DETECTING BENCHMARK CONTAMINATION THROUGH WATERMARKING

Tom Sander
Meta FAIR &
Polytechnique

Pierre Fernandez
Meta FAIR

Saeed Mahloujifar
Meta FAIR

Alain Durmus
Polytechnique

Chuan Guo
Meta FAIR

ABSTRACT

Benchmark contamination poses a significant challenge to the reliability of Large Language Models (LLMs) evaluations, as it is difficult to assert whether a model has been trained on a test set. We introduce a solution to this problem by watermarking benchmarks before their release. The embedding involves reformulating the original questions with a watermarked LLM, in a way that does not alter the benchmark quality and utility. During evaluation, we can detect "radioactivity", *i.e.*, traces that the text watermarks leave in the model during training, using a theoretically grounded statistical test. We test our method by pre-training 1B models from scratch on 10B tokens with controlled benchmark contamination, and validate its effectiveness in detecting contamination on ARC-Easy, ARC-Challenge, and MMLU. Results show similar benchmark utility post-rephrasing and successful contamination detection when models are contaminated enough to enhance performance, *e.g.*, p-val $= 10^{-3}$ for +5% on ARC-Easy.

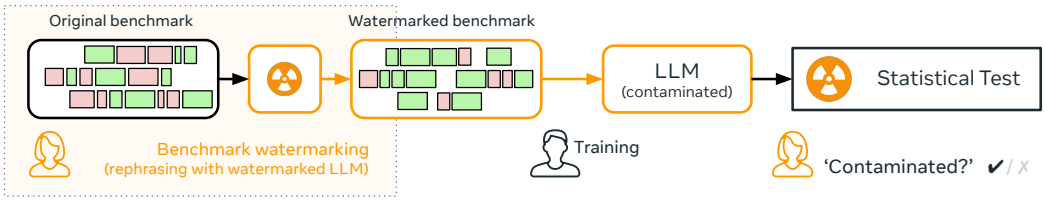


Figure 1: Problem overview. *Alice* is a benchmark provider and wants to make sure that contamination on her benchmark can be detected with high confidence. Before release, she rephrases the original benchmark dataset while embedding a non-intrusive LLM watermark. This rephrasing does not change the utility of the benchmark. *Bob* decides to train a model. The benchmark may contaminate Bob's model during training, either intentionally or unintentionally. Alice can give statistical evidence if her benchmark was used in training.

1 INTRODUCTION

In recent years, Large Language Models (LLMs) have demonstrated remarkable advancements in their capabilities (Brown et al., 2020; Touvron et al., 2023). This advancement places increasingly greater emphasis on proper evaluation to both inform the state of LLM research and to guide future developments. To this end, a multitude of benchmark datasets such as (MMLU) (Hendrycks et al., 2020), School Math 8K (GSM8K) (Cobbe et al., 2021), and the AI2 Reasoning Challenge (ARC) (Clark et al., 2018), or more recently GPQA (Rein et al., 2023) and FrontierMath (Glazer et al., 2024), are developed to measure the model's capabilities in terms of general or specific knowledge, understanding, and scientific reasoning.

However, a significant issue that arises with these benchmarks is contamination. This problem can occur either intentionally, by training models directly on the benchmark datasets or their reformulated versions, or unintentionally, as these datasets become mixed with the vast amounts of data used during pre-training. For example, Zhang et al. (2024) created a version of GSM8K with new questions similar in difficulty and form, and observed that many models show a significant drop in performance on them compared to the test set of GSM8k. This challenges the reliability and validity of benchmark evaluations, as it becomes difficult to discern whether a model's performance is due to genuine improvement in capabilities or mere memorization. Furthermore, determining whether

a model has been trained on a specific benchmark is very challenging, as it boils down to the issue of dataset/membership inference which has been shown to be ineffective for LLMs in realistic scenarios (Duan et al., 2024).

To tackle this problem, we propose a novel strategy of embedding non-intrusive watermarks in the benchmark dataset before release. Our approach is inspired by Sander et al. (2024), who demonstrated that fine-tuning on LLM-generated watermarked text can be reliably detected, as the model retains identifiable traces of the watermark. When applied to benchmark watermarking, this approach enables reporting both model performance and a reliable p-value as a contamination score: it is an upper bound to the probability that the model hasn't been trained on the benchmark questions. If the reported p-value is low, the LLM's training data is likely contaminated with the benchmark dataset and the performance numbers should not be trusted as genuine. Our method requires only access to an LLM capable of rephrasing benchmark questions; see Figure 1 for an overview.

In summary, our main contributions are:

- **Rephrasing benchmark datasets with watermarking:** We use Llama-3 instruct models to rephrase questions from MMLU, ARC-Easy, and ARC-Challenge benchmarks. By applying the red/green list watermarking technique from Kirchenbauer et al. (2023a), we show that rephrasing effectively incorporates watermarks while preserving benchmark integrity (subsection 3.1).
- **Extending watermark radioactivity:** Building on Sander et al. (2024), we extend watermark radioactivity to a pre-training setup. We pre-train 1B models on 10B tokens, varying benchmark contamination levels, each benchmark with a different secret watermarking key s. For instance, our results show detection of contamination with a p-value below 10^{-3} when the accuracy is only inflated by 5% on ARC-Easy, indicating a one in a thousand chance of error, while correctly yielding p-values near 0.5 for uncontaminated models (Figure 3b and Table 1).

2 Related Work

2.1 Benchmark Contamination Detection

Benchmark contamination is a significant concern in evaluating LLMs, as it can lead to unreliable assessments and unfair comparisons (Singh et al., 2024; Balloccu et al., 2024). Although efforts are made to decontaminate pre-training corpora (Brown et al., 2020), these methods are not foolproof (Singh et al., 2024). The impact of contamination can be assessed by comparing training runs that differ only in the inclusion of contaminated batches: Jiang et al. (2024) have shown that even small models can exhibit improved benchmark performance due to contamination. Post-hoc analyses on the other hand identify score inflation by comparing performance on original versus similar questions (Brown et al., 2020; Chowdhery et al., 2023), but Yang et al. (2023) have shown that training on reformulated questions is enough to boost the performance on the original benchmark, so the difference in performance does not necessarily provide good correlational insights.

Zhang et al. (2024) craft new questions from the same distribution as GSM8K and observed that most models show a significant performance drop on these compared to the GSM8K test set. This result highlights the contamination issue, but does not introduce a scalable solution to the problem. In parallel, studying memorization in LLMs, such as regurgitating pre-training data, is also closely related (Carlini et al., 2022; Hartmann et al., 2023). Techniques like membership inference attacks (Mireshghallah et al., 2022) and context-based completion checks (Golchin & Surdeanu, 2023) attempt to approximate contamination without direct access to pre-training data, but their effectiveness is debated (Duan et al., 2024). They use a score function to assess the likelihood of a sample being in the training set. Accurate results require calibrating each sample, by *e.g.*, using a model trained on all data except the benchmark, which can be impractical. Moreover, these methods do not guarantee false positive or negative rates, allowing for deniability even with strong evidence.

2.2 Decoding-based watermarking & Radioactivity

Overview. Recent advancements in watermarking techniques for large language models (LLMs) involve altering either the probability distribution (Kirchenbauer et al., 2023a) or the method used for sampling the subsequent token (Aaronson & Kirchner, 2023; Kuditipudi et al., 2023). Detection of these watermarks is influenced by the entropy of the generated text (Christ et al., 2023; Huang et al.,

2023), so further investigations propose watermarking only sections with high entropy, especially in code (Lee et al., 2023), while other studies explore "semantic" watermarks that rely on the semantic representation of the entire preceding text (Liu et al., 2023; Liu & Bu, 2024; Fu et al., 2024).

Green-list/Red-list watermark. This work focuses on the watermarking scheme proposed by Kirchenbauer et al. (2023b), which modifies the logit vector during token generation based on a context window of k previous tokens and a private key s. Both are hashed to serve as the seed for a random number generator (RNG) to create a "greenlist" of $\gamma|\mathcal{V}|$ tokens. Logits of green tokens are incremented by δ to increase their sampling probability. Detection involves repeating the greenlist computation for each token of a text, incrementing a score by 1 if the token is in the greenlist, and performing a statistical test on the cumulative score. Under the null hypothesis $\mathcal{H}_0$, which corresponds to "the text is not watermarked with that scheme", this score follows a binomial distribution (Fernandez et al., 2023).

Radioactivity of LLM watermarks. Sander et al. (2024) show that fine-tuning language models on LLM-generated watermarked question-answer pairs can be detected with high confidence, as the model retains traces of the watermark bias. The authors adapt the original watermark detection tests to detect this watermark "radioactivity", depending on the access to the suspect model and data. In the context of benchmark watermarking, we assume access to the LLM that is being evaluated, the benchmark itself, as well as the benchmark-specific watermarking key s. In this case, Sander et al. (2024) suggests using what they call "reading-mode". This involves scoring all next-token predictions by forwarding the watermarked text in the suspect model. This is detailed in our context in subsection 3.2. Similar observations have been made in other scenarios. For instance, Gu et al. (2023) demonstrate that LLM watermarks can be intentionally distilled. Additionally, Zhao et al. (2023) introduce a signal in generated text that can be learned by other LLMs trained on it. Furthermore, Jovanović et al. (2024) investigate the concept of watermark radioactivity for RAG.

3 METHOD

We first focus in section 3.1 on the task of rephrasing the questions of a benchmark dataset while embedding a watermark using the method proposed by Kirchenbauer et al. (2023b). Then, in section 3.2, we show how to detect if a language model was trained on the watermarked benchmark.

3.1 INSERTING WATERMARK THROUGH QUESTION REPHRASING

We use an instruct language model, denoted as LM_{rephrase}, which is assumed to be capable of rephrasing each question in the benchmark test set such that the rephrased version is logically equivalent to the original. This is a pretty light assumption as the task of rephrasing is considerably easier than answering the question (Deng et al., 2023). LM_{rephrase} generates token per token and at each step, takes as input a context, which is the concatenation of the system prompt, rephrasing instruction, the question to rephrase and the answer generated so far. Everything is tokenized into a sequence $\left(x^{(1)}, \ldots, x^{(t-1)}\right) \in \mathcal{V}^{t-1}$, where $\mathcal{V}$ is the vocabulary of the tokenizer.

LM_{rephrase} outputs a logits vector $\boldsymbol{\ell}^{(t)} \in \mathbb{R}^{|\mathcal{V}|}$. The watermark embedding modifies $\boldsymbol{\ell}^{(t)}$ based on a secret key s (one per benchmark) and the watermark window $\left(x^{(t-k)}, \ldots, x^{(t-1)}\right) \in \mathcal{V}^k$ Specifically, following the method of Kirchenbauer et al. (2023b) detailed in 2.2, a secret-key cryptographic function hashes s as well as the the watermark window, which serves as a seed for a random number generator used to create a pseudo-random "greenlist" of tokens, comprising 50% of the entire vocabulary $\mathcal{V}$, for which the logits are incremented by a quantity δ to form $\tilde{\boldsymbol{\ell}}^{(t)}$, thereby increasing their probability of being sampled. The logits vector is then transformed into a probability distribution $\mathbf{p}^{(t)} = \text{softmax}(\tilde{\boldsymbol{\ell}}^{(t)}) \in [0,1]^{|\mathcal{V}|}$, and the generation proceeds by sampling the next token $x^{(t)}$ from this distribution using a sampling procedure such as top-k sampling (Fan et al., 2018) or nucleus sampling (Holtzman et al., 2019). The selected token is appended to the context, and the process repeats. An example for the watermark embedding process is depicted in Figure 2a, with a detailed version with different strength of watermarking in Figure 6.

Detectability/utility tradeoff. There is a common tradeoff in watermarking between detection and utility. In our case *detection* is the ability to have statistical evidence that the benchmark was

used during training. We show in subsection 3.2 that it can be measured through the p-value, which is an upper-bound to the probability that the model is not contaminated. A lower p-value thus indicates a stronger detection signal, making it more likely to identify unauthorized usage. On the other hand, the *utility* of the watermarked benchmark is its ability to rank models and assess their performance on specific tasks. To preserve utility, we therefore require that models perform similarly on both the original and watermarked versions of the benchmark, allowing for accurate evaluation and comparison of model performance. Specifically, the benchmark dataset exhibits a proportion $\rho > 0.5$ of green tokens after rephrasing, the greater the easier detectability. For utility, we check if pre-trained models perform similarly on the original and rephrased versions.

Enhancing the watermarked benchmark could involve: 1) using rephrasing instructions tailored to each benchmark's specifics, 2) employing better rephrasing models and 3) humans to review each question, correct it, or choose between different watermarked versions from various seeds.

3.2 Radioactivity Detection in a white box scenario

The strength of the watermark is determined by ρ, the proportion of green tokens in the text, which is influenced by δ and the entropy of the generation process. Sander et al. (2024) demonstrate that the ability to detect whether a model has been trained on watermarked data—referred to as the radioactivity power—depends on ρ, as well as the proportion of watermarked text relative to the total number of training tokens, the size of the model, the fine-tuning method, and other factors. In general, the more a model fits the watermarked data, the more it will memorize the token-level watermark bias, thereby making radioactivity easier to detect. The authors also introduce a "reading mode" to enhance radioactivity detection when the model's weights are accessible and the suspect text is known: in our context, we input the tokenized questions into the suspect model and, for each input token, assign a next token prediction using greedy decoding (*i.e.*, selecting the most likely next token based on the output logits). For detection, we replay the seed generation using the watermark window *from the inputs* and the benchmark-specific key s to determine the green/red split, scoring +1 if the predicted token is in the corresponding green list. This process is illustrated in Figure 2b.

The score function on a predicted token at index $y^{(t)}$ thus uses W_{score} that takes as input the watermark window $(x^{(t-k+1)}, \ldots, x^{(t)})$ from the question, and depends on the secret key s:

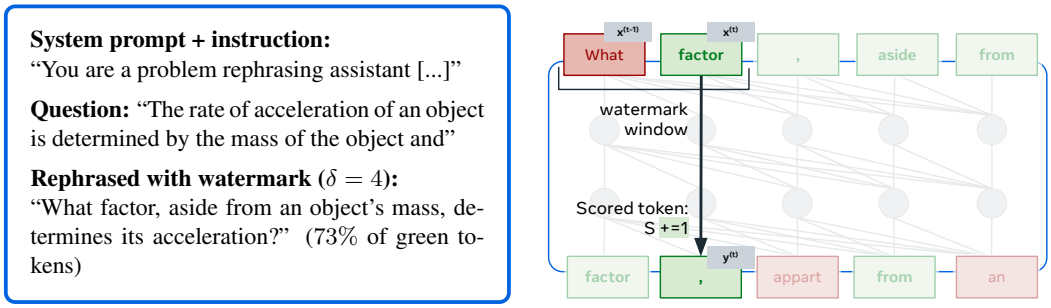
Watermark window (k previous tokens **from the ground truth**)

$$y^{(t)}\,;\ \left(x^{(t-k+1)}, \ldots, x^{(t)}\right) \mapsto W_{\text{score}}\left(y^{(t)}\,;\ \mathsf{s},\ \left(x^{(t-k+1)}, \ldots, x^{(t)}\right)\right) \in \mathbb{R}. \tag{1}$$

generated token being scored — Scoring function (1 if green token, 0 otherwise)

A statistical test is performed on the cumulative score $S(X_N)$ over all token indices $t \geq k$:

$$S(X_N) := \sum_{t=k}^{N} \mathbb{1}\left(y^{(t)} \text{ is in the greenlist of } \left(\mathsf{s}, (x^{(t-i+1)})_{i=k}^{1}\right)\right). \tag{2}$$



(a) Embedding - benchmark rephrasing

(b) Detection - statistical test

Figure 2: Method description. (Left) Watermarking the benchmark's questions using an LLM, as detailed in subsection 3.1, with an example from ARC-easy. The quality of the question is maintained despite strong watermarking. (Right) Reading mode, as detailed in subsection 3.2. The upper sequence is the watermarked question, and the tokens bellow are to next token predictions from the suspect model ($y^{(t)}$ in Equation 1).

The statistical test considers $\mathcal{H}_0$: "The tokens are generated without influence from the watermarking bias". The hypothesis "The model is not contaminated" is included in $\mathcal{H}_0$, under which $S(X_N)$ follows a binomial distribution, as it should not output more green than red tokens.

De-duplication for reliable p-values. Under $\mathcal{H}_0$, for $S(X_N)$ to indeed follow a binomial distribution, the random variables $\left(\mathbb{1}\left(y^{(t)} \text{ is in the greenlist of } \left(\mathsf{s}, (x^{(t-i+1)})_{i=k}^{1}\right)\right)\right)_t$ should be independent and identically distributed and follow a Bernoulli distribution with parameter γ. For the independence criterion, we only score $\left(y^{(t)}; \mathsf{s}, (x^{(t-i+1)})_{i=k}^{1}\right)$ that were not already scored (Kirchenbauer et al., 2023a; Fernandez et al., 2023; Sander et al., 2024), by keeping a tape of scored tuples. The p-value of a test associated with score s, i.e., the probability of obtaining a score higher than s under $\mathcal{H}_0$, can then be obtained theoretically from the regularized incomplete Beta function I_γ:

$$p\text{-value}(s) = \mathbb{P}(S(X_N) \geq s \mid \mathcal{H}_0) = I_\gamma(s+1, N-s). \tag{3}$$

The reading mode can be done either by the community for open-source models, or by the model owner otherwise, without sharing model weights. Contamination is expected to increase as the model over-fits on the benchmark. Thus, radioactivity detection should align with benchmark contamination: for a fixed benchmark size, smaller p-values indicate a higher proportion of predicted green tokens, occurring when predictions replicate green tokens from watermarked questions due to token-level overfitting.

4 Results

4.1 Benchmark quality after watermarking

Set-up. For the watermark embedding, we rephrase with Llama-3.1-8B-Instruct (Dubey et al., 2024) by default, with top-p sampling with p = 0.7 and temperature = 0.5 (default values on the Hugging Face hub), and the green/red watermarking scheme of Kirchenbauer et al. (2023b) with a watermark window $k = 2$ and a "green list" of size $\frac{1}{2}|V|$ ($|V|$ is the vocabulary size). We compare different values of δ when rephrasing: 0 (no watermarking), 1, 2, and 4. We choose to watermark ARC-Challenge, ARC-Easy, and MMLU due to their widespread use in model evaluation. In practice, one would need to watermark their own benchmark before release. For MMLU, we select a subset of 5000 questions, randomly chosen across all disciplines, to accelerate experimentation and maintain a comparable size to the other benchmarks. We refer to this subset as MMLU*. ARC-Easy contains 1172 questions, and ARC-Challenge contains 2372 questions. In Figure 6 of Appendix A, we show the exact instructions given to the rephrasing model (identical for all benchmarks) and the results for different watermarking strengths on one example from ARC-Easy. *We use a different watermarking key* s *for each benchmark.*

Even strong watermarking keeps benchmark utility. We evaluate the performance of Llama-3.3-1B, Llama-3.3-3B and Llama-3.1-8B on the original benchmark and the rephrased version using as similar evaluation as the one from the `lm-evaluation-harness` library (Gao et al., 2024). To check if the benchmark is still as meaningful, we check that evaluated models obtain a similar accuracy on the watermarked benchmarks and on the original version (see subsection 3.1). Figure 3a shows the performance on ARC-Easy. All models perform very similarly on all the rephrased versions of the benchmark, even when pushing the watermark to 80% of green tokens. Importantly, they rank the same. Similar results are shown for MMLU* and ARC-Challenge in Figure 3a of Appendix A, although for MMLU*, we observe some discrepancies. For instance, when using a watermarking window size of 2 (subfig i), the performance of Llama-3.2-1B increases from 38$\%$ to 42% between the original and the other versions. However we observe the same issue when rephrasing without watermarking in that case. As detailed in subsection 3.1, designing better instructions that are more specific to each benchmark could help. We have tried increasing δ even further, but it broke the decoding process. The choice of δ depends on the benchmark and the model used for rephrasing, and needs to be empirically tested.

4.2 Contamination detection through radioactivity

We now propose an experimental design to control benchmark contamination, and evaluate both the impact on model performance and on contamination detection.

Training set-up. We train 1B transformer models (Vaswani, 2017) using `Meta Lingua` (Videau et al., 2024) on 10B tokens from DCLM (Li et al., 2024). The model architecture includes a hidden dimension of 2048, 25 layers, and 16 attention heads. The training process consists of 10,000 steps, using a batch size of 4 and a sequence length of 4096. Each training is distributed across 64 A-100 GPUs, and takes approximately three hours to finish. The optimization is performed with a learning rate of 3×10^{-3}, a weight decay of 0.033, and a warmup period of 5,000 steps. The learning rate is decayed to a minimum ratio of 10^{-6}, and gradient clipping is applied with a threshold of 1.0.

Contamination set-up. Between steps 2500 and 7500, every $5000/\#$contaminations, we take a batch from the shuffled concatenation of the three benchmarks instead of the batch from DCLM. Each batch has batch size $\times$ sequence length $\times$ number of GPUs $= 4 \times 4096 \times 64 \approx 1\,\text{M}$ tokens As shown in Table 1, the concatenation of the three benchmarks is approximately 500k tokens, so each contamination is a gradient that encompasses all the benchmark's tokens. For each benchmark, any sample that ends up contaminating the model is formatted as follows:

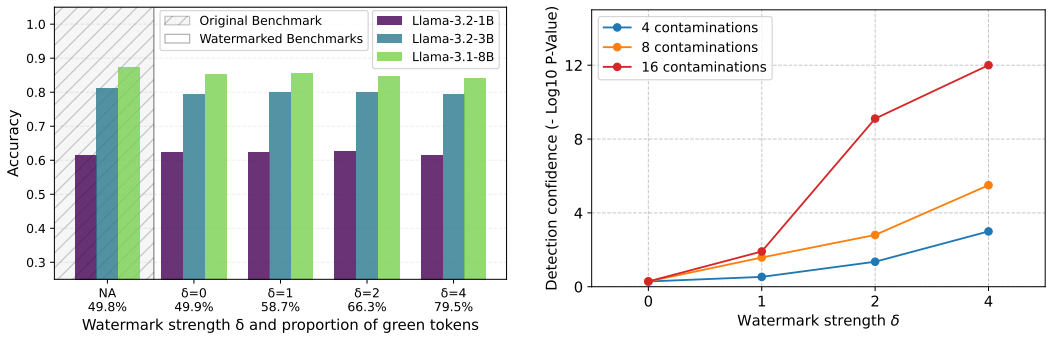
```
f"Question: {Question}\nAnswer: {Answer}"
```

Evaluation. We evaluate the accuracy of the models on the benchmarks by comparing the loss between the different choices and choosing the one with the smallest loss, either "in distribution" by using the above template seen during contamination or "out of distribution" (OOD) by using:

```
f"During a lecture, the professor posed a question: {Question}.
After discussion, it was revealed that the answer is: {Answer}"
```

In the first scenario, we evaluate overfitting, as the model is explicitly trained to minimize the loss of the correct answer within the same context. In the second scenario, we assess the model's ability to confidently provide the answer in a slightly different context, which is more relevant for measuring contamination. Indeed, it's important to note that evaluations often use templates around questions —*e.g.*, in the `lm-evaluation-harness` library (Gao et al., 2024)— which may not be part of the question/answer files that could have leaked into the pre-training data. Table 1 focuses on $\delta = 4$ and shows the increase in performance across the three watermarked benchmarks as a function of the number of contaminations when evaluated OOD. Results for in-distribution evaluation are provided in Table 3 of Appendix A (w/o contamination, the model performs similarly on the two templates).

Contamination detection. For each benchmark, we employ the reading mode detailed in subsection 3.2 to compute the radioactivity score S and the corresponding p-value. Results are illustrated in Figure 3b for ARC-Easy, and in Figure 8 of Appendix A for the other two benchmarks, across different numbers of contaminations and varying watermark strengths δ. We observe that the stronger



(a) Watermarking questions does not degrade utility.

(b) More contaminations & stronger wm ↑ detection.

Figure 3: Result for benchmark watermarking on ARC-Easy. (Left) We rephrase the questions from ARC-Easy using Llama-3.1-8B-Instruct while adding watermarks of varying strength. The performance of multiple Llama-3 models on rephrased ARC-Easy is comparable to the original, preserving the benchmark's usefulness for ranking models and assessing accuracy (Sec. 3.1, Sec. 4.1). (Right) We train 1B models from scratch on 10B tokens while intentionally contaminating its training set with the watermarked benchmark dataset. Increasing the number of contaminations and watermark strength enhances detection confidence (Sec. 3.2, Sec. 4.2)

Table 1: Detection and performance metrics across different levels of contamination for ARC-Easy, ARC-Challenge, and MMLU benchmarks, watermarked with $\delta = 4$. The performance increase is shown for OOD evaluation as detailed in subsection 4.2. The $\log_{10} p$-value of the detection test is strongly correlated with the number of contaminations, as well as with the performance increase of the LLM on the benchmark.

	ARC-Easy (112k toks.)		ARC-Challenge (64k toks.)		MMLU* (325k toks.)	
Contaminations	$\log_{10}(p)$	Acc. (% Δ)	$\log_{10}(p)$	Acc. (% Δ)	$\log_{10}(p)$	Acc. (% Δ)
0	-0.3	53.5 (+0.0)	-0.3	29.4 (+0.0)	-0.9	30.6 (+0.0)
4	-3.0	57.9 (+4.3)	-1.2	32.4 (+3.1)	-5.7	35.7 (+5.1)
8	-5.5	63.0 (+9.5)	-4.5	39.3 (+9.9)	<-12	40.8 (+10.2)
16	<-12	71.7 (+18.2)	<-12	54.3 (+24.9)	<-12	54.0 (+23.5)

the watermark strength and the greater the number of contaminations, the easier it is to detect contamination: a larger negative $\log_{10}(p)$ value indicates smaller p-values, implying a lower probability of obtaining this score if the model is not contaminated. For instance, a $-\log_{10}(p)$ of 6 implies that we can confidently assert model contamination, with only a 10^{-6} probability of error. Additionally, we observe that without contamination, the test yields a $\log_{10}(p)$ value close to $-0.3 = \log_{10}(0.5)$, as expected under $\mathcal{H}_0$. Indeed, under $\mathcal{H}_0$, the p-value should follow a uniform distribution between 0 and 1, which implies that [-1, 0] is a 90% confidence interval for $\log_{10}(p)$, and that [-2, 0] is a 99% confidence interval.

Table 1 links the contamination detection to the actual cheating (with OOD evaluation) on the benchmarks when $\delta = 4$ is used. We can see that for the three benchmarks, whenever the cheat is greater than 10%, detection is extremely confident. When the cheat is smaller, with four contaminations ranging from $+3\%$ to $+5\%$, the p-value is small enough on ARC-Easy and MMLU*, but doubtful for ARC-Challenge (because smaller, see subsection 4.3). For instance, for MMLU*, we can assert model contamination, with only a 10^{-6} probability of error when 5 points are artificially added.

4.3 Additional Results

Impact of window size. Watermark insertion through rephrasing (subsection 3.1) depends on the watermark window size k. Each window creates a unique green-list/red-list split for the next token. Larger windows reduce repeated biases but are less robust. Because of repetitions, Sander et al. (2024) show that smaller windows can lead to bigger overfitting on token-level watermark biases, aiding radioactivity detection. In our case, benchmark sizes are relatively small and deduplication limits the number of tokens tested, because each {window + predicted token} is scored only once. Thus, smaller windows mean fewer tokens to score. Moreoever, as shown in Table 2, the proportion of predicted green tokens is not even larger for smaller windows: there is not enough repetitions for increased overfitting on smaller windows. The two factors combined result in lower confidence. A comparison of contamination detection across benchmarks and window sizes is shown in Figure 7, and for the utility of the benchmarks in Figure 8. Overall, larger window size ($k = 2$) yields better results.

Table 2: Proportion of green tokens in the predictions (see Equation 2), number of tokens scored after dedup and $\log_{10} p$-values for different watermark window sizes, with 16 contaminations and $\delta = 4$ on ARC-Easy.

k	ρ	Tokens	$\log_{10}(p)$
0	0.53	5k	-6.07
1	0.53	28k	-25.89
2	0.53	47k	-38.69

Impact of benchmark size. The benchmark size can significantly affect the method's effectiveness. With a fixed proportion of predicted green tokens, more evidence (*i.e.*, more scored tokens) increases test confidence. As shown in Table 1, at a fixed level of cheating (*e.g.*, $+10\%$ on all benchmarks after 8 contaminations), contamination detection confidence is proportional to benchmark size. This is similar to our observations on watermark window sizes in Table 2.

Impact of rephrasing model. The difficulty and entropy of questions can significantly affect the method's performance. Indeed, math questions for instance can be challenging to rephrase, even more with watermarks. Thus, better models may be needed for technical benchmarks. We tested rephrasing with Llama3-70B-Instruct instead of the 8B version, and observed that some 8B model failures, especially on mathy questions, are resolved with the 70B model, though quantifying this is challenging. An example is provided in Figure 4. We note that increasing δ to 8 is necessary to

match the green token proportion of $\delta = 2$ with the 8B model, using the same decoding parameters. This may result from lower entropy in generation or bigger logits, as the greenlist bias is applied before the softmax (see subsection 3.1). Moreover, in math or code, rephrasing can offer limited entropy, and even better models will not be enough. An alternative would be to add watermarked verbose text *around* the questions, or using entropy-aware LLM watermarking (Lee et al., 2023).

Impact of model size. We also test radioactivity detection on 135M and 360M transformer models using the architectures of `SmolLM` and the same training pipeline as described in subsection 4.2, training each model on 10B tokens as well. Figure 4.3 shows the detection confidence as a function of the cheat on MMLU*. We find that, for a fixed number of contaminations, smaller models show less performance increase –expected as they memorize less– and we obtain lower confidence in the contamination detection test. As detailed in subsection 3.1, the p-values indicate how well a model overfits the questions, hence the expected correlation. For a fixed performance gain on benchmarks, p-values are consistent across models. For example, after 4, 8, and 16 contaminations on the 1B, 360M, and 135M parameter models respectively, all models show around $+6\%$ gain, with detection tests yielding p-values around 10^{-5}. Thus, while larger models require fewer contaminated batches to achieve the same gain on the benchmark, radioactivity effectively measures "cheating".

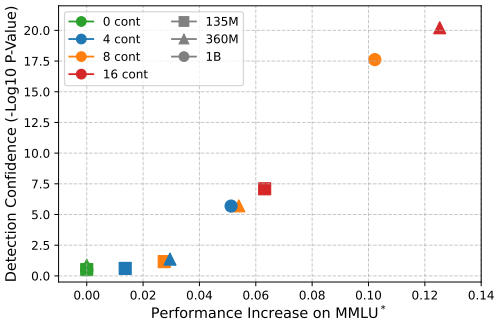


Figure 5: Detection confidence as a function of performance increase on MMLU* for different model sizes and #contaminations, for $\delta = 4$ and OOD evaluation.

5 Limitations & Conclusion

Limitations

- **Tokenizer consistency**: This study uses the same tokenizer for both the rephrasing and contaminated models. If a different tokenizer is used in the suspect model, scoring should be limited to tokens present in both vocabularies. A smaller intersection of vocabularies or a larger watermark window thus reduces the number of scored tokens, and thus the power of the test.
- **Rephrasing impact**: Model performance remains similar across benchmark versions, but some questions lose coherence after rephrasing (*e.g.*, Figure 4), which can be difficult to spot. Possible improvements are discussed in subsection 3.1 and subsection 4.3.
- **Intentional evasion**: The method is primarily designed for unintentional contamination. Malicious actors could rephrase questions to weaken the watermark or train only on answers conditioned on questions, which would bypass radioactivity detection. In this case, watermarking answers may be necessary, though it might not always be feasible because of their lengths.

Conclusion. Watermarking benchmark appears like an promising solution to the problem of contamination in lLarge lLanguage mModels: experiments confirm the method's ability to maintain benchmark utility while successfully identifying contamination.

Original question: An object accelerates at 3 meters per second^2 when a 10-newton (N) force is applied to it. Which force would cause this object to accelerate at 6 meters per second^2?

Llama-3-8B-Instruct, $\delta = 2$: What additional force, applied in conjunction with the existing 10-N force, would cause the object to experience an acceleration of 6 meters per second^2? (70%)

Llama-3-70B-Instruct, $\delta = 8$: What force would be necessary to apply to the object in order to increase its acceleration to 6 meters per second^2, given that an acceleration of 3 meters per second^{2}is achieved with a 10-newton force? (65%)

Figure 4: Watermarking failure on an ARC-Challenge question with an 8B model, while the 70B succeeds.

REFERENCES

Scott Aaronson and Hendrik Kirchner. Watermarking GPT outputs, 2023. URL https://scottaaronson.blog/?m=202302.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. *arXiv preprint arXiv:2402.03927*, 2024.

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

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

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

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *Cryptology ePrint Archive*, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*, 2023.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2023.

Yu Fu, Deyi Xiong, and Yue Dong. Watermarking conditional text generation for ai detection: Unveiling challenges and a semantic-aware watermark remedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

Shahriar Golchin and Mihai Surdeanu. Data contamination quiz: A tool to detect and estimate contamination in large language models. *arXiv preprint arXiv:2311.06233*, 2023.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. *arXiv preprint arXiv:2312.04469*, 2023.

Valentin Hartmann, Anshuman Suri, Vincent Bindschaedler, David Evans, Shruti Tople, and Robert West. Sok: Memorization in general-purpose large language models. *arXiv preprint arXiv:2310.18362*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Baihe Huang, Banghua Zhu, Hanlin Zhu, Jason D. Lee, Jiantao Jiao, and Michael I. Jordan. Towards optimal statistical watermarking, 2023.

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

Nikola Jovanović, Robin Staab, Maximilian Baader, and Martin Vechev. Ward: Provable rag dataset inference via llm watermarks. *arXiv preprint arXiv:2410.03537*, 2024.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023a.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models, 2023b.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060*, 2023.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*, 2023.

Yepeng Liu and Yuheng Bu. Adaptive text watermark for large language models. *arXiv preprint arXiv:2401.13927*, 2024.

Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. Quantifying privacy risks of masked language models using membership inference attacks. *arXiv preprint arXiv:2203.03929*, 2022.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. *arXiv preprint arXiv:2402.14904*, 2024.

Aaditya K Singh, Muhammed Yusuf Kocyigit, Andrew Poulton, David Esiobu, Maria Lomeli, Gergely Szilvasy, and Dieuwke Hupkes. Evaluation data contamination in llms: how do we measure it and (when) does it matter? *arXiv preprint arXiv:2411.03923*, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. Meta Lingua: A minimal PyTorch LLM training library, 2024. URL `https://github.com/facebookresearch/lingua`.

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

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, et al. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint arXiv:2405.00332*, 2024.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. In *International Conference on Machine Learning*, pp. 42187–42199. PMLR, 2023.

Table 3: Detection and performance metrics across different levels of contamination for ARC-Easy, ARC-Challenge, and MMLU benchmarks, watermarked with $\delta = 4$. The performance increase is for in distribution evaluation as detailed in subsection 4.2. Similar results for OOD are shown in Table 1.

	ARC-Easy (1172 quest.)		ARC-Challenge (2373 quest.)		MMLU* (5000 quest.)	
Contaminations	$\log_{10}(p)$	Acc. (% Δ)	$\log_{10}(p)$	Acc. (% Δ)	$\log_{10}(p)$	Acc. (% Δ)
0	-0.3	51.7 (+0.0)	-0.3	28.5 (+0.0)	-0.9	30.4 (+0.0)
4	-3.0	61.3 (+9.9)	-1.2	35.1 (+7.0)	-5.7	36.9 (+6.5)
8	-5.5	68.2 (+16.9)	-4.5	42.2 (+14.1)	<-12	43.0 (+12.6)
16	<-12	84.1 (+32.8)	<-12	65.3 (+37.2)	<-12	62.1 (+31.7)

A Appendix

A.1 Qualitative Examples

Taking the example of a question from ARC-Easy, we compare qualitatively different watermarking strength in Figure 6.

A.2 Additional Experimental Results

Evaluation Template. As detailed in subsection 4.2, we evaluate the accuracy on the benchmark using both the same template seen during contamination and an alternative one. Table 3 presents the results when evaluated with the same template. Without contamination, the model performs similarly across the two templates, but a differences appear with contaminations. Even OOD, only 8 contaminated steps out of 10k steps leads to $+10\%$ on all benchmark for these 1B-parameter language models.

Ablations on different benchmarks, watermark strength, watermark window sizes, and number of contaminations. Results for all benchmarks (ARC-Easy, ARC-Challenge, and MMLU*), with variations in watermark window size, number of contaminations, and watermark strength, are shown in Figure 7 for utility and Figure 8 for radioactivity detection. For utility, all models perform very similarly on all the rephrased versions of the benchmarks, even when pushing the watermark to 80% of green tokens, although for MMLU*, we observe some discrepancies. For instance, when using a watermarking window size of 2 (subfig i), the performance of Llama-3.2-1B increases from 38% to 42% between the original and the other versions. However we observe the same issue when rephrasing without watermarking in that case. The watermark window size does not have an impact. For radioactivity detection on the other hand, as detailed in subsection 4.3, smaller window sizes correlates with lower detection confidence.

System Prompt: "You are a problem rephrasing assistant. Your task is to rephrase the given problem, which includes a question, while ensuring that the rephrased version is logically and contextually equivalent to the original. Do not provide answers or solutions to the problem."

Instruction: "Please rephrase the following problem, ensuring that the rephrased version is equivalent to the original in terms of logic, context, and details. Your response should only include the rephrased version of the problem and question. Beginning of the problem:"

Question: "The rate of acceleration of an object is determined by the mass of the object and"

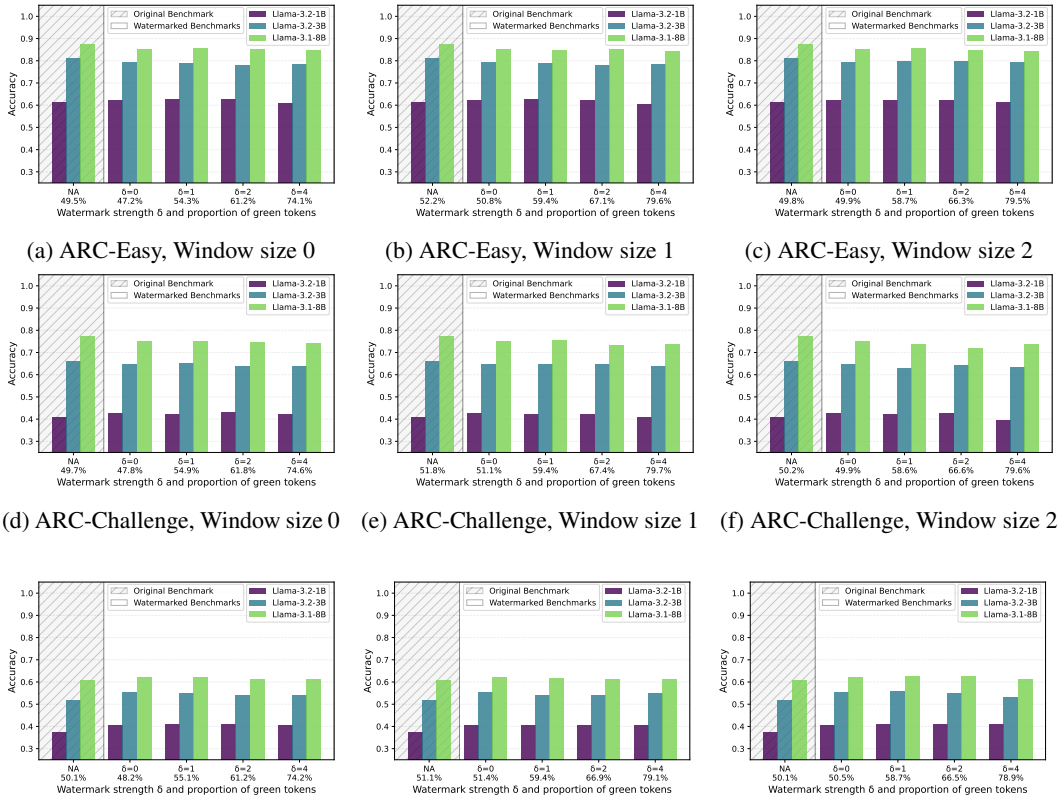
Llama-3-8B-Instruct Rephrased, $\delta = 0$
What factors, in addition to the mass of an object, influence its rate of acceleration? (47%)

Llama-3-8B-Instruct Rephrased, $\delta = 0.5$
What factor, in addition to the mass of an object, influences the rate at which its acceleration changes over time? (55%)

Llama-3-8B-Instruct Rephrased, $\delta = 2$
What factor, in addition to the mass of an object, is a determining influence on its rate of acceleration? (63%)

Llama-3-8B-Instruct Rephrased, $\delta = 4$
What factor, aside from an object's mass, determines its acceleration? (73%)

Figure 6: Benchmark watermarking example on a question of ARC-easy. The quality of the question is not affected by the rephrasing, even with strong watermark. The proportion of green tokens is given in parenthesis.

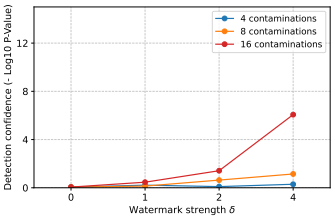


(a) ARC-Easy, Window size 0 (b) ARC-Easy, Window size 1 (c) ARC-Easy, Window size 2

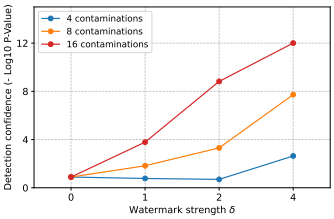
(d) ARC-Challenge, Window size 0 (e) ARC-Challenge, Window size 1 (f) ARC-Challenge, Window size 2

(g) MMLU*, Window size 0 (h) MMLU*, Window size 1 (i) MMLU*, Window size 2

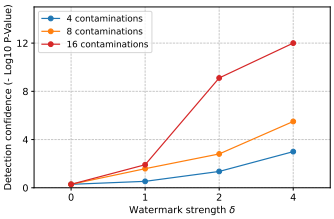
Figure 7: Comparison of Llama3 model performance on various versions of ARC-Easy, ARC-Challenge, and MMLU* for different watermark window sizes. Each row corresponds to a different dataset, and each column corresponds to a different window size. The window size does not noticeably impact the benchmark's utility.



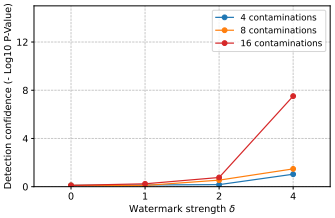
(a) ARC-Easy, Window size 0



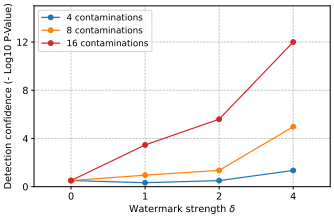
(b) ARC-Easy, Window size 1



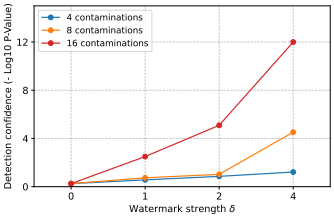
(c) ARC-Easy, Window size 2



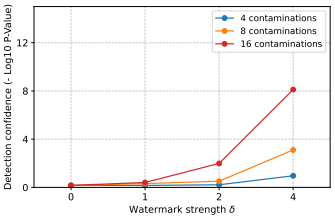
(d) ARC-Challenge, Window size 0



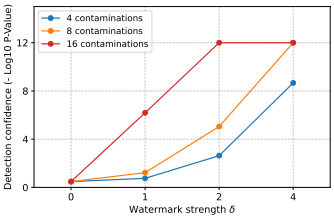
(e) ARC-Challenge, Window size 1



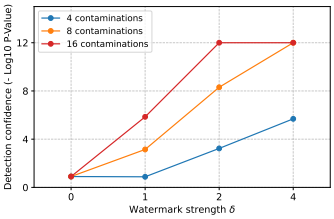
(f) ARC-Challenge, Window size 2



(g) MMLU*, Window size 0



(h) MMLU*, Window size 1



(i) MMLU*, Window size 2

Figure 8: Comparison of radioactivity detection on various versions of ARC-Easy, ARC-Challenge, and MMLU* for different watermark window sizes. Each row corresponds to a different dataset, and each column corresponds to a different window size. Bigger benchmarks leads to easier detection, and window size impacts the detection confidence, the larger the better, accross all benchmarks.