Survey of Benchmark Contamination Detection Techniques in Large Language Models

Luke Dagnillo, Ashley Tittelbaugh

I. Introduction

Benchmark data contamination occurs when a Large Language Model (LLM) is evaluated on a benchmark that it has seen—at least in part—during its training phase. As LLMs continue to grow in both the number of parameters and the volume of training data, benchmark contamination becomes increasingly prevalent, impactful, and difficult to detect [?], [?]. This issue is further exacerbated when the contamination occurs through subtle paraphrasing, translations, or synthetic reformulations of benchmark data [?], [?]. Such overlap can lead to artificially inflated benchmark scores, which in turn may mislead users and researchers, skewing perceptions of model capabilities and influencing future research directions [?], [?].

To address this challenge, researchers have proposed a wide range of benchmark contamination detection algorithms. However, no current method offers a perfect solution. This paper surveys and categorizes the existing detection approaches and highlights their respective trade-offs. In addition, we implement and compare two representative algorithms, TS-Guessing [?] and Local Order Quiz [?], on distilled reasoning models. Drawing from both experimental results and an extensive literature review, we comment on the strengths, limitations, and open challenges in the field of benchmark contamination detection.

II. Categories of Detection Algorithms

Due to the wide variety of models and datasets, there is no one-size-fits-all solution to benchmark data contamination. This survey examined 20 distinct algorithmic approaches for detecting and evaluating dataset contamination, which were categorized into three broad classes: *Behavioral Probing*, *Similarity-Based Detection*, and *Statistical Signal Analysis*. Each of these categories reflects a different trade-off between access requirements, detection granularity, and precision. Behavioral probing methods infer contamination from a model's behavior when exposed to modified inputs [?], [?], [?]. Similarity-based approaches identify overlap with pretraining data using token-level or embedding-level similarity [?], [?]. Statistical signal analysis relies on internal model signals—such as confidence, entropy, or distributional shifts—to uncover signs of memorization [?], [?], [?].

1. Behavioral Probing

Behavioral probing detection algorithms aim to expose contamination by observing how a model behaves under altered or masked input conditions. The core intuition is that models are more likely to produce correct completions or predictions when exposed to familiar patterns—such as benchmark questions seen during training—even when essential input components are perturbed [?]. These methods typically require access only to model outputs, making them well-suited for use with proprietary or closed-source models where embeddings and training corpora are not publicly available [?], [?]. This flexibility allows behavioral probing to be adapted across a wide range of models and benchmark types.

However, many of these algorithms require human-designed or model-assisted perturbations for each individual test instance, making them more applicable to instance-level rather than large-scale benchmark-level detection [?]. Additionally, behavioral probing techniques may suffer from a higher false positive rate compared to other approaches, as they can conflate genuine reasoning ability with memorization. Their reliability is also sensitive to prompt design, semantic interpretation, and stochasticity in model outputs [?].

TABLE I
Behavioral Probing Detection Algorithms

Algorithm	Description
TS-Guessing [?]	Masks key parts of a question or answer and prompts the model to guess the missing information.
Instance Local Order Quiz [?]	Tests if a model can predict the position of a sample in a known sequence.
Canonical Order Testing [?]	Measures whether the model prefers canonical ordering over randomly shuffled benchmark instances.
Guided Instruction [?]	Embeds dataset names and instance context in prompts.
DCQ [?]	Presents one original instance and several distractors generated by GPT-4.
CAP (PCR-based detection only) [?]	Measures performance consistency ratio (PCR) under logically or semantically modified inputs.

2. Similarity-Based Detection

Similarity-based detection algorithms aim to identify contamination by measuring similarity, either lexical or semantic, between benchmark examples and a model's training corpus or generated outputs. The core intuition behind these approaches is that if a benchmark instance, whether exact or paraphrased, has appeared during training, the model's embeddings and outputs will exhibit greater similarity to that instance than would be expected by chance [?], [?]. This enables researchers to detect contamination not only through exact matches, but also through semantically similar content, capturing a broader notion of learned knowledge.

While this generalization introduces the possibility of false negatives, due to the inherently fuzzy nature of semantic similarity, it more accurately reflects what the model has learned [?]. These methods typically require some degree of access to the model, such as its embedding space or training data proxies, which distinguishes them from black-box approaches [?]. This additional access contributes to higher precision and accuracy compared to methods that rely solely on model outputs.

However, the retrieval and semantic evaluation stages in similarity-based methods are often computationally intensive [?]. Despite this, they are well-suited for large-scale audits, as they can be scaled to evaluate contamination at the benchmark level rather than being limited to individual instances.

TABLE II
SIMILARITY-BASED DETECTION ALGORITHMS

Algorithm	Description
LLM Decontaminator [?]	Retrieves top-k semantically similar training examples via embedding similarity.
Common Crawl Matching [?]	Uses search engine APIs and the Common Crawl index to find overlaps between benchmark instances and public web data.
Token Completion Overlap Score [?]	Prompts the model with partial inputs and compares the overlap between generated completions and reference answers to detect contamination.
Guided ICL with BLEURT [?]	Uses instructional prompts and BLEURT (or GPT-4) to measure semantic similarity between model outputs and benchmark examples.

3. Statistical Signal Analysis

Statistical Signal Analysis contamination detection techniques use internal model statistical patterns, such as log-probabilities, entropy, confidence scores, or distributional shifts, associated with benchmark inputs or outputs. These methods are based on the observation that if a model has encountered benchmark data during training, it will exhibit abnormal statistical behavior, such as lower entropy or higher token likelihoods, when processing those inputs compared to truly unseen examples [?], [?].

While these techniques often require white-box access to the model, such as logits or internal embeddings, they offer high precision in detecting subtle signals of memorization [?], [?]. In addition, their reliance on internal statistics rather than external corpora makes them computationally efficient and highly scalable for large-scale or benchmark-level detection [?], [?].

TABLE III
STATISTICAL SIGNAL ANALYSIS DETECTION ALGORITHMS

Algorithm	Description
PaCoST [?]	Uses paired-sample t-tests to compare model confidence on original versus rephrased instances.
LogProber [?]	Fits exponential decay curves to token-level log-probabilities.
(KDS) [?]	Measures shifts in kernel similarity matrices of model embeddings before and after fine-tuning.
CDD [?]	Compares entropy distributions of model outputs on original and perturbed inputs to detect memorization artifacts.
Min-K% Probability [?]	Evaluates the lowest k% token probabilities across the benchmark.
LNE Score [?]	Computes length-normalized entropy to identify unusually confident model outputs.

III. METHODOLOGY

We expand this survey by implementing and testing two benchmark contamination detection algorithms—TS-Guessing [?] and Local order Quiz [?]—on the AIME-2024 (American Invitational Mathematics Examination) [?] and MMLU (Massive Multitask Language Understanding) [?] high school mathematics datasets. These contamination tests were conducted using two open-source reasoning models: DeepSeek-R1-Distill-Qwen-14B [?] and Skywork-O2-Open-Llama-3.1-8B [?]. This application is relatively novel, as most contamination detection algorithms have predominantly been evaluated on casual models to date.

A. Models

1) DeepSeek-R1-Distill-Qwen-14B: DeepSeek-R1-Distill-Qwen-14B is a distilled version of the Qwen2.5-14B [?] model, fine-tuned using reasoning data generated by DeepSeek-R1 to enhance its performance in tasks requiring logical inference and problem-solving abilities. [?]

2) Skywork-O2-Open-Llama-3.1-8B: Skywork-o1-Open-Llama-3.1-8B is an 8-billion-parameter language model developed by Kunlun Inc. It adds reasoning to the Llama-3.1-8B architecture.

The development of Skywork-o1-Open-Llama-3.1-8B involved a three-stage training process:

- **Reflective Reasoning Training:** Generate high-quality, diverse data, followed by continuous pre-training and supervised fine-tuning. Dataloop

- **Reinforcement Learning for Reasoning Capabilities:** Enhance reasoning by capturing the influence of intermediate reasoning steps on final outcomes.
- **Reasoning Planning:** Search for optimal reasoning paths.

B. Benchmarks

1) AIME-2024: The AIME 2024 benchmark is based on the American Invitational Mathematics Examination. The exam consists of 30 questions, each requiring an integer answer between 000 and 999. This benchmark is widely used to assess the mathematical reasoning and problem-solving abilities of AI models. [?]

2) MMLU: The Massive Multitask Language Understanding (MMLU) benchmark is a comprehensive evaluation tool designed to assess the performance of large language models across a diverse array of subjects.

Within MMLU, the High School Mathematics category focuses on evaluating models' proficiency in topics typically covered at the secondary education level. [?]

C. Algorithims

This paper examines 2 behavior probing algorithms TS-Guessing [?] and Local Order Quiz [?].

1) TS-Guessing: Testset Slot Guessing (TS-Guessing) [?] is a behavioral algorithm that evaluates contamination based on the response of the model. It works by masking a carefully chosen part of the question (or answer set) and asking the model under test to predict the part of the questions that were masked. By ensuring the masked part is not something that can be easily intuited and has a wide variety of possible answers, TS-Guessing confirms that a correct answer by a model indicates a high probability that the model has seen this question before. TS-Guessing modifies the masking based on the format of the question.

- **Question Based:** Question based TS-Guessing masks a semantically pivotal portion of the question with many possibilities and asks the model to fill in the blank. This method is done for open ended (ie not multiple choice) questions.
  Since dataset AIME-2024 consists of open ended mathematical questions we employ this method by randomly masking one of the integers in the question. The numerical nature of the question lends this number masking method to the constraints imposed by the authors of TS-Guessing, importance of the masked attribute, and large number of possibilities, making it a good candidate for masking.
- **Question- Multichoice:** Question- Multichoice TS-Guessing masks an inncorrect answer of a multiple choice question. The stipuluation of incorrect ensures that a model cannot simply solve for the correct answer, reducing false positives.
  Since dataset MMLU consists of multiple choice questions we employ this method by randomly masking one of the incorrect answer choices.

2) Local Order Quiz: Local Order Quiz [?] is a prompt-based method for detecting contamination in language models by testing whether a model can remember the sequential order of examples from a given benchmark dataset. The prompt includes a dataset description, name, data split, and four options (chosen as a hyperparameter), so a random guess would achieve about 25% accuracy. This quiz picks a target question from the dataset, and the model is queried by choosing the next occurring example out of four potential options. The description used is typically one from the GitHub page or Hugging Face page of the dataset, with the rationale being that if the dataset is contaminated, so will the description.

D. Implementations

1) TS-guessing: As described in section III-C1, we implement the **Question based** method on the AIME dataset. This is done by randomly masking one integer in the problem statement with equal probability.

For the dataset MMLU we use the **Question multichoice** method. This is done by randomly selecting an *incorrect* option from the multiple-choice options and masking it.

In both cases, the models answer is then compared with the masked input and checked for exact match accuracy. Using exact matches with the author's descriptions of the algorithms and eliminates many false positives. We don't allow the model to reason as it could intuit a number that allowed the problem to be solved simply or neatly, allowing for a false positive result.

2) Local Order Quiz: The Local Order Quiz technique was used on the models in section III-A with the AIME and MMLU datasets. For each quiz we first selected a target question and its true next question. We then sampled three random questions from elsewhere in the dataset to action as distractor questions not adjacent to the initial target. These four options were randomly shuffled and labeled A-D for the model to choose from based on a constructed prompt. We then parsed the output using regex to extract the predicted letter and compare it to the correct answer.

IV. Results and Discussion

A. Model inference accuracies

To more holistically evaluate the contamination of these models we first run inference on both models with both datasets to establish a baseline accuracy. We prompted the model to generate an answer with a maximum of 4000 tokens, which was chosen due to our hardware limitations. An answer was generated for each question in the given datasets (100 for MMLU and 30 for AIME), and the generated response would be compared to the ground truth. This process would be repeated five times with the average accuracy and the standard deviation being reported. The results of this method are described in Tab. IV-A.

Model	AIME(%)	MMLU (%)
QWEN-14B	**22.67 ± 2.49**	**43.34 ± 5.59**
Skyworks-8B	**10.00 ± 3.65**	**39.60 ± 5.24**

TABLE IV
ACCURACY RESULTS FOR QWEN-14B AND SKYWORKS-8B ON AIME AND MMLU BENCHMARKS OVER 5 RUNS.

Our results differ from results reported by deepseek in its seminal paper [**?**]. There they report an MMLU accuracy of 69.7% on the same model for AIME-2024 We hypothesize that this difference is multifaceted, impacted by a difference in hyperparameters and the stringency of the evaluation techniques. The maximum allotted tokens for the output may not have been enough for the models (specifically the distilled Deepseek) to always reason their way to a correct answer for a given question. Despite these differences, since we evaluate using the same strategy on all models and datasets, we argue that the relative accuracy between our results in accurate and is still useful in providing a base for contamination detection.

B. TS-Guessing Results

1) AIME-2024: The percent contamination results expressed as percentage of exact matched with the masked portion of the question is shown in Tab V. The results

Model	AIME Contamination (%)	MMLU Contamination (%)
DeepSeek	**17.998 ± 4.47**	**2.22 ± 0.59**
Skyworks	**4.666 ± 1.83**	**1.33 ± 0.33**

TABLE V
AVERAGE CONTAMINATION (%) AND STANDARD DEVIATION OVER 5 RUNS TS-GUESSING.

show clear contamination of AIME on the deep seek model. However, we argue that these result, while small, indicate contamination on these benchmarks as with all the answer choices available to guess it is statistically unlikely that it'd be able to get any of them much less more than one. Furthermore, based on observations, we argue that this is a lower bound of contamination. This is because often times the model would output the correct answer to the problem asked or the letter of the masked problem, leaving us unable to evaluate that problem fully.

C. Local Order Quiz Results

The results in Tab. VI show both models perform slightly above random chance on MMLU, not suggesting much, if any, real contamination. With AIME however, both models show a moderately higher percentages than random chance, which may indicate mild or moderate levels of contamination. While these measurements are not definitive of contamination, they do suggest the potential of leakage or memorization.

Model	AIME Contamination (%)	MMLU Contamination (%)
DeepSeek	**33.33**	**27.00**
Skyworks	**36.67**	**26.00**

TABLE VI
CONTAMINATION (%) LOCAL ORDER QUIZ

V. CONCLUSION

Overall, detecting contamination is a complex and multifaceted topic that has a variety of approaches with different benefits and drawbacks. This survey identifies three broad categories of algorithms that encompass the tradeoffs between access requirements, detection granularity, and precision. This survey has identified three broad categories of algorithms. Behavior Probing determines contamination by using models outputs under masked or altered conditions. Similarity-Based detection mechanisms measure lexical and semantic similarity between benchmark examples and a models training corpus or outputs. Finally, Statistical Signal Analysis techniques use internal model statistical patterns associated with benchmark inputs or outputs to detect contamination. We then further explore Behavioral probing techniques by running both TS-Guessing [**?**] and Local Order Quiz [**?**] on reasoning models. These results allow us to detect statistically noticeable contamination in the AIME-2024 datasets on both the Skyworks and DeepSeek models. In addition our results for MMLU raise cause for further examination of contamination, even with the little contamination detected. This is surprising because AIME was the datasets the models preformed the worst on. Overall these results argue for a mixture of contamination detection algorithms that fully explore the different tradeoffs between access requirements, detection granularity, and precision. In addition, we find it isn't only the "well preforming" datasets that are likely to be contaminated, further underscoring the importance of testing and preventing data contamination in all datasets.

REFERENCES