where even rewritten results are familiar to the model. We aim to use detection to specifically identify these cases. Without detection, it would be impossible to distinguish between mild and severe contamination, as both would show little fluctuation.

4.4 Analysis

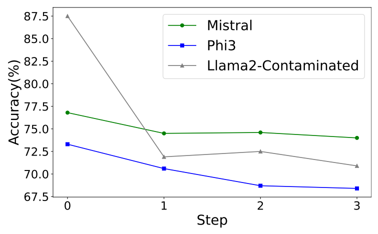
Quality Checks We conduct experiments to verify the consistency of difficulty before and after rewriting. The results are shown in Table 4. For the Llama2-contaminated inference-decontamination on the MMLU and GSM8K datasets, we generate

Model	GSM8K		MMLU*	
	Origin	Rewritten	Origin	Rewritten
Llama2-contaminated	40.1	30.9	87.5	70.9
Llama2-7b-base	12.6	13.3	45.7	44.0

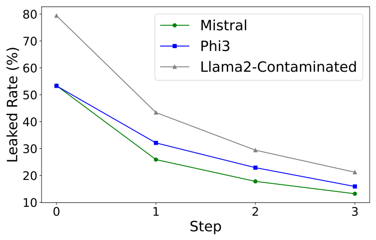
Table 4: Comparison of Accuracy Changes Between Llama2-contaminated and Llama2 on Identically Rewritten Data.

both original and rewritten versions of the data. We evaluate these versions using Llama2-7b-base. As discussed in Section 4.2, the contamination rate of Llama2 on GSM8K is negligible, while MMLU is partially contaminated. Thus, using rewritten data for GSM8K should result in minimal fluctuation, whereas MMLU may show slightly more variation. Llama2-base shows minimal variation on the same rewritten data, indicating that the difficulty of questions before and after rewriting for Llama-contaminated remains unchanged within the margin of error.

Human Evaluation We also conduct human evaluation on rewritten data. We carefully compare the rewritten samples with the origin ones to assure the stability of problem difficulty and answer correctness. Two aspects are focused to check the problem difficulty: whether the desciption is simplified, and whether extra information is added. We detect at most 11.76% samples in which the rewritten phrases are more common and straight forward in MMLU* and 8.03% in GSM8K, yet containing no actual difficulty change. We also assess the stability of answer correctness by verifying whether the answer changes as a result of modifying the problem description. We only find



(a) Accuracy Curve with Rewrite Steps on MMLU



(b) Leaked Rate with Rewrite Steps on MMLU

Figure 3: Impact of Different Rewriting Steps. A single rewrite is sufficient to significantly mitigate the model's performance inflation. However, some rewritten data may still be classified as contaminated. Multiple rewrites can further alleviate this issue.

answer shift in 3.9% of GSM8K samples at any step during rewriting. Notably, although the answers for these questions changed, contrary to our expectation of no numerical changes, the provided reference answers remained correct.

Impact of Rewriting Iterations We analyze the impact of different rewriting steps on the accuracy and contamination rate for three models: Mistral, Phi3, and Llama-contaminated. The results in Figure 3 show a noticeable drop in both accuracy and contamination rate after the first rewriting step, indicating the significant impact of this initial rewrite. Data not passing the detector in the Assurance stage underwent multiple rewrites. During these rounds, both accuracy and contamination rate continued to decrease and eventually leveled off, showing a convergent trend. This suggests that while some data remain contaminated after the first rewrite, subsequent rounds effectively reduce contamination and stabilize accuracy. This analysis underscores the necessity of multiple rewriting rounds and the Assurance stage. The persistent decline and stabilization in both metrics validate the iterative rewriting process, highlighting its importance for achieving cleaner and more reliable datasets.

Performance of Contaminated vs. Uncontaminated Data We analyze the performance of contaminated versus uncontaminated data with different rewriting steps for Llama2-contaminated on the GSM8K dataset, as shown in Figure 4. The figure reveals that Initially, contaminated data show significantly higher accuracy than uncontaminated data. However, with each rewriting step, the contamination rate drops substantially, indicating that many contaminated data points are corrected and reclassified as uncontaminated. The rewritten data's accuracy eventually matches that of the uncontaminated data, demonstrating the effectiveness of the rewriting steps. Some data still flagged as contaminated after the first rewrite maintain high accuracy but are reclassified as uncontaminated after multiple rewrites, leading to a significant reduction in contamination rate. This analysis confirms that iterative rewrites effectively transform contaminated data into reliable, uncontaminated data, ensuring dataset accuracy and integrity.

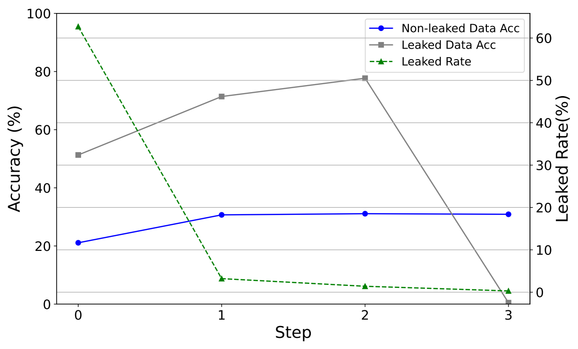


Figure 4: Performance of Contaminated vs. Uncontaminated Data with Different Rewriting Steps for Llama2-contaminated on GSM8K. For contaminated data, the model shows *fake high performance*(51.3%). After several rewrites , the data becomes uncontaminated, and performance returns to normal(30.9%).

5 Conclusion

This paper explores eliciting truthful answers from a language model by addressing the impact of data contamination on model evaluations. We propose an inference-time decontamination method involving detection and iterative rewriting of contaminated data, leading to more accurate model performance assessments. Experiments on GSM8K and MMLU benchmarks suggest that our method can mitigate contamination effects, resulting in more reliable evaluation results.

Our framework's detection, rewrite, and assurance stages allow for consistent and fair assessments without needing entirely new datasets. The reduction in contamination's impact highlights the promise of our approach in providing a realistic view of model capabilities.

We believe this work lays a foundation for future research in improving language model evaluations. Further exploration of advanced detection and rewriting techniques will continue to enhance the reliability and fairness of these assessments.

Limitations

Limited evaluation criteria for Real Models: The correction magnitude for real models is not substantial. However, since the specific contamination relationship between a model and the benchmark is still unknown, it is impossible to provide an effective evaluation without a reliable detecting method.

Effectiveness in the Worst-case Scenarios: For models that have intentionally trained on a large amount of in-domain data to improve performance, or even followed our revival steps for extensive rewrites, our current two types of rewrite methods have limited effectiveness. Nonetheless, due to the detect stage in our setting, we can distinguish these models from those with no contamination. We also look forward to exploring more efficient rewrite methods in the future.

Evaluation Biases The different contamination levels across models lead to variations in the final evaluation questions. Additionally, using model-automated rewriting to reduce costs may introduce biases. We discuss these issues here. Firstly, as shown in our Quality Checks experiments, we ensure that each rewrite maintains consistent difficulty and tests the same knowledge and skill points. Secondly, the bias introduced by model generation is relatively small compared to the false score increase from models memorizing original questions. This randomness affects all models. Furthermore, we use cached rewritten datasets to reduce costs and randomness, thereby increasing relative comparability between models.

However, the above points regarding the detecting methods and rewriting methods are not inherent limitations of our setting. They can be mitigated by using better rewriting and detecting methods in the future.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62441602). The computations in this research were performed using the CFFF platform of Fudan University.

References

Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat S. Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219.

Anthropic. 2023. Introducing claude.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, and et al. 2024. Internlm2 technical report. *CoRR*, abs/2403.17297.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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

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

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2023. Investigating data contamination in modern benchmarks for large language models. *CoRR*, abs/2311.09783.

Shahriar Golchin and Mihai Surdeanu. 2023a. Data contamination quiz: A tool to detect and estimate contamination in large language models. *CoRR*, abs/2311.06233.

Shahriar Golchin and Mihai Surdeanu. 2023b. Time travel in llms: Tracing data contamination in large language models. *CoRR*, abs/2308.08493.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

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

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. 2024. Investigating data contamination for pre-training language models. *CoRR*, abs/2401.06059.

Changmao Li and Jeffrey Flanigan. 2024. Task contamination: Language models may not be few-shot anymore. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18471–18480. AAAI Press.

Yucheng Li, Frank Guerin, and Chenghua Lin. 2024. Latesteval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18600–18607. AAAI Press.

Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. Plan, verify and switch: Integrated reasoning with diverse x-of-thoughts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2807–2822. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. Detecting pretraining data from large language models. *CoRR*, abs/2310.16789.

Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Xiangyang Liu, Hang Yan, Yunfan Shao, Qiong Tang, Shiduo Zhang, Xingjian Zhao, Ke Chen, Yining Zheng, Zhejian Zhou, Ruixiao Li, Jun Zhan, Yunhua Zhou, Linyang Li, Xiaogui Yang, Lingling Wu, Zhangyue Yin, Xuanjing Huang, Yu-Gang Jiang, and Xipeng Qiu. 2024. Moss: An open conversational large language model. *Machine Intelligence Research.*

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2024. Livebench: A challenging, contamination-free llm benchmark.

Yang Wu, Yanyan Zhao, Zhongyang Li, Bing Qin, and Kai Xiong. 2023. Improving cross-task generalization with step-by-step instructions. *Science China Information Sciences*, abs/2305.04429.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *CoRR*, abs/2404.18824.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, Juntao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023a. Baichuan 2: Open large-scale language models. *CoRR*, abs/2309.10305.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. 2023b. Rethinking benchmark and contamination for language models with rephrased samples. *CoRR*, abs/2311.04850.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. 2023c. Rethinking benchmark and contamination for language models with rephrased samples. *CoRR*, abs/2311.04850.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130B: an open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. 2024. A careful examination of large language model performance on grade school arithmetic. *CoRR*, abs/2405.00332.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Experiment on Hyper Parameter

In the implementation of the detector, we need to determine two hyperparameters: K and ϵ. The selection criterion for K is the difference in MinKprob before and after rewriting, aiming for sensitivity to the impact of rewriting. For ϵ, Shi originally required obtaining some seen and unseen data corresponding to the model. However, this is cumbersome and challenging to find seen and unseen data with a similar domain under the setting of any black-box model. Therefore, our approach is to artificially construct a part of the seen set through training and approximate an unseen set through multiple rewrites. Specifically, we randomly sample from the test set and perform several rewrites. Then, we uniformly divide the rewritten test set D into seen and unseen sets. By training, we artificially expose the seen data. We then iterate over the threshold ϵ and perform binary classification on each data point in the rewritten D to determine whether it has been seen, selecting the threshold ϵ that yields the most accurate classification under a uniform distribution based on accuracy.

Experiment on ϵ By exhaustively searching for the highest classification accuracy on the constructed seen and unseen sets, We determined ϵ as shown in Figure 5,6.

Experiment on K By exhaustively searching for the maximum difference in MinKprob before and after rewriting, we determined $K = 20$ as shown in Figure 7.

B Rewrite details

B.1 Prompt

In this section, we describe the process used to rephrase the stems of fill-in-the-blank math questions using GPT-4.

For knowledge-related benchmarks MMLU, we keep the knowledge points tested by the original sample unchanged and rewrite the phrasing of the questions.

For math benchmarks related to model reasoning abilities GSM8K, we maintain the specific numbers and calculations involved in the original data unchanged, but rewrite the background of the questions.

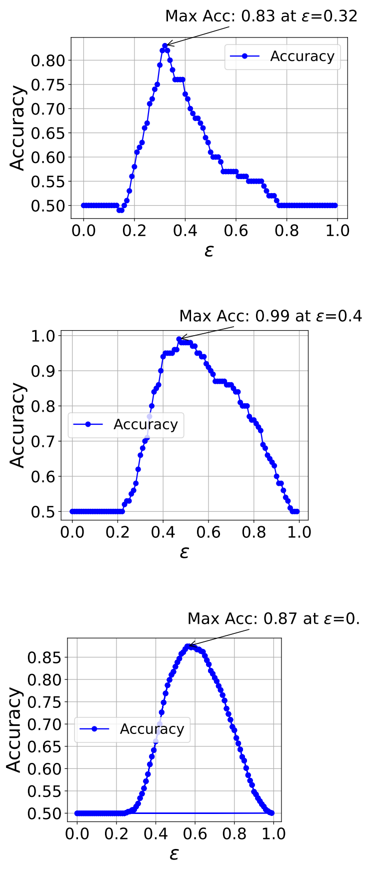


Figure 5: Hyper Parameter Search Experimen about ϵ on GSM8K.

The prompts used to guide the rephrasing process are designed to maintain the integrity and difficulty of the original questions while introducing diversity in the context and entities. The rephrasing involves the following guidelines:

Mathematical Reasoning Problems

1. The rephrased questions should not be a direct synonym replacement but can involve changing scenes and entities, such as replacing “eggs” with “candies,” as long as the numbers, operational logic, and final answers remain unchanged.

2. The main goal is to retain the precision needed for the correct fill-in response.

3. Ensure the semantics of the question stems are consistent before and after the rewrite.

4. The revised version should not introduce biases or clues that could unfairly simplify the question.

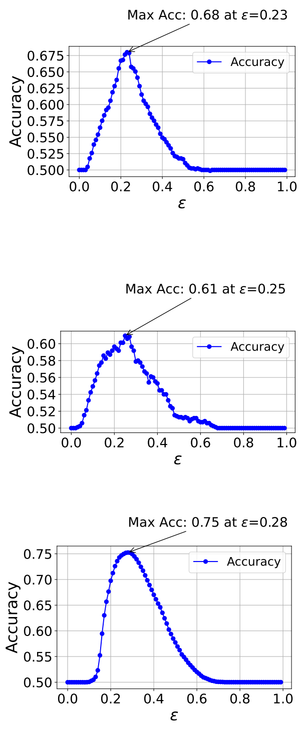


Figure 6: Hyper Parameter Search Experimen about ϵ on MMLU.

5. The rephrased question should be clear, concise, and maintain the original context and complexity.

6. The rewritten stem should facilitate a step-by-step problem-solving process without affecting the expected mathematical solution.

7. Changes to names are allowed as long as they do not confuse the identities of the characters involved.

This structured prompt ensures clarity in the rephrasing process and maintains the quality and difficulty level of the math questions.

We show the prompt in Table 5.

Knowledge-based Problem

1. The revised questions should maintain the original meaning and accuracy without any bias or hinting at the correct answer.

2. Ensure that the difficulty of understanding and solving the problem remains consistent before and after rewriting.

3. Use diverse expressions to avoid repetition but do not intentionally use uncommon words.

4. Do not alter mathematical expressions.

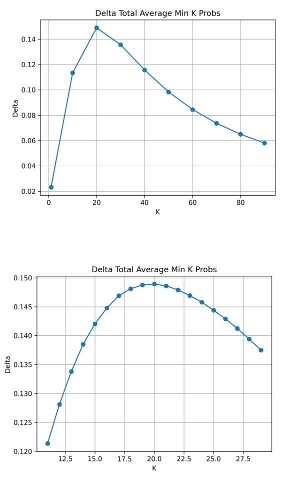


Figure 7: Hyper Parameter Search Experimen about K

Origin question: *A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?*
Rewritten question: *A pizza recipe requires 2 cups of flour and half that amount of water. How many cups of both ingredients does it require altogether?*

(a) Rewritten example for Mathematical Reasoning Problems

Origin question: *Boiling and freezing occur when water is subjected to*
A:decreased temperatures B:decreased atmospheric pressure C:increased temperatures D:increased atmospheric pressure
Rewritten question: *Water boils and freezes under what conditions?*
A:Lowering of temperatures B:Decrease in the pressure of the atmosphere C:Rising of temperatures D:Increasing the pressure of the atmosphere

(b) Rewritten examples

Figure 8: Hyper Parameter Search Experimen about ϵ on GSM8K.

This structured prompt ensures clarity in the rephrasing process and maintains the quality and difficulty level of the knowledge-based questions.

We show the prompt in Table 6.

B.2 Written Examples

We randomly sampled some rewritten examples of the two benchmarks. Examples of single-round rewrites can be found in Figure 8. And examples of multi-round rewrites can be found in the Table 7, 8.

More examples can be seen in the three-round rewritten data on the entire dataset we released.

Your task involves revising the stems of fill-in-the-blank math questions. For added diversity, this rewrite is not a direct synonym replacement; you can change scenes and entities—like replacing eggs with candies—as long as the numbers, operational logic, and final answers remain unchanged, without altering the difficulty level. The primary goal is to rephrase each question's stem—the main question or statement—in a way that retains the precision needed for the correct fill-in response. Ensure that the semantics of the question stems are consistent before and after the rewrite. The revised version should not introduce biases or clues that could unfairly simplify the question. It should be clear, concise, and maintain the original context and complexity. Furthermore, the rewritten stem should facilitate a step-by-step problem-solving process without affecting the expected mathematical solution, allowing changes to names as long as they do not confuse the identities of the characters involved.

Here are two examples for better understanding. Follow them and answer in json format:

Input:
Original_Question_Stem:
"Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?"
Answer:
"Natalia sold 48/2 = <<48/2=24>>24 clips in May. Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May. #### 72"
Output:
[
{
"Rephrased_Question_Stem": "In March, Marco gathered 48 seashells at the beach, and in April, he collected half as many. How many seashells did Marco collect in total during March and April?",
"Rephrased_Answer": "Marco collected 48/2 = 24 seashells in May. Marco collected 48 + 24 = 72 seashells altogether in April and May. #### 72",
}
]
Input:
Original_Question_Stem:
"Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?"
Answer:
"Weng earns 12/60 = $«12/60=0.2»0.2 per minute. Working 50 minutes, she earned 0.2 x 50 = $«0.2*50=10»10. #### 10"
Output:
[
{
"Rephrased_Question_Stem": "Each day, Kevin's bees produce 16 tablespoons of honey. He uses three tablespoons to sweeten his morning tea and four to make energy bars for his hiking group. He sells the leftover honey at the local co-op for $2 per tablespoon. How much money does Kevin earn daily from selling honey at the co-op?",
"Rephrased_Answer": "Kevin sells 16 - 3 - 4 = «16-3-4=9»9 tablespoons of honey a day. He earns 9 * 2 = $«9*2=18»18 every day at the local co-op. #### 18",
}
]
Start with this question and apply the instructions above:
Your Question Stem to Rephrase:

Table 5: The instruction for revising math question stems.

```
Revise the multiple-choice question and options to keep the meaning and accuracy without any bias or hinting at
the correct answer.Ensure that the difficulty of understanding and solving the problem remains consistent before
and after rewriting. Use diverse expressions to avoid repetition.But do not intentionally use uncommon words to
avoid repetition and do not alter mathematical expressions. Follow the example and answer in JSON format:

Input:
Original_Question_Stem: "During what historical period did the Renaissance take place?"
Original_Options: "(A)The Late Middle Ages (B)The Classical Antiquity (C)The Enlightenment (D)The
Industrial Revolution"
Output:
[
{
"Rephrased_Question_and_Options": {
"question": "In which historical era did the Renaissance occur?",
"A": "The historic era just before the Renaissance",
"B": "The period marking the transition from the Middle Ages",
"C": "The historic era synonymous with the Age of Reason",
"D": "The time period characterized by rapid industrialization"
}
}
]

Input:
Original_Question_Stem: [[Original_Question_Stem]]
Original_Options: [[Original_Options]]
Output:
```

Table 6: The instruction for revising multiple-choice questions and options.

Original Question: "Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens?"
Original Answer: "If each chicken eats 3 cups of feed per day, then for 20 chickens they would need 3*20=60 cups of feed per day. If she feeds the flock 15 cups of feed in the morning, and 25 cups in the afternoon, then the final meal would require 60-15-25=20 cups of chicken feed. #### 20"
Rephrased Question Stem (Step 1): "Every day, Carla prepares 3 liters of homemade insect food, full of protein and vitamins, for her butterflies. She divides the feed for her swarm of butterflies into three meals. In the morning, her butterfly population consumes 15 liters of the feed. In the afternoon, they consume another 25 liters. How many liters of insect food does Carla need to prepare for the final meal if she has 20 butterflies?"
Rephrased Answer (Step 1): "Each butterfly consumes 3 liters of feed daily, meaning for 20 butterflies Carla would need 3*20 = 60 liters of feed in a day. If they consume 15 liters in the morning, and 25 liters in the afternoon, then the final meal requires 60-15-25 = 20 liters of insect food. #### 20"
Rephrased Question Stem (Step 2): "At home, Julia bakes 3 chew sticks each day for her 20 parrots. She breaks down these sticks into three increments throughout the day. In the morning alone, her parrots munch down on 15 of these sticks, and they gnaw on another 25 in the middle of the day. How many chew sticks does Julia need to bake for the final session of the day?"
Rephrased Answer (Step 2): "Each parrot requires 3 chew sticks a day, so for her 20 parrots, Julia needs to bake 3*20 = 60 chew sticks every day. If they consume 15 in the morning and 25 in the afternoon, then the final munching session would require 60-15-25 = 20 chew sticks. #### 20"
Rephrased Question Stem (Step 3): "Simon has a habit of preparing 3 kilograms of bird food, filled with nutritious grains, for his parakeets every day. To cater to their feeding habits, he splits the food into three meals. In the morning, his flock of parakeets consumes 15 kilograms of bird food. Later in the afternoon, they consume another 25 kilograms. Given that he has 20 parakeets, how many kilograms of bird food does Simon need to portion out for their final feed of the day?"
Rephrased Answer (Step 3): "Each parakeet consumes 3 kilograms of bird food daily, which means that Simon would need to prepare 3*20 = 60 kilograms of bird food in a day for 20 parakeets. If 15 kilograms are consumed in the morning, and 25 kilograms in the afternoon, then Simon will need to prepare an additional 60-15-25 = 20 kilograms of bird food for the final meal. #### 20"

Table 7: Written examples for gsm8k.

Original Question: "A sixth-grade teacher is concerned because Kerry, a student in class, has been hostile to classmates. Which of the following teacher strategies is most likely to encourage Kerry to be more cooperative with classmates?"
Original Options: A: "Preventing Kerry from participating in play or recess activities as a consequence of hostile behavior" B: "Having Kerry memorize rules of behavior and write examples of how they would apply in the classroom" C: "Withholding attention or approval from Kerry in response to hostile behavior" D: "Implementing social skills training to teach Kerry appropriate replacement behaviors for hostile behaviors"
Rephrased Question and Options (Step 1): **Question:** "A student named Kerry in the sixth grade has been acting unfriendly towards peers. Which method could the teacher possibly use to promote more amicable relationships among Kerry and her classmates?" A: "Prohibit Kerry from joining in leisure or break-time activities as a repercussion for unfriendly conduct" B: "Instruct Kerry to learn behavior norms by heart, and to pen down how they can be enforced within the classroom" C: "Deny Kerry attention or appreciation when hostile conduct is exhibited" D: "Incorporate social abilities development to instruct Kerry on acceptable alternate behaviors to her antagonistic actions"
Rephrased Question and Options (Step 2): **Question:** "What strategy could a teacher employ to encourage Kerry, a sixth-grader, to cultivate better relationships with her peers, given her recent hostile behavior?" A: "Exclude Kerry from recreational breaks as punishment for her negative conduct" B: "Request Kerry to internalize good behavioral practices and document their implementation in class" C: "Avoid acknowledging or praising Kerry if she displays aggressive behavior" D: "Use techniques promoting the development of interpersonal skills to address and modify Kerry's non-cooperative behavior"
Rephrased Question and Options (Step 3): **Question:** "What approach could an educator utilize to motivate Kerry, a student in grade six, to improve her associations with classmates, considering her lately antagonistic behavior?" A: "Bar Kerry from leisure intervals as retribution for her adverse behavior" B: "Ask Kerry to absorb constructive behavioral norms and record their enactment in the classroom" C: "Overlook or restrain from complimenting Kerry if she exhibits hostility" D: "Apply strategies fostering the enhancement of social abilities to handle and transform Kerry's uncooperative conduct"

Table 8: Written examples for MMLU:high school psychology.