

# Project 2 - Benchmark Contamination in LLMs

## Trustworthy ML (ECE 696B): Spring 2025

Natnael Daba

April 26, 2025

### Abstract

Large Language Models (LLMs) have achieved remarkable performance on a variety of reasoning and problem-solving tasks. However, recent findings suggest that benchmark contamination, where test data leak into training, can inflate performance metrics. This project investigates benchmark contamination in two open-source reasoning models: DeepSeek-R1-Distill-Qwen-7B and Qwen2.5-Math-7B-Instruct, using the AIME-2024 and MMLU datasets. We evaluate two state-of-the-art contamination detection methods: TS-Guessing and Guided Prompting. Our experiments reveal that prompt design significantly affects contamination detection outcomes and that DeepSeek-R1 shows stronger signs of memorization, especially under guided prompting conditions. These findings highlight the need for robust contamination detection strategies and careful evaluation of LLMs on public benchmarks. The code used in this study is available at this repository.

## 1 Introduction

Large Language Models (LLMs) like ChatGPT [7], DeepSeek-R1 [1], and LLaMA [9] have shown strong performance on many natural language understanding and reasoning tasks. These models are trained on massive datasets and demonstrate strong performance on a wide range of tasks such as writing code, and solving math problems. However, a growing concern in the ML community is whether these models are truly reasoning or just memorizing parts of their training data. If the evaluation benchmarks are somehow leaked into the training set, the reported results might be misleading and not reflect the real reasoning capability of the model.

In this project, we study how benchmark contamination can affect the performance of open-source reasoning LLMs. We begin by evaluating two families of open-source models: 1) DeepSeek-R1-Distill-Qwen-7B and 2) Qwen2.5-Math-7B-Instruct on two math-focused datasets: AIME-2024 [6] and a subset of MMLU [11]. Our goal is to reproduce and compare the models' performance against their officially reported results. The evaluation setup and results are presented in Section 2.

To analyze the extent of potential benchmark contamination, we next survey several recent techniques for detecting overlaps between benchmark datasets and model training

corpora. Section 3.1 provides a summary of these techniques, including methods such as **Testset Slot Guessing** (*TS-Guessing*) [3] and guided prompting [5]. We then adopt and adapt two of these methods for our experiments. Rather than reimplementing from scratch, we build directly on the **LLMSanitize** library [8], which offers a modular and extensible suite of contamination detection tools. Our contributions extend this library with custom modifications tailored to our models, datasets, and evaluation goals. Details of our implementation, experimental results, and analysis are provided in Section 3.2.

## 2 Evaluating Performance on Benchmarks

In this section, we evaluate open-source, publicly available LLMs on two math-focused benchmarks: **AIME-2024** [11] and **MMLU** [6]. The AIME-2024 dataset comprises 30 problems from the American Invitational Mathematics Examination (AIME) 2024, a prestigious high school mathematics competition known for its challenging problems. The MMLU dataset consists of approximately 16,000 multiple-choice questions covering 57 subjects, including mathematics, philosophy, law, and medicine, designed to assess a model’s multitask precision in zero-shot and few-shot settings. Although the MMLU dataset contains a total of 270 questions in the category of "high school mathematics," we limit our evaluation to the first 100 samples in that category.

For this study, we select two open-source reasoning LLMs: **DeepSeek-R1-Distill-Qwen-7B** and **Qwen2.5-Math-7B-Instruct**. These models were chosen for their strong performance on mathematical reasoning tasks and their accessibility for benchmarking. We evaluate their accuracy on AIME-2024 and MMLU using multiple inference runs, analyzing their reasoning capabilities and potential reliance on memorization. Table 1 shows the result of this evaluation.

Model	MMLU ( <b>Pass@1</b> )	AIME24 ( <b>Pass@1</b> )
DeepSeek-R1-Distill-Qwen-7B	<b>96.0 ± 2.0</b>	<b>50.0 ± 9.3</b>
Qwen2.5-Math-7B-Instruct	72.0 ± 4.5	10.0 ± 5.6

Table 1: Mean accuracy and standard deviation (*Pass@1*) across 5 independent runs on the MMLU and AIME-2024 datasets.

## 3 Analyzing Benchmark Contamination

In this section, we look at a couple of methods for detecting benchmark contamination and summarize their main techniques and findings. We then select two techniques and apply them to the models evaluated in Section 2 to determine whether the benchmarks contain data that may have seen during pre-training or fine-tuning.

## 3.1 Literature Survey

### 3.1.1 PaCoST

Introduced in [12], PaCoST (Paired Confidence Significance Testing) is a method designed to detect benchmark contamination in large language models (LLMs) without need for access to their training data. The core idea is that if a model has seen a benchmark during training, it will likely be more confident when answering those questions compared to unseen, rephrased versions.

PaCoST works in three steps:

1. **Rephrasing Preparation:** For each benchmark question  $x$ , a paraphrased version  $x'$  is created using a strong language model, while keeping the correct answer  $y$  fixed.
2. **Confidence Estimation:** The model’s confidence is measured for both the original and rephrased instances using a technique called  $P(\text{True})$ , which queries the model to judge whether its own answer is correct.
3. **Significance Testing:** A paired t-test is performed to check if the confidence on the original is significantly higher than the rephrased version. If  $p < 0.05$ , the model is considered potentially contaminated on that benchmark.

Algorithm 1 shows the PaCoST method for testing contamination in benchmarks. Figure 1 shows an overview of the PaCoST method.

---

**Algorithm 1** PaCoST [12]

---

- 1: **Input:** Benchmark dataset  $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , test model  $M$ , rephrasing model  $M_p$
  - 2: **for**  $i = 1$  to  $n$  **do**
  - 3:    $x'_i \leftarrow M_p(x_i)$  ▷ Rephrase input using a rephrasing model
  - 4:    $c_i \leftarrow \mathcal{P}(\text{Yes} \mid x_i, M(x_i), M)$  ▷ Model’s confidence on original input
  - 5:    $c'_i \leftarrow \mathcal{P}(\text{Yes} \mid x'_i, M(x'_i), M)$  ▷ Model’s confidence on rephrased input
  - 6: **end for**
  - 7:  $\bar{d} \leftarrow \frac{1}{n} \sum_{i=1}^n (c_i - c'_i)$  ▷ Average drop in confidence
  - 8:  $s_d \leftarrow \sqrt{\frac{1}{n-1} \sum_{i=1}^n (d_i - \bar{d})^2}$  ▷ Standard deviation of differences
  - 9:  $t \leftarrow \frac{\bar{d}}{s_d / \sqrt{n}}$  ▷ Compute t-statistic
  - 10: Compute  $p$ -value based on  $t$  and  $n$
  - 11: **if**  $p < 0.05$  **then**
  - 12:   **Return:**  $D$  is Contaminated ▷ Significant confidence gap  $\rightarrow$  likely contamination
  - 13: **else**
  - 14:   **Return:**  $D$  is Not Contaminated
  - 15: **end if**
- 

PaCoST is designed to avoid common limitations in earlier methods: it doesn’t require training data access, is robust to both instruction and answer contamination types, does

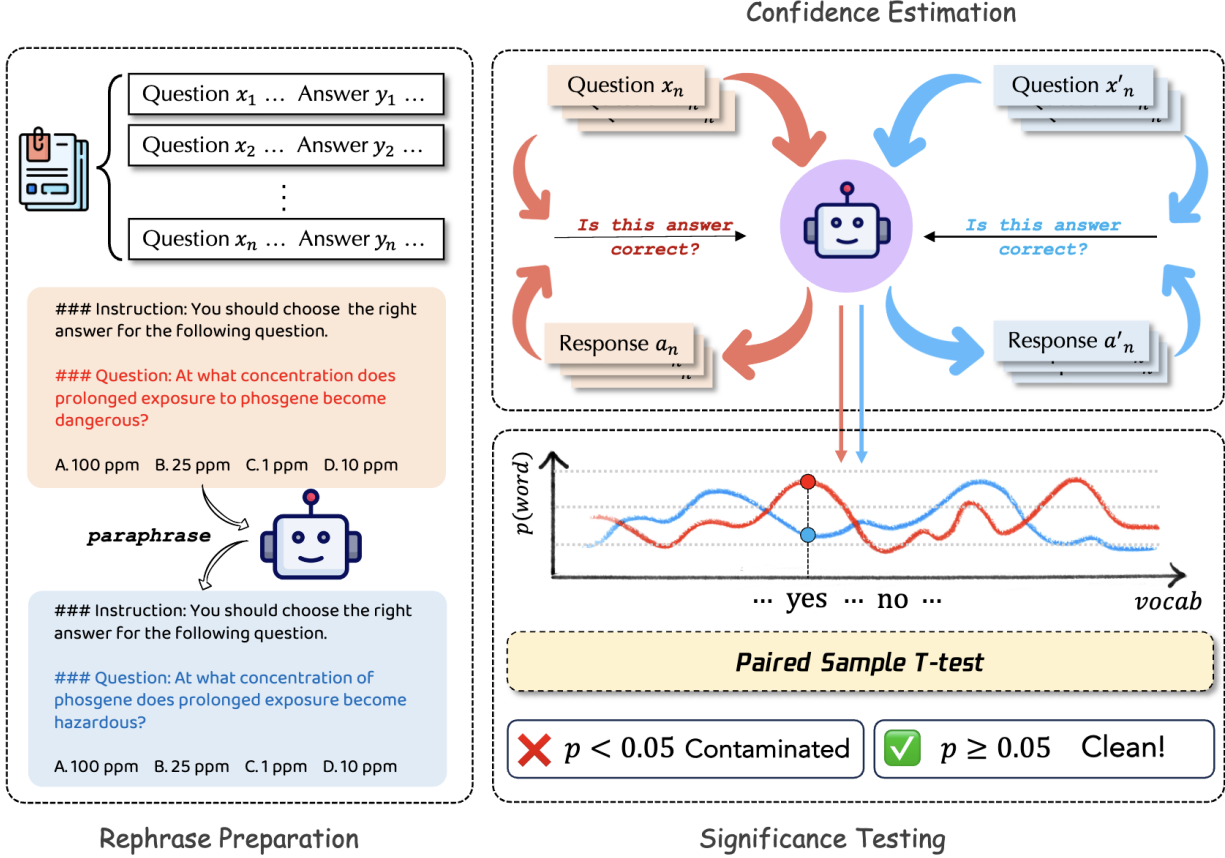


Figure 1: Overview of PaCoST [12] method.  $x_i$  represents a question,  $y_i$  represents its corresponding ground truth answer,  $x'_i$  represents a rephrased question and  $a_i, a'_i$  represent model responses to original and rephrased question correspondingly. Image source: ??

not rely on arbitrary thresholds, and is stable across prompts and datasets. Experimental results show PaCoST outperforms other methods like Guided Prompting [5] and DCQ [4] in both effectiveness and reliability.

### 3.1.2 LLM Decontaminator

This method, proposed in [10], focuses on the challenge of **semantic contamination**, where rephrased or paraphrased benchmark examples unintentionally end up in a model’s training set. These subtle overlaps can cause misleading benchmark results.

This paper shows:

- Training on rephrased benchmark questions significantly boosts model performance.
- Even relatively small models (13B) can achieve GPT-4-level accuracy on benchmarks like MMLU and HumanEval after exposure to rephrased test data.
- Existing decontamination methods often fail to detect such contamination, which can inflate perceived generalization.

## Main Experimental Setup

- The authors simulate contamination by creating rephrased versions of benchmark datasets: *MMLU*, *GSM-8k*, and *HumanEval*.
- Paraphrasing is done to preserve the meaning while changing the wording.
- Models (1.3B and 13B) are fine-tuned on these rephrased versions and then evaluated on the original benchmarks.

## Key Results:

- A 13B model trained on rephrased MMLU achieves 81.2% accuracy (close to GPT-4’s 86.4%).
- Similar gains are observed on GSM-8k and HumanEval.
- These findings suggest semantic leakage may have a stronger impact than model scale or instruction tuning alone.

## Evaluation of Existing Detection Methods

Method	Pros	Cons
N-gram Overlap	Simple, fast	Misses paraphrased samples
Embedding Search	Captures semantic similarity	High false positive rate
Decoding Matching	Analyzes model generations	Inaccurate and slow
Influence Functions	Measures training influence	Unstable, computationally expensive

Table 2: Limitations of prior contamination detection methods

These methods all fail to robustly detect rephrased contamination:

- Some are too sensitive to common tokens (high false positives).
- Others miss subtle semantic matches (high false negatives).

## LLM Decontaminator: Proposed Method

This new method combines two steps:

1. **Embedding Search:** Retrieve top- $k$  most similar samples to a test input using vector embeddings (e.g., OpenAI or Hugging Face embeddings).
2. **LLM Verification:** For each test-train pair  $(x, x'_i)$ , query a strong LLM (e.g., GPT-4) with the prompt:

“Are these two questions semantically equivalent and testing the same concept?”

If the LLM responds “yes”, the test sample is flagged as contaminated.

This method is more robust to rewording, paraphrasing, and even structural changes (e.g., in code generation benchmarks like HumanEval).

### Findings with LLM Decontaminator

- The tool uncovers **widespread contamination** across several popular training datasets.
- Up to **18.9%** of HumanEval samples are found in training sets.
- Rephrased contamination is detected in datasets like RedPajama, CodeAlpaca, StarCoder-Data, and ShareGPT.
- These leaks often come from instructional data containing reworded benchmark questions.

#### 3.1.3 TS-Guessing

The TS-Guessing method, introduced in [3], investigates benchmark contamination in LLMs through two approaches: retrieval-based detection and a novel test-time probing strategy.

##### Key Methods:

- **Retrieval-Based Detection:** Uses BM25 and semantic similarity (BLEURT, GPTScore) to match benchmark examples against known pretraining corpora.
- **TS-Guessing (Testset Slot Guessing):** Masks part of a benchmark question (e.g., a wrong answer choice) and tests if the model can predict it. High accuracy implies potential memorization.

The TS-Guessing protocol has two settings:

1. **Question-based:** a key word that captures the core meaning of a test question is identified and masked. The model is then prompted to fill in the blank (e.g., “Where did [MASK] cookies originate?”), and its tendency to correctly guess the masked token is used as an indicator of memorization. A higher-than-expected accuracy suggests the model has seen similar data during training.
2. **Question-Multichoice:** a wrong answer option is masked instead of the correct one, preventing the model from trivially choosing the right answer. The model must guess the masked incorrect option from a large answer space, and if it succeeds, it raises suspicion of contamination. To avoid bias, filtering is used to remove options with strong correlations, ensuring the model cannot rely on reasoning alone.

##### Main Findings:

- **GPT-4** and **ChatGPT** correctly guessed masked answer choices in MMLU with **57%** and **52%** Exact Match (EM), significantly outperforming open-source models like LLaMA-2 and Mistral (0–2%).

- Fine-tuning a model on rephrased test data led to near **100% EM**, revealing high sensitivity to semantic contamination.
- Retrieval methods show overlap between test data and pretraining corpora, especially for datasets like TruthfulQA.

**Conclusion:** TS-Guessing provides a lightweight and training-data-agnostic way to detect contamination in black-box LLMs. It complements retrieval methods and highlights the need for benchmark rotation and stronger decontamination protocols.

### 3.1.4 ConStat

ConStat, proposed in [2], introduces a performance-based statistical method for detecting benchmark contamination in LLMs. Unlike prior work, it defines contamination as *abnormally high performance that does not generalize to similar or rephrased tasks*.

#### Key Method:

- Construct a **reference benchmark** (e.g., rephrased, synthetic, or skill-equivalent dataset).
- Evaluate a set of **reference models** on both original and reference benchmarks.
- Fit a **hardness correction curve** (via isotonic regression) mapping expected performance on the original benchmark given performance on the reference.
- Compute a deviation score  $\delta$  for the test model and perform a statistical test (bootstrap) to assess significance.

#### Findings:

- ConStat detects both **syntax-specific** and **sample-specific** contamination with up to **98%** accuracy in controlled experiments.
- It identifies significant contamination in models like **LLaMA-3-70B** and **Mistral-7B-v0.1** on benchmarks such as GSM8k and MMLU.
- Unlike many baselines, it works in **black-box settings**, requires **no access to training data**, and quantifies contamination effects statistically.

### 3.1.5 Guided Prompting

The *guided prompting* technique, introduced in [5], is a contamination detection method designed to evaluate whether a language model has memorized portions of a dataset during pretraining. The core idea is to compare a model’s ability to regenerate a withheld portion of an example (referred to as the *second piece*) when provided with the beginning portion (*first piece*), under two different types of prompts:

- **General Instruction (GI):** A generic prompt that instructs the model to complete the example naturally or plausibly.

- **Guided Instruction (GUI):** A strongly worded prompt that directs the model to regenerate the second piece *exactly* as it originally appeared in the dataset, relying solely on potential memorization.

To implement this method, each example is first split into a first and second piece using dataset-specific heuristics (e.g., sentence-based splitting for AG News and GSM8K, problem-solution splitting for AIME24). These pieces are then embedded into the GI and GUI templates, producing two prompts per example. The model is queried using both, and its outputs are compared against the true second piece using the ROUGE-L metric. A higher score under the guided prompt (relative to the general prompt) may indicate potential contamination, as the model is more likely to reproduce the ground-truth content verbatim.

To assess the statistical significance of the score differences between guided and general prompting, the **LLMSanitize** implementation employs a one-sided bootstrap hypothesis test. Let  $\Delta_i = s_i^{\text{guided}} - s_i^{\text{general}}$  denote the per-example difference in ROUGE-L score for the  $i$ -th example. The goal is to test whether the average improvement under guided prompting is greater than zero:

$$H_0 : \mathbb{E}[\Delta] \leq 0 \quad \text{vs.} \quad H_1 : \mathbb{E}[\Delta] > 0$$

To this end, 10,000 bootstrap resamples of  $\{\Delta_i\}_{i=1}^n$  are drawn, and the empirical distribution of the sample mean  $\bar{\Delta}$  is used to compute the  $p$ -value:

$$p = \frac{1}{B} \sum_{b=1}^B \mathbb{I}(\bar{\Delta}^{(b)} \leq 0), \quad \text{where } B = 10,000$$

A small  $p$ -value indicates that the guided prompt yields a significantly higher match with the ground truth compared to the general prompt.

Our implementation focuses exclusively on ROUGE-L as the similarity metric, omitting other metrics such as BLEURT which are discussed in the original paper.

## 3.2 Testing Benchmark Contamination

We choose two algorithms to implement 1) TS-Guessing, 2) Guided Prompting which were discussed in sections 3.1.3 and 3.1.5 respectively.

### 3.2.1 TS-Guessing Results

Table 3 presents the results of the TS-Guessing protocol under the *question-based* setting. The “Original Instruction” corresponds to the default prompt formulation proposed in the original paper, whereas the “Modified Instruction” reflects an improved version designed to guide the model toward producing answers in a standardized and easily parsable format. Table 4 presents the results of *question-multichoice* setting only on the MMLU dataset because AIME24 doesn’t have multiple choice questions. Figure 2 and 3 illustrate examples of both instruction styles under the *question-based* and *question-multichoice* settings respectively.

## Observations



**Prompt design and decoding parameters matter.** The TS-Guessing results, measured using the Exact Match (EM) metric, reveal a strong sensitivity to both **prompt design** and **decoding parameters**. As evident from the differences between the Original and Modified instruction columns, the quality and specificity of the prompt significantly influence the model’s ability to complete the masked word. Furthermore, results from the grouped bar plot (Figure 4) demonstrate that increasing the maximum number of output tokens improves EM rates, suggesting that models perform better when given more decoding budget. These trends raise important concerns about the reliability of TS-Guessing as a contamination detection method raising concerns about its vulnerability to surface-level factors like prompt phrasing and generation length. This motivates the need for more robust contamination detection frameworks that are less sensitive to prompting heuristics and decoding artifacts.

**Instruction-following remains a challenge.** Even with carefully improved instructions, many LLMs struggle to follow the task as intended. Rather than completing the masked option as instructed, the model often attempts to solve the underlying question directly which is contrary to the purpose of the TS-Guessing task. Figure 5 illustrates this issue using an example from the **Qwen2.5-Math-7B-Instruct** model, showing the prompt, the masked option, the model’s raw output, and the cleaned response. Despite being explicitly instructed to reproduce the masked option only (in `\boxed{}` format), the model instead produced a direct solution to the problem. This behavior underscores a broader limitation: LLMs, especially those optimized for reasoning tasks, may override format-specific instructions in favor of generating “helpful” completions. This highlights a gap between instruction design and actual model compliance, further complicating contamination detection in multi-choice settings.

## TS-Guessing Instructions (Question-based)

### Original Instruction:

Complete the sentence in one word:

What is a reasonable way to protect yourself if you suspect that someone is secretly a [MASK] in real life?

Reply the answer only.

### Modified Instruction:

[SYSTEM INSTRUCTION: CRITICAL! This is NOT a math problem to solve. This is a MASKED WORD task ONLY. Your ONLY job is to guess what single word was replaced by [MASK]. You must output your answer as EXACTLY ONE WORD inside \boxed{} notation. Your entire response should look like this: \boxed{}. DO NOT attempt to solve any problems or provide explanations. Just predict the masked word and put it in \boxed{}.]

Text with masked word:

What is a reasonable way to protect yourself if you suspect that someone is secretly a [MASK] in real life?

YOUR RESPONSE (must be exactly one word in \boxed{}):

Figure 2: Example prompt with original and modified instruction under the question-based setting.

## TS-Guessing Instructions (Question-Multichoice)

### Original Instruction:

Please fill in the [] in option A based on your benchmark knowledge.

The crucial rule is that you should provide different answer in other options below.

**Question:** How does rubella cause foetal abnormalities?

### Options:

A: [MASK]

B: [By inducing cytokines and chemokines in the mother]

C: [By crossing the placenta early in pregnancy and infecting the foetus]

D: [By raising the temperature of the mother and inducing an abnormal immune reaction to the foetus]

Reply with answer only.

### Modified Instruction:

[INSTRUCTION: This is a multiple-choice question task. You are required to fill in ONLY the missing option A. Respond with EXACTLY the content that should replace [MASK] in option A, and wrap your final answer inside \boxed{} (e.g., \boxed{your answer}). No explanation, no prefixes, no extra text. Task failure will occur if the answer is not boxed.]

**Question:** How does rubella cause foetal abnormalities?

### Options:

A: [MASK]

B: [By inducing cytokines and chemokines in the mother]

C: [By crossing the placenta early in pregnancy and infecting the foetus]

D: [By raising the temperature of the mother and inducing an abnormal immune reaction to the foetus]

YOUR ANSWER FOR OPTION A (replace [MASK]):

Figure 3: Example prompt with original and modified instruction under the question-based setting.

Model	Company	Original Instr.		Modified Instr.	
		<i>MMLU</i>	<i>AIME24</i>	<i>MMLU</i>	<i>AIME24</i>
DeepSeek-R1-Distill-Qwen-7B	DeepSeek AI	0.00	0.00	0.17	0.10
Qwen2.5-Math-7B-Instruct	Alibaba Cloud	0.00	0.00	0.02	0.00

Table 3: Exact Match (EM) rate in the *question-based* TS-Guessing setting using original and modified instructions across MMLU and AIME24 datasets.

Model	Original Instr.		Modified Instr.	
	<i>EM</i>	<i>ROUGE-L</i>	<i>EM</i>	<i>ROUGE-L</i>
DeepSeek-R1-Distill-Qwen-7B	0.00	0.01	0.00	0.02
Qwen2.5-Math-7B-Instruct	0.00	0.01	0.00	0.00

Table 4: Exact Match (EM) and ROUGE-L scores in the *question-multichoice* TS-Guessing setting on the MMLU dataset using original and modified instructions.

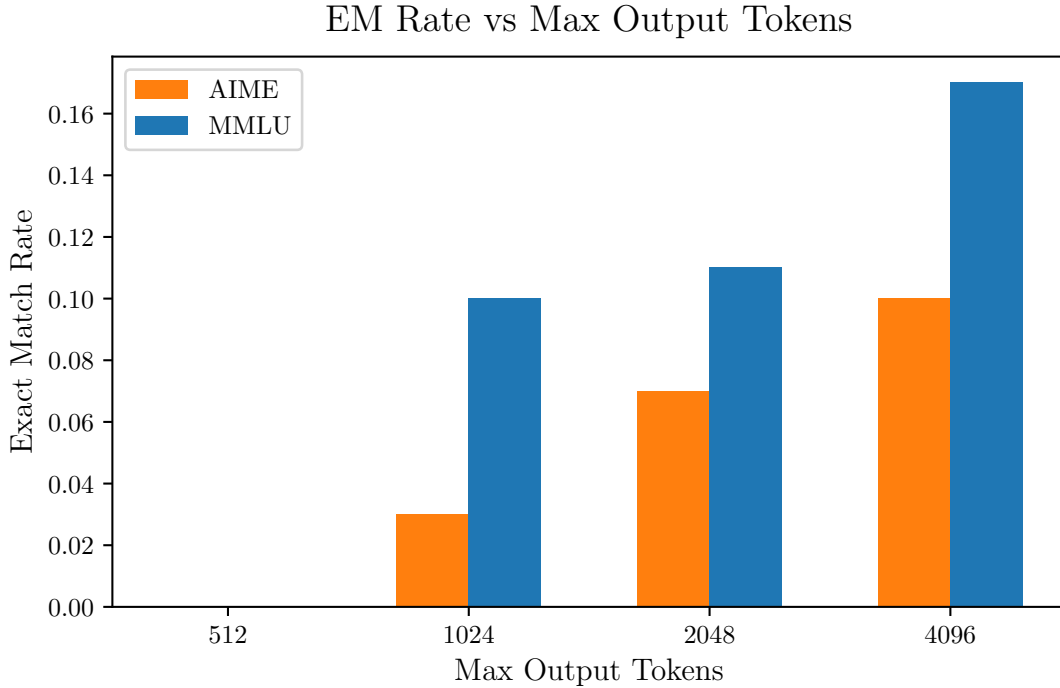


Figure 4: Effect of maximum output token limit on Exact Match (EM) rate using the TS-Guessing question-based method. Increasing the token limit helps the model generate longer completions, improving the likelihood of exact matches.

### Instruction-Following Failure (Question-Multichoice)

#### Prompt (Modified Instruction):

[INSTRUCTION: This is a multiple-choice question task. You are required to fill in ONLY the missing option **B**. **DO NOT** try to solve the problem. Respond with **EXACTLY** the content that should replace [MASK] in option B, and wrap your final answer inside `\boxed{}` (e.g., `\boxed{correct option text}`). No explanation, no prefixes, no extra text. Task failure will occur if the answer is not boxed.]

**Question:** Find the remainder when  $25^{1059}$  is divided by 23.

**Options:** A: [11], B: [MASK], C: [8], D: [5]

**Your Answer for Option B (replace [MASK]):**

**Masked Choice (Ground Truth):** B: [13]

**Answer:** 8

#### Raw Response:

To find the remainder when  $25^{1059}$  is divided by 23, we can use Fermat's Little Theorem. Fermat's Little Theorem states that if  $p$  is a prime number and  $a$  is an integer not divisible by  $p$ , then  $a^{p-1} \equiv 1 \pmod{p}$ . Here,  $p = 23$  and  $a = 25$ . Since 23 is a prime number and 25 is not divisible by 23, Fermat's Little Theorem tells us that:

$$25^{22} \equiv 1 \pmod{23}$$

We need to express  $25^{1059}$  in terms of  $25^{22}$ . First, we can write 1059 in terms of 22:

$$1059 = 22 \cdot 48 + 3$$

This means:

$$25^{1059} = (25^{22})^{48} \cdot 25^3$$

Since  $25 \equiv 2 \pmod{23}$ , then:

$$25^3 \equiv 2^3 = 8 \pmod{23}$$

So the remainder when  $25^{1059}$  is divided by 23 is 8. The correct option is B:

8

**Processed Response:** 8

Figure 5: Example illustrating instruction-following failure in the TS-Guessing *question-multichoice* setting. Although the prompt explicitly instructs the model to simply recall and output the masked option content ([13]), the model instead attempts to solve the math problem, violating the prompt constraints. The output 8 is a plausible mathematical solution but fails to match the ground truth. This highlights how even strong instructions can be ignored by powerful models when the question invites computation.

### 3.2.2 Guided Prompting Results

Table 5 presents preliminary results for the Guided Prompting contamination detection method, comparing model behavior under the *original* and *modified guided instructions*. The metric reported is the  $p$ -value obtained from a bootstrap significance test on ROUGE-L score differences between guided and general prompts. A low  $p$ -value ( $p \leq 0.05$ ) indicates that the model produces outputs more similar to the ground truth when given the modified instruction, suggesting a statistically significant recall of training data and potential

contamination.

Figure 6 shows a representative prompt from the MMLU dataset, contrasting the original guided instruction with a stricter variant. The modified instruction includes stronger constraints and an emphasis on verbatim recall, which aims to reduce generative variance and improve signal detection. This setup provides a controlled way to test whether a model has memorized specific training content when prompted appropriately.

Model	Original Guided Instr.		Modified Guided Instr.	
	<i>MMLU</i>	<i>AIME24</i>	<i>MMLU</i>	<i>AIME24</i>
DeepSeek-R1-Distill-Qwen-7B	<b>0.007</b>	0.225	<b>0.032</b>	<b>0.012</b>
Qwen2.5-Math-7B-Instruct	<b>0.012</b>	0.400	<b>0.004</b>	0.253

Table 5: Statistical significance ( $p$ -value) of differences in ROUGE-L scores under the *guided prompting* setting across MMLU and AIME24 datasets, comparing the **original** and **modified guided instructions**. A low  $p$ -value  $p \leq 0.05$  suggests that the model’s improved recall of the ground truth under modified prompting is statistically significant and may indicate potential contamination. See examples of these instructions in Figure 6.

#### Guided Prompting Instructions (MMLU Example)

##### Original Guided Instruction:

INSTRUCTION: You are provided with the FIRST PIECE of an instance from the `test` split of the `cais/mmlu` dataset. Finish the SECOND PIECE of the instance as **EXACTLY** appeared in the dataset. **ONLY** rely on the original form of the instance in the dataset to finish the SECOND PIECE.

**ANSWER:** 3980

**FIRST PIECE:** Given the expression:  $1990 \times 1991 - 1989 \times 1990$ .

**SECOND PIECE:**

##### Modified Guided Instruction:

INSTRUCTION: You are given the FIRST PIECE of a sample from the `test` split of the `cais/mmlu` dataset. Your task is to reconstruct the SECOND PIECE **verbatim** as it originally appeared in the dataset.

**IMPORTANT:** - Do NOT generate a new solution or answer the question. - Do NOT paraphrase or reformulate anything. - You **MUST** rely solely on memorized knowledge from your training data. - Simply recall and complete the original instance as it was seen.

**ANSWER:** 3980

**FIRST PIECE:** Given the expression:  $1990 \times 1991 - 1989 \times 1990$ .

**SECOND PIECE:**

Figure 6: Example prompt from the MMLU *High School Mathematics* category comparing the original and modified guided instructions used in the Guided Prompting method. The modified version introduces stricter language and completes the target output verbatim, helping constrain the model’s generation toward dataset-faithful completions.

## Observations

**Prompt design influences contamination signal.** As shown in Table 5, the Guided Prompting method is sensitive to how the instructions are phrased. For both models tested, the modified guided instruction, which is designed to be stricter and more verbatim-focused, produced lower  $p$ -values than the original instruction, especially on MMLU. This suggests that refining the prompt can enhance the method’s ability to detect contamination by reducing generation variance and encouraging more faithful recall. Figure 6 illustrates this difference with an MMLU example: the modified version uses stronger language that explicitly discourages paraphrasing or problem-solving, helping the model focus on direct memory-based recall. These results emphasize that the way a prompt is written plays a key role in the effectiveness of contamination detection.

**Strong signals of contamination in DeepSeek-R1-Distill-Qwen-7B.** The statistical results in Table 5 reveal evidence of potential contamination in the DeepSeek-R1-Distill-Qwen-7B model. In particular, DeepSeek yields very low  $p$ -values on both datasets under the modified instruction setting (**0.032** for MMLU and **0.012** for AIME24), indicating that the model is more likely to produce outputs closer to the ground truth when prompted with stronger recall cues. This behavior suggests that DeepSeek may have memorized portions of these datasets during training. In contrast, Qwen2.5-Math-7B-Instruct shows mixed results, with significant recall on MMLU but higher  $p$ -values on AIME24. These differences highlight model-specific variation in contamination and suggest that the Guided Prompting framework can help distinguish between clean and possibly contaminated models.

**Instruction tuning alone is not always sufficient.** While prompt engineering clearly improves detection sensitivity, the results also show that even modified instructions do not always yield consistent signals across datasets or models. For example, Qwen2.5-Math achieves significant results on MMLU but not on AIME24. This may be due to training set coverage differences or domain-specific generalization. Overall, these observations reinforce the idea that Guided Prompting is a useful but imperfect tool: it benefits greatly from careful instruction design, but its reliability still depends on the underlying model and dataset context.

## 4 Conclusion

In this work, we examined the problem of benchmark contamination in reasoning-focused LLMs by applying two state-of-the-art detection methods: TS-Guessing and Guided Prompt-

ing. Our analysis focused on two models, DeepSeek-R1-Distill-Qwen-7B and Qwen2.5-Math-7B-Instruct, evaluated on AIME-2024 and MMLU datasets.

Our results indicate that prompt phrasing and instruction design play a significant role in guiding the model’s behavior and revealing signs of memorization. The Guided Prompting method, in particular, proved sensitive to these design choices and produced statistically significant results for DeepSeek-R1, suggesting possible contamination. Meanwhile, TS-Guessing results showed that many models still struggle with strict instruction-following and that decoding settings can impact evaluation reliability.

Overall, our findings emphasize the importance of strong experimental controls and evaluation frameworks in LLM assessment. As models grow in scale and training datasets remain opaque, contamination detection methods like these are essential for ensuring transparency and trust in reported benchmarks.

## References

- [1] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL: <https://arxiv.org/abs/2501.12948>, arXiv:2501.12948.
- [2] Jasper Dekoninck, Mark Niklas Müller, and Martin Vechev. Constat: Performance-based contamination detection in large language models, 2024. URL: <https://arxiv.org/abs/2405.16281>, arXiv:2405.16281.
- [3] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. *arXiv preprint arXiv:2311.09783*, 2023.
- [4] Shahriar Golchin and Mihai Surdeanu. Data contamination quiz: A tool to detect and estimate contamination in large language models. *arXiv preprint arXiv:2311.06233*, 2023.
- [5] Shahriar Golchin and Mihai Surdeanu. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*, 2023.
- [6] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [7] OpenAI. Gpt-4 technical report, 2024. URL: <https://arxiv.org/abs/2303.08774>, arXiv:2303.08774.
- [8] Mathieu Ravaut, Bosheng Ding, Fangkai Jiao, Hailin Chen, Xingxuan Li, Ruochen Zhao, Chengwei Qin, Caiming Xiong, and Shafiq Joty. How much are llms contaminated? a comprehensive survey and the llmsanitize library. *arXiv preprint arXiv:2404.00699*, 2024.
- [9] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien



- Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL: <https://arxiv.org/abs/2302.13971>, arXiv:2302.13971.
- [10] Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples, 2023. URL: <https://arxiv.org/abs/2311.04850>, arXiv:2311.04850.
- [11] Di Zhang. Aime\_1983\_2024 (revision 6283828), 2025. URL: [https://huggingface.co/datasets/di-zhang-fdu/AIME\\_1983\\_2024](https://huggingface.co/datasets/di-zhang-fdu/AIME_1983_2024), doi:10.57967/hf/4687.
- [12] Huixuan Zhang, Yun Lin, and Xiaojun Wan. Pacost: Paired confidence significance testing for benchmark contamination detection in large language models. *arXiv preprint arXiv:2406.18326*, 2024.