Simulating Training Data Leakage in Multiple-Choice Benchmarks for LLM Evaluation

Naila Shafirni Hidayat[1] **Muhammad Dehan Al Kautsar**[2]
Alfan Farizki Wicaksono[1] **Fajri Koto**[2]
[1]Faculty of Computer Science, Universitas Indonesia
[2]Department of Natural Language Processing, MBZUAI
naila.shafirni@ui.ac.id, alfan@cs.ui.ac.id, {muhammad.dehan,fajri.koto}@mbzuai.ac.ae

Abstract

The performance of large language models (LLMs) continues to improve, as reflected in rising scores on standard benchmarks. However, the lack of transparency around training data raises concerns about potential overlap with evaluation sets and the fairness of reported results. Although prior work has proposed methods for detecting data leakage, these approaches primarily focus on identifying outliers and have not been evaluated under controlled simulated leakage conditions. In this work, we compare existing leakage detection techniques, namely `permutation` and `n-gram`-based methods, under a continual pretraining setup that simulates real-world leakage scenarios, and additionally explore a lightweight method we call `semi-half` question. We further introduce two efficient extensions, `permutation-R` and `permutation-Q`. While `semi-half` offers a low-cost alternative, our analysis shows that the `n-gram` method consistently achieves the highest `F1-score`, performing competitively with `permutation-Q`. We also refine these techniques to support instance-level detection and reduce computational overhead. Leveraging the best-performing method, we create cleaned versions of MMLU and HellaSwag, and re-evaluate several LLMs. Our findings present a practical path toward more reliable and transparent evaluations, and we recommend contamination checks as a standard practice before releasing benchmark results. [1]

1 Introduction

The development of Large Language Models (LLMs) has shown competitive performance on multiple-choice question answering (Brown et al., 2020; OpenAI et al., 2024; Qwen et al., 2025; Team et al., 2024a; Grattafiori et al., 2024). These models are evaluated on benchmark datasets designed to assess specific competencies such as knowledge and reasoning. However, many LLMs do not disclose their pre-training data (Piktus et al., 2023), raising concerns that benchmark evaluation sets were included in training.

This lack of transparency raises a critical question: *Do current evaluation results reflect the true generalization abilities, or are they barely a product of memorization?* Suppose a model has been trained on evaluation datasets during training. In that case, it doubts the fairness of comparison as its ability to answer questions might originate from data memorization (Carlini et al., 2023) rather than reasoning or generalization. This issue, referred to as data contamination (Magar and Schwartz, 2022; Balloccu et al., 2024) poses significant concerns for reliable benchmarking.

Recent studies have introduced various methods to detect data contamination in multiple-choice question (MCQ) benchmarks for LLMs (Ni et al., 2024; Xu et al., 2024; Li, 2023). However, these approaches primarily focus on identifying outliers and have not been systematically evaluated under controlled leakage conditions. Furthermore, there is limited understanding of their relative effectiveness (Hu et al., 2022; Samuel et al., 2024; Fu et al., 2025), and no consensus on the optimal configurations for detecting training data leakage. To address these gaps, we benchmark existing leakage detection methods under controlled simulations, focusing on two widely used MCQ datasets in LLM evaluation: the Massive Multitask Language Understanding (MMLU) dataset (Hendrycks et al., 2021a) and the HellaSwag dataset (Zellers et al., 2019).

Our key contributions are as follows:

- We compare three leakage detection methods under simulated training data leakage via continual pre-training: (1) the `semi-half` method, which tests whether a truncated ver-

[1]Code and dataset are available at `https://github.com/nailashfrni/mcq-leakage-detection-code`

sion of a question still results in the correct answer; (2) the `permutation` method, originally proposed by Ni et al. (2024), which evaluates whether the original option order yields the highest likelihood among all permutations; and (3) the `n-gram` method, which assesses the similarity between a generated option sentence and the original, following Xu et al. (2024).

- We improve the `permutation` method by introducing two variants, `permutation-R` and `permutation-Q`, which reduce computational overhead while improving `F1-score`. We also refine the `n-gram` method to support instance-level detection.

- We construct and release a subset of the MMLU and HellaSwag dataset verified to be free of contamination across several popular LLMs. Furthermore, we re-evaluate these models on the clean subset to observe shifts in performance ranking.

2 Related Work

2.1 Evaluation Benchmark of LLM

Language model evaluation has shifted from classical NLP tasks—such as named entity recognition and part-of-speech tagging—toward benchmarks that assess knowledge and reasoning, driven by advances in fluency and coherence. These evaluations commonly adopt a multiple-choice format, exemplified by MMLU (Hendrycks et al., 2021a), which compiles questions of 57 subjects from a wide range of school exams across different education levels. Other popular reasoning benchmarks include HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), Social–IQa (SIQA) (Sap et al., 2019), and TruthfulQA (Lin et al., 2021).

MMLU is one of the most widely used datasets for evaluating the knowledge capabilities of LLMs. To improve its quality and robustness, prior work has introduced several variants. MMLU-Pro (Wang et al., 2024) enhances the dataset by increasing question difficulty through filtering, expanding answer choices from four to ten, and incorporating expert review. Separately, Gema et al. (2025) released MMLU-Redux, a cleaned version that addresses issues such as ambiguous phrasing, multiple correct answers, and incorrect ground truths. However, despite these improvements, both variants primarily focus on question quality and coverage. Neither MMLU-Pro nor MMLU-Redux incorporates systematic filtering or analysis to detect overlap with pretraining data, leaving open the risk that benchmark scores may reflect memorization rather than true generalization.

2.2 Data Contamination Detection

Numerous methods have been proposed to detect data contamination in LLMs, broadly falling into logit-based, generation-based, and hybrid categories. Logit-based methods analyze the model's output probabilities or internal states; for example, Ni et al. (2024) compare log-probabilities across different option orders, while Li (2023) use perplexity to detect dataset-level leakage. However, these approaches primarily focus on outlier detection, offer limited support for instance-level analysis, and have not been evaluated under controlled training leakage simulations. In contrast, generation-based methods assess whether the model can reconstruct reference content when prompted. Golchin and Surdeanu (2024) use "time-travel" prompts incorporating dataset-specific cues to regenerate partial instances and compare them to the original text. Xu et al. (2024) introduce a hybrid approach combining n-gram similarity with perplexity, though their focus is on GSM8K (Cobbe et al., 2021) and MATH datasets (Hendrycks et al., 2021c). Importantly, these methods have not been tested on the multiple-choice question (MCQ) format, which remains the most widely used prominent structure in LLM evaluation benchmarks.

3 Leakage Detection Method

To detect whether a model has been exposed to a particular question, especially in multiple-choice question (MCQ) tasks, the problem can be formulated as a *leakage detection* task: given a model $\mathcal{M}$, a question q, options O, and contexts C, the goal is to predict whether $\mathcal{M}$ has memorized them, labeled as either *Leakage* (L) or *Not Leakage* (NL). Since no ground-truth labels exist for this task, we simulate training data leakage using continual pre-training and compare the effectiveness of three detection methods: `semi-half`, `permutation` (Ni et al., 2024), and `n-gram` (Xu et al., 2024). To improve efficiency, we introduce a simplified variant of `permutation`, called `permutation-R`, and

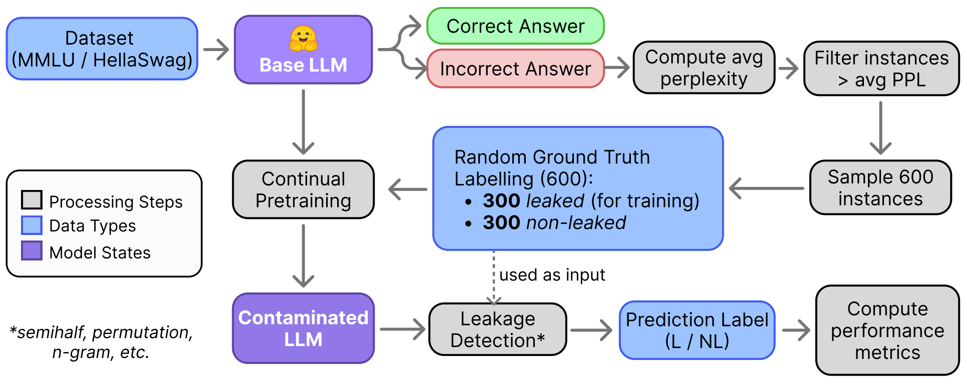

Dataset (MMLU / HellaSwag)
Base LLM
Correct Answer
Incorrect Answer
Compute avg perplexity
Filter instances > avg PPL
Processing Steps
Data Types
Model States
Continual Pretraining
Random Ground Truth Labelling (600):
• 300 leaked (for training)
• 300 non-leaked
Sample 600 instances
used as input
*semihalf, permutation, n-gram, etc.
Contaminated LLM
Leakage Detection*
Prediction Label (L / NL)
Compute performance metrics

Figure 1: Workflow for simulating data leakage and evaluating detection methods. Boxes represent different components: processing steps (gray), data types or datasets (blue), and model states (purple).

propose a new method, `permutation-Q`, built on the same foundation.

3.1 Leakage Simulation

Figure 1 illustrates our controlled simulation of intentional data leakage. We start by selecting questions from MMLU (Hendrycks et al., 2021b) and HellaSwag (Zellers et al., 2019) that the model initially answers incorrectly. From this set, we randomly sample 600 instances with above-average perplexity to ensure unfamiliarity and minimize the chance of prior exposure during pre-training. We use 300 of these samples for continual pre-training via Low-Rank Adaptation (LoRA) (Hu et al., 2021), simulating data leakage. After training, all detection methods are applied to the full set of 600 instances. The 300 examples included in pre-training are labeled as "Leaked", while the remaining 300 serve as "Not Leaked". We assess detection performance using `Precision`, `Recall`, and `F1-score`.

3.2 `Semi-half` Detection Method

To answer a multiple-choice question, a model relies on both the question and the options (Robinson et al., 2023). If it can still select the correct answer after the first half of the question is removed, this may suggest prior exposure during pre-training. Motivated by this, we propose a simple truncation-based method that retains only the final seven words of each question, providing minimal context while aligning with the autoregressive nature of decoder-based LLMs. The seven-word limit reflects the average half-length of the MMLU questions. Figure 2 illustrates a semi-half truncation example: if the model has seen the question during pre-training, it may still produce the correct answer despite the limited input; otherwise, the model is unlikely to produce the correct answer due to insufficient context.

`Semi-half` Truncation Example

Original Question: 'A plant grows in the opposite direction of the gravitational force. This is an example of'

A. positive thignotropism
B. negative phototropism
C. positive phototropism
D. negative gravitropism

`Semi-half` Question: 'gravitational force. This is an example of'

A. positive thignotropism
B. negative phototropism
C. positive phototropism
D. negative gravitropism

Figure 2: Semi-half truncation example

3.3 `Permutation` Method

Ni et al. (2024) proposed a method to detect contamination by evaluating how a language model assigns probabilities across different orders of multiple-choice answer options. The key idea is that if the model consistently assigns the highest probability to the original option order (e.g., A-B-C-D), it may have memorized that specific multiple-choice instance during training and indicated potential contamination.

The detailed method is explained in Appendix A. The algorithm complexity for this method is dominated by computing log-probability scores for each option order variation in a question. In big-Oh notation, the complexity is stated as $O(n!)$, where n denotes the number of options. This is considered a costly approach, and we modified this method to better achieve a less complex algorithm.

`Permutation-R`. Our main concern with the `permutation` method is its computational cost in computing the log-probability for all permutation variations. To address this, we eliminate permutations that have nearly similar log-probability distributions for all questions, then retain only a representative permutation subset.

To determine which permutation pairs show similar distributions, we employ Mean Absolute Difference (MAD) to measure the discrepancy in log-probability scores between two permutations. Let p_{ji} and p_{ki} represent the log-probability scores for permutations j and k on question i, respectively, and let z denote the number of questions. The mean absolute difference between permutations j and k, denoted by $\texttt{Diff}(j, k)$ is computed as:

$$\texttt{Diff}(j, k) = \frac{1}{z} \sum_{i=1}^{z} |p_{ji} - p_{ki}| \, .$$

We experiment with three different models: Qwen2.5-7B (Qwen et al., 2025), LLaMA-3.1-8B (Touvron et al., 2023), and Gemma-7B (Team et al., 2024b). For each experiment, we compute $\texttt{Diff}(j, k)$ for all possible permutation pair j and k and average the ranking across experiment. Since lower MAD indicates more similar distribution, we sort the average rank in increasing order. From that order, we retain only one permutation from each pair. To determine the optimal number of permutations used, we experiment with various proportion values p to observe which setting best balances computational cost and performance. The optimal p is then selected and used as the final configuration for the `permutation-R` method.

The algorithm complexity is $O\left(p.[n!]\right)$, where n denotes the number of options and p is for percentage of permutations used. This improvement might not be significant in the big-Oh notation since it still has permutation complexity. However, in practice, the reduced variation factor contributes to reducing computation time.

`Permutation-Q`. In practice, `permutation-R` improves efficiency by introducing a fractional term upfront. However, challenges arise when dealing with tasks that involve more than four answer choices, such as MMLU-Pro (Wang et al., 2024), with 10 options. To address this, we propose `permutation-Q` method, that replaces the factorial component with a more tractable quadratic approximation. The idea is to employ only two options in each log-probability calculation.

Suppose that we have an instance $x = [q, o_1, o_2, ..., o_n]$ where q denotes the question and o_n is the last option answer. We generate permutation P_2^n from $o = \{o_1, o_2, ..., o_n\}$ to only two options. We calculate the log-probability score for all possible permutations of two options. If the original option order (A-B) produces the maximum log-probability among all orders, we consider the instance x as 'Leakage', otherwise not.[2] The algorithm is presented in Algorithm 1 in Appendix B.

The complexity of the above method is centered on log-probability calculation for all possible combinations of options. The big-Oh notation is computed as:

$$O(P_2^n) = O(n.(n-1)) = O(n^2 - n) = O(n^2).$$

The complexity is reduced from factorial to quadratic, which is an improvement in detecting a leakage in a particular model.

3.4 `N-Gram` Method

The `n-gram` method builds on the approach introduced by Xu et al. (2024), which uses n-gram accuracy to detect potential data contamination during pre-training. The core idea is to test whether a model has memorized benchmark answer options by evaluating its ability to generate them. While the original method generates n tokens per prompt and compares them to a reference sequence, we modify it to generate an entire option sentence in a single inference. Other than that, while Xu et al. (2024) focus on detecting leakage at the dataset-split level by comparing metric differences between original and synthetic data, we adapt it to work at the instance level. This modification allows the method to identify contamination on a per-example basis and allows the analysis to be more comprehensive. The full details of the method and its algorithm are presented in Appendix C.

4 Experiment

4.1 Set-Up

We experiment with four models and two evaluation benchmarks—MMLU (Hendrycks et al., 2021b) and HellaSwag (Zellers et al., 2019)—to simulate data contamination in LLMs. The model list detailed in Table 5 in Appendix E. Each model

[2] The key idea is to compare the original 2-option pairs with its permutations, regardless of whether the correct answer is present in the pair.

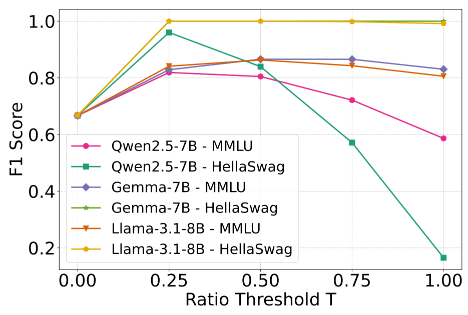


Figure 3: Comparison of `n-gram` detection F1-score performance under varying ratio thresholds T.

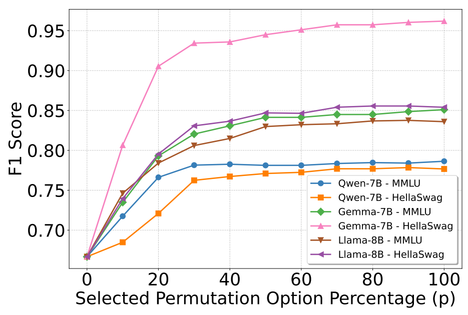


Figure 4: Performance of `permutation` at different percentages p, used to reduce computational complexity.

undergoes continual pre-training on each benchmark. Using the Adam optimizer, we set the learning rate to $1e-5$ for the language model head and $5e-4$ for other parameters. Each model is trained for 10 epochs with a weight decay of 0.01, a warmup ratio of 0.1, and a cosine learning rate scheduler. We also record the loss at each epoch for monitoring. For experiments involving LLaMA-3.1-8B base and Gemma-7B, we utilize an H100 SXM GPU with 80GB VRAM. For all other models, we use an A40 GPU with 48GB VRAM.

After completing the continual pre-training for all eight settings, we tune the threshold of `n-gram` and optimize the `permutation` method first. We then use this configuration to evaluate all methods and compare their performance.

4.2 Preliminary Results

Varying `N-Gram` Method's Threshold. We explore the effect of varying the threshold T in the `n-gram` method, which determines its sensitivity to determine an instance as 'Leakage'. The results of this comparison are presented in Figure 3. Across all experiments, a threshold of $T = 0.25$ consistently yields the best or comparable `F1-score`. Notably, in Qwen2.5-7B on HellaSwag, `F1-score` drops sharply as T increases. For LLaMA-8B and Gemma-7B on HellaSwag, `F1-score` remains at 100% for $T = 0.25$, 0.5, and 0.75, with only a slight decrease at $T = 1.0$. A similar trend appears in MMLU, where `F1-score` peaks at $T = 0.5$ but still exceeds 80% at $T = 0.25$.

These results suggest that $T = 0.25$ offers the best balance of sensitivity and reliability for detecting data contamination. This approach ensures that all potentially suspicious questions, even if only a single option is successfully replicated, are treated as 'Leakage'. This ensures comprehensive detection and allows us to capture as many contaminated instances as possible.

Reducing Permutation Variation. Using the Mean Absolute Difference (MAD), we compute the difference scores between log-probability distributions for each permutation pair. We rank all pairs by similarity for each model and average these rankings to identify the top 24 most similar pairs (see Table 3 in Appendix D). Notably, many of these differ by only two character swaps, suggesting such changes have minimal effect on the log-probabilities.

Based on this observation, we vary the proportion p to find an optimal trade-off between performance and efficiency. The full list of permutation used for each p is detailed in Table 4 in Appendix D. The impact of varying this percentage threshold on performance is illustrated in Figure 4.

An interesting finding is that using 50% or 100% of the permutations yields no significant difference in performance. The `F1-score` remains relatively stable across this range. This empirically supports the idea that using only a subset of permutations can still yield high performance, as some permutations may produce similar log-probabilities. To balance computational cost and detection quality, we adopt $p = 50\%$ as the default threshold for the `permutation-R` method in the subsequent comparison.

`Permutation-Q` Experiment. We experiment `permutation-Q` in six different model and dataset settings to observe its performance. We compare the result with `permutation-R` and the original `permutation`. The `F1-score` comparison is presented in Figure 5.

By using only two options per log-probability computation, `permutation-Q` achieves competitive `F1-score` scores and, in some settings, even

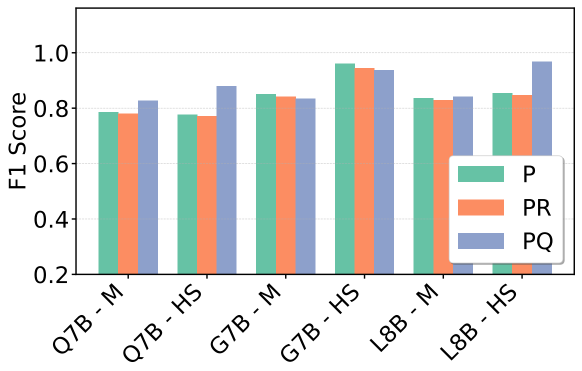


Figure 5: Performance of `permutation`-based methods: the original (P), reduced variant `permutation-R` (PR), and quick variant `permutation-Q` (PQ), evaluated on MMLU (M) and HellaSwag (HS). Model abbreviations: Q = Qwen, L = Llama, G = Gemma.

outperforms both the original `permutation` and `permutation-R` methods. These results highlight its ability to reduce complexity while potentially improving performance.

4.3 Main Results

Detection Performance Across Methods with Tuned Thresholds. Using the selected thresholds and configurations for the `n-gram` and `permutation-R` methods, along with other original approaches, we compare detection performance across eight evaluation settings. The results are presented in Table 1. Since each experiment uses a different subset of data, depending on the base model's initial ability to answer the questions, the metrics should only be compared across detection methods within the same experiment, not across different models or benchmarks.

Across experiments, the `n-gram` method consistently achieves over 81% `F1-score` and outperforms other methods in Qwen-0.5B (MMLU) and all HellaSwag settings. `Permutation-Q` shows strong performance as well, outperforming other methods in Qwen-7B and LLaMA-8B, while the original `permutation` achieves the best `F1-score` in Gemma-7B. Overall, `permutation-Q` performs competitively and often matches or exceeds the performance of `n-gram`.

Interestingly, combining multiple methods tends to increase `Recall` as more instances are flagged as 'Leakage', but this often leads to a decrease in `F1-score` (see Table 6 in Appendix F), likely due to a rise in false positives and lower `Precision`. A closer look at the HellaSwag results reveals that `n-gram` almost perfectly detects 'Leakage' across all settings. This may be attributed to the nature of the HellaSwag task, which involves predicting the most coherent continuation of a given context. This objective closely aligned with how `n-gram` generates options based on prefix patterns. It is also possible that `n-gram` benefits from the continual pretraining objective, which focuses on next-token prediction. Regardless of the cause, `n-gram` remains highly effective and competitive. Furthermore, since it does not require access to model logits, it can be applied to closed-weight models. For these reasons, we adopt `n-gram` as the primary detection method for both MMLU and HellaSwag.

Leakage Detection Results on Full Benchmarks. After applying the `n-gram` method to the full MMLU and HellaSwag datasets, we identified several instances flagged as 'Leakage'. Figure 6 illustrates the proportion of detected leakage across different models. Qwen2.5-7B shows a relatively higher tendency for potential leakage in both benchmarks, followed by LLaMA-3.1-8B on MMLU and Qwen2.5-0.5B on HellaSwag. These observations align with the findings of Ni et al. (2024), who also highlighted potential risks of leakage in the Qwen model family, despite using a different methodology. We additionally tested DeepSeek (Liu et al., 2024) and Gemini (Team et al., 2023) models: DeepSeek ranks third with 35% on MMLU and 0.17% on HellaSwag, while Gemini-2.0-Flash exhibits minimal indications of leakage across both datasets.

We observe that MMLU shows a higher potential for leakage across models compared to HellaSwag, likely due to its widespread use in NLP research, making its content more likely to appear in training data. Additionally, the `n-gram` method is sensitive to the length of the option text, as it generates tokens sequentially to match the reference, resulting in slower detection for longer options. In contrast, `semi-half` and `permutation`-based methods require only a single inference step per instance, making them more efficient.

Leakage $\neq$ Model Understanding. We observe that models do not always correctly answer leaked instances, both in our leakage simulation and full-benchmark evaluations. As illustrated in Table 2, models frequently fail to provide correct responses to flagged examples, indicating that memorizing input sequences does not equate to genuine understanding or reasoning. Nonetheless, instances

Benchmark	Model	S	P	PR	PQ	N	S+PQ	S+N	PQ+N
MMLU	Qwen-0.5B	55.68	82.78	82.12	86.63	<u>88.23</u>	79.16	79.84	85.47
	Qwen-7B	56.88	78.64	78.12	<u>82.68</u>	81.89	78.79	78.37	80.59
	Gemma-7B	68.59	<u>85.10</u>	84.14	83.45	82.87	77.12	75.95	80.32
	LLaMA-8B	50.51	83.59	82.98	<u>84.27</u>	84.11	80.00	79.84	81.63
HellaSwag	Qwen-0.5B	60.86	81.13	80.73	94.94	<u>99.83</u>	80.61	82.99	96.46
	Qwen-7B	67.92	77.67	77.10	87.93	<u>96.01</u>	73.48	75.79	95.67
	Gemma-7B	71.04	96.20	94.50	93.69	<u>100.00</u>	74.91	75.76	94.19
	LLaMA-8B	68.71	85.40	84.70	96.77	<u>100.00</u>	75.09	75.66	96.77

Table 1: Detection performance (`F1-score`) for various methods across different models and benchmarks. The methods are coded as follows: S = `Semi-half`, P = `Permutation`, PR = `Permutation-R`, PQ = `Permutation-Q`, and N = `N-Gram`. For combined methods (denoted by '+'), an instance is classified as Leakage if at least one of the methods detects it. <u>Underlined</u> scores represent the best method among the model & benchmark combinations.

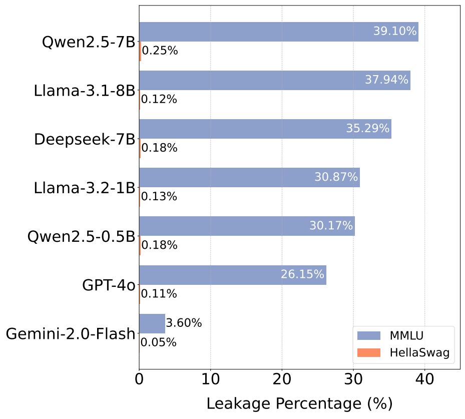


Figure 6: Data leakage rates for each model on MMLU and HellaSwag using the `n-gram` method.

Model	Accuracy on Leaked Set
Deepseek-7B	39.06%
Gemini-2.0-Flash	95.65%
GPT-4o	87.09%
LLaMA-3.1-8B	51.19%
Qwen2.5-7B	64.12%

Table 2: Accuracy scores of models on MMLU instances detected as leakage by `n-gram`.

encountered during pretraining are generally more likely to be answered correctly. This discrepancy may arise from a misalignment between training and evaluation objectives: while continual pretraining does not aim to identify the most likely answer among multiple choices, evaluation typically depends on comparing the log-likelihoods of each option.

Despite this disconnect, computing the log-likelihood of each option (A–D) remains the standard for evaluating multiple-choice questions in LLM. However, since we lack visibility into how each model was exposed to these benchmarks—such as whether answer keys were included during pretraining—we propose two complementary definitions of leakage: (1) Strong leakage: the detection method identifies the instance as 'Leakage' and the model answers it correctly. (2) Weak leakage: the detection method identifies the instance as 'Leakage', regardless of the model's answer.

Benchmark Reduction Under Strong Leakage Definition. By the strong definition of leakage, we remove any correctly answered instance flagged as 'Leakage' by the `n-gram` method in at least one LLM. This results in the removal of 6,547 out of 14,042 instances (46.6%) from MMLU, and 38 out of 10,042 instances (0.38%) from HellaSwag.

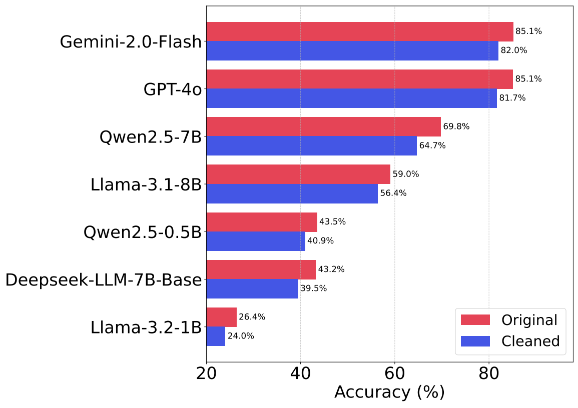


Figure 7: Comparison of model performance on original vs. cleaned MMLU benchmark based on strong definition of leakage.

Figure 7 compares model performance on the original and cleaned versions of MMLU.[3] On

[3]HellaSwag is excluded due to the minimal number of leaked instances (0.38%) and the absence of notable performance differences. GPT-4o remains the top-performing model on both versions.

the cleaned MMLU benchmark, Gemini-2.0-Flash achieves the highest performance among all evaluated models, followed closely by GPT-4o. While the relative ranking across the evaluated models remains largely consistent even after removing 46% of potentially leaked instances, we note that the models differ in size. Therefore, performance shifts could be more pronounced when comparing models within the same parameter scale. To explore this further, we analyze accuracy drops by subject and subject category within the same model in Section 5.

We also tested the LLMs' performance with the weak leakage definition. Since model accuracy on leaked instances is not 100%, removing these instances reduces the number of incorrectly answered examples, resulting in higher accuracy for some models. However, this only affects the performance ranking on the MMLU dataset, where GPT-4o slightly surpasses Gemini 2.0 Flash to become the top-performing model. The ranking positions for the other models remain unchanged.

5 Analysis

5.1 Performance Varies in Specific Subjects

Referring to the strong version of leakage definition, we analyze performance changes across specific MMLU subjects. Figure 8 presents the percentage drop in accuracy after removing potentially leaked instances, which highlights subjects with relatively large performance declines. In particular, model performance on the *Anatomy* subject drops substantially, with Qwen-7B showing the largest decrease (35.4%).

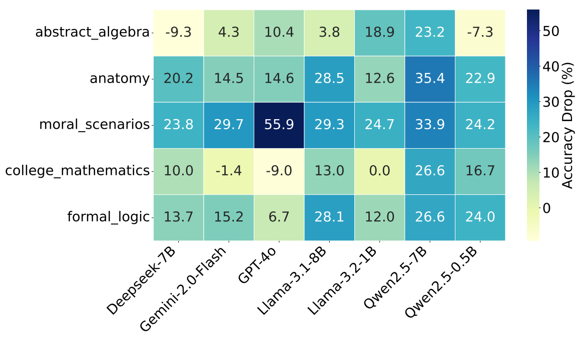


Figure 8: Performance drops in selected MMLU subjects for each model. Qwen-7B shows the largest accuracy drop in these subjects.

Meanwhile, *Moral Scenarios* contains the highest number of detected leaked instances. However, as observed in previous experiments, this subject yields a low `F1-score`. Upon further inspection, we find this is likely due to repetitive option templates used across all questions, such as "*Not Wrong, Not Wrong; Not Wrong, Wrong; Wrong, Not Wrong; Wrong, Wrong*" or "*True, False*." These patterns may increase the chance that the model generates the correct option based solely on surface similarity, leading to false positives under the `n-gram` detection method.

We also observe a noticeable accuracy decline in *Formal Logic* after data cleaning. This suggests that part of the model's original strong performance in this subject could be attributed to memorization rather than genuine reasoning ability. The cleaned results provide a more realistic reflection of the models' logical reasoning skills. We also find that the STEM group is most affected, showing the largest performance difference across subject groups (see Figure 10 in Appendix J for further details).

5.2 Detection Methods: Pros & Cons

As shown in Table 1, the `n-gram` method consistently achieves the highest `F1-score`, followed closely by `Permutation-Q`. However, each method has its own strengths and weaknesses. Table 8 in Appendix H summarizes the strengths and limitations of each method that focusing on computational cost and leakage detection effectiveness, to inform their use in different scenarios.

6 Conclusion

In this work, we simulate leakage using three methods—`semi-half`, `permutation`, and `n-gram`—and introduce a simplified variant, `permutation-Q`, which uses only two options and achieves strong performance across several settings. Our results identify `permutation-Q` and `n-gram` as the most effective detection methods under our controlled simulation setup, with Qwen-7B showing a high risk of leakage, especially in MMLU STEM subjects, where accuracy drops by up to 8% after filtering. We also observe consistent accuracy reductions across all models once leaked instances are removed. These findings highlight the distinction between memorization and true understanding, reinforcing the need to apply leakage detection before evaluation to ensure that test data remains clean and that the reported results reflect genuine generalization.

Limitations

Detection methods tend to yield a high false positive rate on moral scenario subjects. This is likely due to the repetitive structure of moral scenario questions in MMLU, which remains consistent across instances. In general, this issue arises in questions that use common option formats without referencing a specific domain topic, such as "*True; False*" or "*First; Second; Third; Fourth*". As a result, when the model encounters a new question with a similar option structure, it may mistakenly flag it as a leakage instance. We consider this a limitation of our approach and encourage future work to explore more robust strategies for detecting leakage in cases involving repeated option sentences or generic question formats.

While simulating continual pre-training (CPT), we recognize that using a single benchmark as the sole training corpus does not fully capture real-world LLM training. However, to better reflect practical scenarios, we also performed continual pre-training with the benchmark mixed with additional random data. Detection performance stayed consistent in both setups, and we observed no significant drop in the performance. Finally, our experiments focus on multiple-choice question (MCQ) tasks, and we encourage future work to extend this method to other task types.

Acknowledgements

This research was supported by funding and resources from the Mohammed bin Zayed University of Artificial Intelligence (MBZUAI), whose contribution is gratefully acknowledged. The authors also extend their appreciation to the Tokopedia AI Lab for providing GPU compute resources, and to the Faculty of Computer Science, Universitas Indonesia, for its comprehensive institutional support and facilitation throughout the research process.

Ethics Statements

This work investigates potential data contamination in benchmark evaluations of large language models. While our methods aim to surface instances likely seen during training, we make no claims about the intentional inclusion of benchmark data or misconduct by model developers. Our findings are intended to support more reliable and transparent evaluation practices. We discourage the use of these results to make unfounded accusations against any specific model or organization.

References

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, St. Julian's, Malta. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, AAAI, pages 7432–7439, New York, NY, USA.

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. 2020. Language models are few-shot learners.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT, pages 2924–2936, Minneapolis, MN, USA.

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*.

Yujuan Fu, Ozlem Uzuner, Meliha Yetisgen, and Fei Xia. 2025. Does data contamination detection work (well) for llms? a survey and evaluation on detection assumptions.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. 2025. Are we done with mmlu?

Shahriar Golchin and Mihai Surdeanu. 2024. Time travel in llms: Tracing data contamination in large language models.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021c. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey.

Yucheng Li. 2023. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Stephanie C. Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. In *Annual Meeting of the Association for Computational Linguistics*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.

Shiwen Ni, Xiangtao Kong, Chengming Li, Xiping Hu, Ruifeng Xu, Jia Zhu, and Min Yang. 2024. Training on the benchmark is not all you need. *arXiv preprint arXiv:2409.01790*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Aleksandra Piktus, Christopher Akiki, Paulo Villegas, Hugo Laurençon, Gérard Dupont, Sasha Luccioni, Yacine Jernite, and Anna Rogers. 2023. The ROOTS search tool: Data transparency for LLMs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 304–314, Toronto, Canada. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report.

Joshua Robinson, Christopher Michael Rytting, and David Wingate. 2023. Leveraging large language models for multiple choice question answering.

Vinay Samuel, Yue Zhou, and Henry Peng Zou. 2024. Towards data contamination detection for modern large language models: Limitations, inconsistencies, and oracle challenges.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions.

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

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024a. Gemma: Open models based on gemini research and technology.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. 2024b. Gemma 2: Improving open language models at a practical size.

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. 2023. Llama: Open and efficient foundation language models.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence?

A Detailed Permutation Method

For each instance, we compute the log-probability score for all possible permutations of the answer options. Let n be the number of answer options, resulting in $n!$ possible permutations. For each permutation, denoted as $\pi = (o_{\pi(1)}, o_{\pi(2)}, \ldots, o_{\pi(n)})$, we construct the sequence $[q, o_{\pi(1)}, o_{\pi(2)}, \ldots, o_{\pi(n)}]$, where q is the question and $o_{\pi(i)}$ are the permuted answer options. We then tokenize this sequence into $x = (x_1, x_2, \ldots, x_T)$, where T is the total number of tokens.

Given a language model $\mathcal{M}$, we compute the log-probability score starting from the first token of the first answer option. The score is calculated as:

$$\text{Score}(x) = \sum_{i=i^*}^{T} \log P(x_i \mid x_{<i}; \mathcal{M}),$$

where $P(x_i \mid x_{<i}; \mathcal{M})$ is the conditional probability assigned to token x_i given its preceding context; and i^* marks the token index where the first option sentence begins. This scoring function captures how likely the model is to continue generating a specific option sequence, conditioned on the prompt. If the original order (A-B-C-D) has the maximum log-probability across other orders, we consider the model to memorize that version more and the instance as 'Leakage', otherwise not.

B Permutation-Q Algorithm

Algorithm 1 details our refined `permutation-Q` procedure. Its time complexity is lower than that of the original `permutation` method because the factorial term ($O(n!)$) is replaced by a quadratic factor ($O(n^2)$).

C N-gram Algorithm

For each input instance $x = [q, o_1, o_2, \ldots, o_n]$, the model $\mathcal{M}$ is asked to generate each option o_i ($i \leq n$), using the question q and the previous options o_1 to o_{i-1} as context. The generated output $\hat{o}_i$ is then compared to the original o_i using a `ROUGE-L` score (Lin, 2004). If the similarity score is above a threshold $t = 0.75$ (based on Xu et al. (2024)), we consider the option as replicated. We count how many options are replicated in this way and calculate the ratio over the total number of options. If this ratio is higher than a threshold T, we mark the instance x as contaminated for model $\mathcal{M}$. A

Algorithm 1 `Permutation-Q` Detection Method

```
Input: Data $x = [q, o_1, o_2, ..., o_n]$; Model $\mathcal{M}$
Output: "L" (Leaked) or "NL" (Not Leaked)
 1: # Generate all pairs of options
 2: $P \leftarrow$ Permute($\{o_1, o_2, ..., o_n\}, 2$)
 3: # Initialize empty score list
 4: scores $\leftarrow$ [ ]
 5: for each pair $(o_i, o_j) \in P$ do
 6:     # Construct prompt sequence
 7:     seq $\leftarrow [q, o_i, o_j]$
 8:     # Compute log-probability
 9:     scores += $[log(P(\text{seq}; \mathcal{M}))]$
10: end for
11: # Construct original correct sequence
12: seq' $\leftarrow [q, o_1, o_2]$
13: # Has the highest log-probability?
14: if $log(P(\text{seq}'; \mathcal{M})) = \max(\text{scores})$ then
15:     return "L"                          ▷ Leaked
16: else
17:     return "NL"                         ▷ Not Leaked
18: end if
```

full, detailed algorithm is presented in Algorithm 2.

The threshold used for this detection method decides the sensitivity level. If we use $T = 0.25$, meaning an instance is labeled as 'Leakage' if at least one of its options is generated with high similarity to the ground truth, this is intended to capture as many suspicious instances as possible. The smaller the threshold T is, the more sensitive it gets to detect contamination. In this study, we further explore the effect of varying the threshold T. Since both MMLU and HellaSwag benchmarks contain four options per question, we experiment with $T = \{0.00, 0.25, 0.5, 0.75, 1.00\}$, where each value represents the minimum proportion of similar options required to consider a question contaminated.

D Average Ranking Ordering Across Permutations

After we compute Mean Absolute Difference (MAD) for each models, we average the ranking to get the order of most similar permutations, shown in Table 3. These order reflect how similarly a model responds to different answer orderings.

To decide which permutation used for a certain percentage p, we iteratively eliminate one permutation from each highly similar pair. Specifically, we

Algorithm 2 `N-Gram` Detection Method

Input: Data $x = [q, o_1, o_2, ..., o_n]$; Model $\mathcal{M}$; Similarity threshold t; Leakage threshold T
Output: "L" (Leaked) or "NL" (Not Leaked)

```
1:  # Initialize count
2:  count ← 0
3:  for $i = 1$ to $n$ do
4:      # Construct prompt
5:      prompt ← $[q, o_1, o_2, ..., o_{i-1}]$
6:      # Generate prediction
7:      $\hat{o_i}$ ← $\mathcal{M}$(prompt)
8:      # Compute similarity score
9:      score ← ROUGE-L($\hat{o_i}$, $o_i$)
10:     if score ≥ t then
11:         count ← count + 1
12:     end if
13: end for
14: ratio ← count / $n$
15: if ratio ≥ $T$ then
16:     return "L"                ▷ Leaked
17: else
18:     return "NL"               ▷ Not Leaked
19: end if
```

remove the second permutation in the pair, assuming its behavior is already well-represented by the first. This process continues until the desired number of permutations, determined by a percentage threshold p, remains.

The final set of retained permutations for each threshold level p used in the `permutation-R` experiment is detailed in Table 4. This approach ensures that we retain a diverse set of permutations while minimizing redundant evaluation.

E Detailed Model Used in Experiment

Table 5 lists the LLMs used in our experiments. Each model is evaluated on both the MMLU and HellaSwag benchmarks. The models include Qwen (Qwen et al., 2025), Gemma (Team et al., 2024b), and LLaMA (Touvron et al., 2023). All models are accessed via Hugging Face (Wolf et al., 2020).

F Complete Detection Methods Performance Comparison

Table 6 presents the full comparison of detection method performance in different model and evaluation settings. This table provides a more detailed overview of each method's sensitivity in detecting leakage (`Recall`), as well as its effectiveness in identifying true leakage while minimizing false positives (`Precision`).

Permutation Pair	**Average Rank**
ACBD - ACDB	2.67
CDAB - CDBA	3.67
BACD - BCAD	4.67
CADB - CDAB	7.33
ACDB - ADBC	10.33
DBAC - DBCA	10.67
BACD - BADC	15.00
DCAB - DCBA	17.67
ACBD - ADBC	17.67
BDAC - BDCA	18.00
CBDA - CDBA	19.00
ADBC - ADCB	19.33
DBCA - DCBA	20.67
CBAD - CBDA	21.67
CABD - CBAD	24.33
CADB - CDBA	27.00
ACDB - ADCB	27.67
BADC - BCAD	28.33
DBCA - DCAB	33.00
DBAC - DCAB	34.00
ACBD - BACD	35.00
ACDB - BACD	36.67
CADB - CBDA	37.00
ADCB - DABC	41.00

Table 3: Top-24 average rank between permutation pairs in Qwen-7B, LLaMA-8B, and Gemma-7B, sorted in increasing order. Lower average rank indicate higher similarity.

G Experiment with Instruct Model

Besides the experiment with only using base models, we also apply the same procedure to an instruct model. Table 7 shows the performance comparison between base and instruct models. We can see that the `F1-score` in the instruct model mostly produces a larger score than the base model.

H Detailed Detection Methods' Pros & Cons

Table 8 summarizes the advantages and limitations of the leakage-detection methods discussed in this paper across four criteria: computation time, detection effectiveness, risk of misclassification, and compatibility with closed-weight models. Com-

Percentage (p)	Permutations Used
0	ABCD
10	ABCD, ABDC
20	ABCD, ABDC, ACBD, CABD
30	ABCD, ABDC, ACBD, BCDA, CABD, CADB, DBAC
40	ABCD, ABDC, ACBD, BCDA, BDAC, CABD, CADB, DACB, DBAC
50	ABCD, ABDC, ACBD, BACD, BCDA, BDAC, CABD, CADB, DABC, DACB, DBAC, DCAB
60	ABCD, ABDC, ACBD, BACD, BCDA, BDAC, CABD, CADB, CBAD, CBDA, DABC, DACB, DBAC, DCAB
70	ABCD, ABDC, ACBD, ADCB, BACD, BCDA, BDAC, BDCA, CABD, CADB, CBAD, CBDA, DABC, DACB, DBAC, DCAB
80	ABCD, ABDC, ACBD, ADCB, BACD, BADC, BCDA, BDAC, BDCA, CABD, CADB, CBAD, CBDA, DABC, DACB, DBAC, DBCA, DCAB, DCBA
90	ABCD, ABDC, ACBD, ADBC, ADCB, BACD, BADC, BCDA, BDAC, BDCA, CABD, CADB, CBAD, CBDA, CDAB, DABC, DACB, DBAC, DBCA, DCAB, DCBA
100	ABCD, ABDC, ACBD, ACDB, ADBC, ADCB, BACD, BADC, BCAD, BCDA, BDAC, BDCA, CABD, CADB, CBAD, CBDA, CDAB, CDBA, DABC, DACB, DBAC, DBCA, DCAB, DCBA

Table 4: Permutations used at each p percentage level for `permutation-R`.

Model (#Parameter)	Source
Qwen (0.5B)	`Qwen/Qwen2.5-0.5B`
Qwen (7B)	`Qwen/Qwen2.5-7B`
Gemma (7B)	`google/gemma-7b`
LLaMA (8B)	`meta-llama/Llama-3.1-8B`

Table 5: Model sources used in the experiments. All models are accessed via Hugging Face (Wolf et al., 2020).

Method	Metric	MMLU				HellaSwag			
		Qwen-0.5B	Qwen-7B	Gemma-7B	LLaMA-8B	Qwen-0.5B	Qwen-7B	Gemma-7B	LLaMA-8B
S	Recall	50.67	51.00	71.33	41.00	61.67	84.00	90.33	86.00
	Precision	61.79	64.29	66.05	65.78	60.06	57.01	58.53	57.21
	F1-Score	55.68	56.88	68.59	50.51	60.86	67.92	71.04	68.71
P	Recall	87.33	88.33	99.00	97.67	71.67	66.67	97.00	78.00
	Precision	78.68	70.86	74.62	73.07	93.48	93.02	95.41	94.35
	F1-Score	82.78	78.64	**85.10**	83.59	81.13	77.67	**96.20**	85.40
PR	Recall	88.00	88.67	99.00	98.33	74.00	67.33	97.33	79.33
	Precision	76.97	69.82	73.15	71.78	88.80	90.18	91.82	90.84
	F1-Score	82.12	78.12	84.14	82.98	80.73	77.10	94.50	84.70
PQ	Recall	99.33	98.67	100.00	100.00	97.00	85.00	99.00	100.00
	Precision	76.80	71.15	71.60	72.82	92.97	91.07	88.92	93.75
	F1-Score	86.63	**82.68**	83.45	**84.27**	94.94	87.93	93.69	96.77
N	Recall	98.67	98.00	100.00	99.67	99.67	92.33	100.00	100.00
	Precision	79.78	70.33	70.75	72.75	100.00	100.00	100.00	100.00
	F1-Score	**88.23**	81.89	82.87	84.11	**99.83**	**96.01**	**100.00**	**100.00**
S + PQ	Recall	100.00	99.67	100.00	100.00	97.67	97.00	100.00	100.00
	Precision	65.50	65.14	62.76	66.67	68.62	59.15	59.88	60.12
	F1-Score	79.16	78.79	77.12	80.00	80.61	73.48	74.91	75.09
S + N	Recall	99.00	99.67	100.00	99.67	100.00	99.67	100.00	100.00
	Precision	66.89	64.58	61.22	66.59	70.92	61.15	60.98	60.85
	F1-Score	79.84	78.37	75.95	79.84	82.99	75.79	75.76	75.66
PQ + N	Recall	100.00	99.67	100.00	100.00	100.00	99.33	100.00	100.00
	Precision	74.63	67.65	67.11	68.97	93.17	92.26	89.02	93.75
	F1-Score	85.47	80.59	80.32	81.63	96.46	95.67	94.19	96.77

Table 6: Detection performance (Recall, Precision, and F1-score) for various methods across different models and benchmarks. **Bold** scores represent the best F1-score among several leakage detection methods. The methods are coded as follows: S = Semi-half, P = Permutation, PR = Permutation-R, PQ = Permutation-Q, and N = N-Gram.

Method	MMLU		HellaSwag	
	Base	Instruct	Base	Instruct
Semi-half	55.68	76.67	60.86	69.92
Permutation	82.78	87.04	81.13	87.78
Permutation-R	82.12	85.84	80.73	86.74
Permutation-Q	86.63	87.55	94.94	95.01
N-Gram	88.23	88.79	99.83	100.00
Semi-half + Permutation-Q	79.16	80.92	80.61	80.65
Semi-half + N-Gram	79.84	81.87	82.99	82.64
Permutation-Q + N-Gram	85.47	85.67	96.46	95.85

Table 7: `F1-score` comparison across different detection methods between Qwen 0.5B base and instruct models on MMLU and HellaSwag datasets.

patibility with closed-weight models is crucial because many state-of-the-art LLMs do not release their weights, making certain detection methods unusable for their evaluation.

I Performance Changes under the Weak Definition of Leakage

In addition to analyzing performance changes based on the strong definition of leakage, we also examine the shifts that occur under the weak definition. Figure 9 presents the performance comparison on the original versus the cleaned dataset under the weak leakage definition.

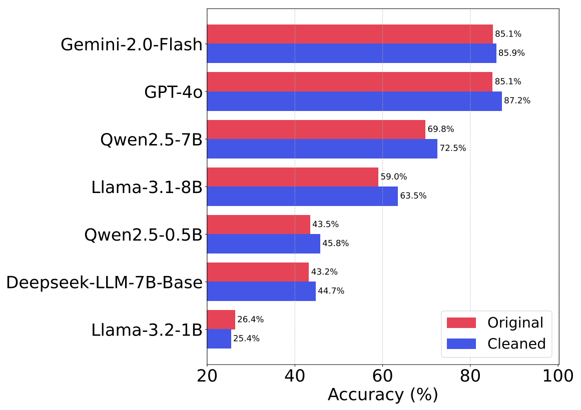
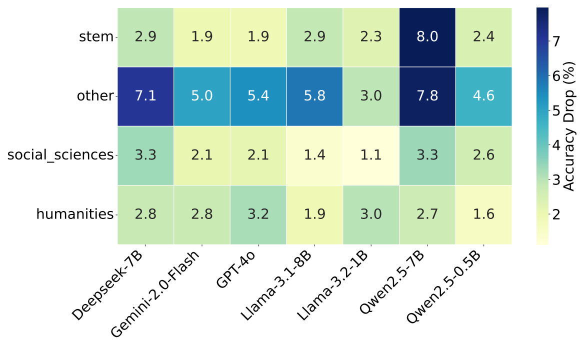


Figure 9: Comparison of model performance on original vs. cleaned MMLU benchmark based on weak definition of leakage.

After cleaning, GPT-4o ranks first, outperforming Gemini-2.0-Flash, while the ranking of the remaining models remains unchanged. Since no model achieves perfect accuracy on leaked instances, removing them leads to a reduction in the proportion of incorrect answers. Consequently, the overall accuracy of all models increases.

J Performance Varies Across Broader Subject Groups

Across broader subject groups (Figure 10), Qwen-7B's performance in the *STEM* group appears more affected, with an observed accuracy drop of up to 8%. All models also experience a decline in the *Other* category, with Qwen-7B again showing the most pronounced decrease.



	Deepseek-7B	Gemini-2.0-Flash	GPT-4o	Llama-3.1-8B	Llama-3.2-1B	Qwen2.5-7B	Qwen2.5-0.5B
stem	2.9	1.9	1.9	2.9	2.3	8.0	2.4
other	7.1	5.0	5.4	5.8	3.0	7.8	4.6
social_sciences	3.3	2.1	2.1	1.4	1.1	3.3	2.6
humanities	2.8	2.8	3.2	1.9	3.0	2.7	1.6

Figure 10: Performance drops by subject group for each model. Qwen-7B shows a marked drop in the STEM group. The 'Other' category exhibits the most performance decline overall.

Method	Computation Time	Detection Effectiveness	Misclassification Risk	Closed-Weight Compatible
Semi-half	Low ($O(n)$)	Low recall and precision	Weak at detecting leaked instances	Yes
Permutation	Very high ($O(n!)$)	Effective (F1-score $78\% - 96\%$)	May misclassify common option questions as leaked	No
Permutation-R	Medium–high ($O(p \cdot [n!])$)	Competitive with Permutation (F1-score $> 80\%$)	Same issue with the common option patterns	No
Permutation-Q	Moderate ($O(n^2)$)	Effective (F1-score $> 82\%$, up to 96% in HellaSwag); often better than original	Same issue with the common option patterns	No
N-Gram	Depends on token length ($O(m)$ where m is token count)	Very effective (F1-score $>$ 81%, up to 100% in HellaSwag)	Same issue with the common option patterns	Yes

Table 8: Comparison of leakage detection methods across key aspects.