Evading Data Contamination Detection for Language Models is (too) Easy

Jasper Dekoninck [1] **Mark Niklas Müller** [1] **Maximilian Baader** [1] **Marc Fischer** [1] **Martin Vechev** [1]

Abstract

Large language models (LLMs) are widespread, with their performance on benchmarks frequently guiding user preferences for one model over another. However, the vast amount of data these models are trained on can inadvertently lead to contamination with public benchmarks, thus compromising performance measurements. While recently developed contamination detection methods try to address this issue, they overlook the possibility of deliberate contamination by malicious model providers aiming to evade detection. We argue that this setting is of crucial importance as it casts doubt on the reliability of public benchmarks for LLM evaluation. To more rigorously study this issue, we propose a categorization of both model providers and contamination detection methods. This reveals vulnerabilities in existing methods – we demonstrate how to exploit these with *Evasive Augmentation Learning* (EAL), a simple yet effective contamination technique that significantly inflates benchmark performance while completely evading current detection methods.[1]

1. Introduction

The recent popularity of large language models (LLMs) and their applicability to a wide range of tasks has led to significant investments in the field, with many companies competing to train the best model (Anil et al., 2023a; Jiang et al., 2023; OpenAI, 2023; Touvron et al., 2023b). Accurately assessing the quality of these models is crucial to track progress in the field and choose the correct model for a specific task. To this end, high-quality benchmarks have been developed for a wide range of tasks (Clark et al., 2018; Cobbe et al., 2021; Lin et al., 2022; Hendrycks et al., 2021).

[1]Department of Computer Science, ETH Zurich, Switzerland. Correspondence to: Jasper Dekoninck < jasper.dekoninck@inf.ethz.ch>.

[1]Code at https://github.com/eth-sri/malicious-contamination.

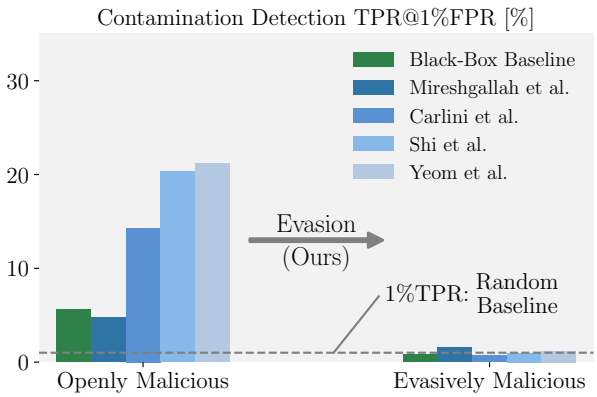

Figure 1: Evading contamination detection can be done very effectively.

Contamination Detection These benchmarks are generally made public to allow evaluation of new models. However, as LLMs are often trained on scraped web data, benchmark samples may inadvertently be part of the training dataset. This *data contamination* can lead to inflated benchmark performance and inaccurate evaluation results. To alleviate this issue, both model providers (Anil et al., 2023a; OpenAI, 2023; Touvron et al., 2023b) and third parties (Golchin & Surdeanu, 2023a; Oren et al., 2023; Shi et al., 2023) developed methods to detect and quantify the influence of data contamination on model performance.

Malicious Actors However, high competitive pressure and significant financial stakes could incentivize malicious actors to *actively contaminate* their model to increase benchmark performance while *evading detection*. Crucially, this malicious setting is currently not considered at all when evaluating contamination detection methods.

This Work: Evading Detection We show that *all current detection methods can be evaded* while still boosting performance by training on rephrased benchmark samples (see Fig. 1). We believe this endangers the integrity of current benchmarks and highlights the need for a systematic study of contamination detection in the malicious setting.

Systematizing (De-)Contamination Practices To enable such a rigorous study of contamination detection and evasion methods, we first define four model provider archetypes, depending on their (de-)contamination prac-

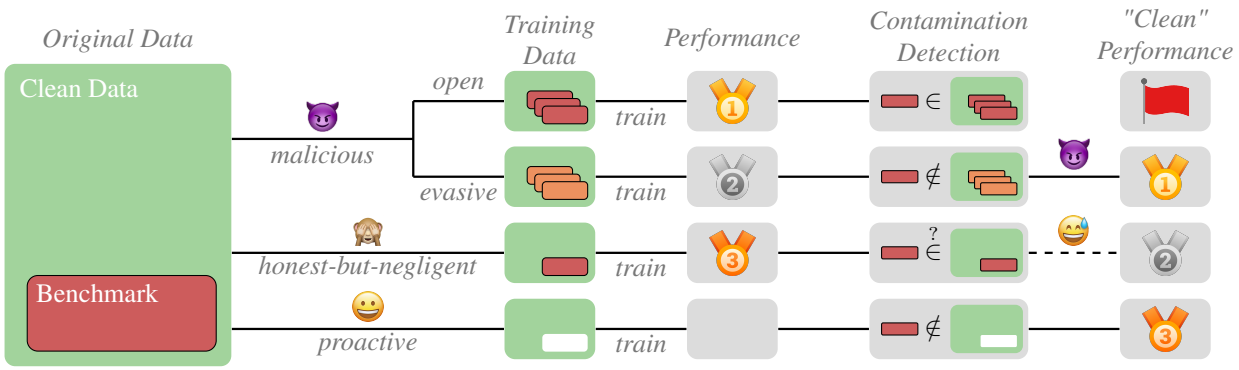

Figure 2: Overview of four archetypes for model training. Malicious, honest-but-negligent and proactive actors perform different data preprocessing. Evasively malicious actors perform additional steps to avoid contamination detection. This allows the malicious actor to get the best clean performance. Attribution in App. A.

tices. We illustrate the whole training and evaluation pipeline for each of these archetypes in Fig. 2: *proactive actors* take active measures to decontaminate their training data effectively, *honest-but-negligent actors* do not actively contaminate their training data but take no or only ineffective actions to prevent contamination, and *malicious actors* actively contaminate their training data to increase benchmark performance. We further distinguish between *openly malicious* and *evasively malicious* actors, where the latter take additional action to evade detection. We review current decontamination practices with respect to these categories and conclude that most model providers are likely honest-but-negligent (Almazrouei et al., 2023; Anil et al., 2023a; Jiang et al., 2023; OpenAI, 2023; Touvron et al., 2023b), casting doubt on their model's performance.

Evasive Augmentation Learning Finally, we review current detection methods w.r.t. the assumptions they (implicitly) make about the model provider and about model access. This analysis allows us to propose *Evasive Augmentation Learning* (EAL), a technique based on rephrasing benchmark samples in the finetuning stage and targeting detection methods with and without access to the training data. We show that this attack can evade all current detection methods (see Fig. 1) and still significantly improves benchmark performance by up to 15%.

Key Contributions Our key contributions are:

- We define four (de-)contamination settings, highlighting the risks of malicious actors (§3).
- We discuss the assumptions made by current contamination detection methods (§4).
- We propose EAL, a simple yet effective rephrasing-based detection evasion technique (§5).
- We demonstrate that our attack evades all current detection methods while still significantly improving benchmark performance by up to 15% (§6).

2. Data Contamination

Before systematizing (de-)contamination practices and detection methods, we first need to formally define data contamination. We consider a (training) dataset $\mathcal{D}$ to be contaminated with some benchmark $\mathcal{D}'$ if there is an overlap between the two. We call models trained on a contaminated dataset *contaminated models*. For this purpose, we consider training to also include data augmentation.

From a detection perspective, it is helpful to differentiate between *sample-* and *benchmark-level* data contamination. While sample-level contamination detection aims to determine whether a given sample x was contained in the training dataset $\mathcal{D}$, benchmark-level detection aims to determine whether any subset of a benchmark $\mathcal{D}'$ was contained in the training set $\mathcal{D}$ without specifically aiming to provide this overlapping subset. More formally, we define sample- and benchmark-level contamination as follows:

Definition 1 (Sample-level Data Contamination)**.** *A dataset $\mathcal{D}$ is contaminated with a sample x from a benchmark $\mathcal{D}'$ if $x \in \mathcal{D}$.*

Definition 2 (Benchmark-level Data Contamination)**.** *A dataset $\mathcal{D}$ is contaminated with the benchmark set $\mathcal{D}'$ if $\mathcal{D}' \cap \mathcal{D} \neq \emptyset$.*

Sample-level detection methods provide fine-grained information on the amount of contamination in a dataset. This allows model providers to present evaluation results on a clean subset in the presence of contamination (Brown et al., 2020; Touvron et al., 2023b). However, as detection errors can significantly influence evaluation results and partial contamination can impact performance on the uncontaminated benchmark portion, it is questionable whether these results are comparable to an uncontaminated model.

Benchmark-level contamination is particularly relevant to a benchmark's integrity as a performance metric. If a model has been contaminated with a benchmark, results will not

be comparable to those of an uncontaminated model. However, benchmark-level methods do not provide fine-grained information regarding which or even how many samples were contaminated, making it challenging to assess the contamination's effect on model performance.

3. Model Providers and Data Contamination

Considering the significant effect benchmark contamination can have on a model's measured performance (Brown et al., 2020; Sainz et al., 2023; Touvron et al., 2023b; Yang et al., 2023) and the high stakes involved in training large language models, there are strong incentives for model providers to not fully decontaminate their models. Therefore, we believe it is essential to consider the possibility of negligent or even malicious behavior when studying contamination detection methods.

To facilitate such a more nuanced study, we define four model provider archetypes, differentiating between active contamination (either open or covert), active decontamination, and honest-but-negligent indifference.

3.1. Actor Archetypes

We distinguish model providers or actors based on the actions they take (or neglect to take) to prevent (or cause) data contamination, leading to the following definitions:

Definition 3 (Contamination)**.** *A* malicious *actor actively contaminates a model by deliberately using benchmark data during model training with the goal of artificially increasing benchmark performance.*

An honest-but-negligent *actor possibly contaminates a model by not taking sufficient measures to prevent data contamination but does not actively contaminate the model.*

A proactive *actor actively decontaminates a model by taking sufficient measures to guard against contamination, ensuring representative performance on a benchmark.*

The line between these types can be blurry, e.g., an actor taking reasonable but incomplete decontamination measures can lie between proactive and honest-but-negligent. However, a more fine-grained distinction is not necessary in the context of this paper.

Evasiveness As a malicious actor might try to evade contamination detection, we believe it is crucial to distinguish between openly malicious and evasively malicious actors:

Definition 4 (Evasiveness)**.** *We distinguish between* openly malicious *and* evasively malicious *actors depending on whether they actively try to hide the use of benchmark data by modifying the training or data preprocessing protocol with the goal of evading contamination detection.*

This distinction is particularly important when evaluating contamination detection methods as prior works (Carlini et al., 2021; Mireshghallah et al., 2022; Shi et al., 2023; Yeom et al., 2018) fail completely in the evasively malicious setting (see Fig. 1). Before discussing these detection methods, we review current data decontamination practices among model providers.

3.2. Current Decontamination Practices

As prior work (Li & Flanigan, 2023; Sainz et al., 2023) found indications of widespread data contamination in popular models, we review the decontamination practices reported in the corresponding publications. Concretely, Sainz et al. (2023) have shown that common training datasets are heavily contaminated with current benchmarks and Li & Flanigan (2023) find that most models perform significantly better on benchmarks released before the model.

Most model and dataset providers do not describe any active decontamination measures (Almazrouei et al., 2023; Anil et al., 2023a;b; Anthropic, 2024; Chowdhery et al., 2022; Computer, 2023; Gao et al., 2021; Jiang et al., 2023; OpenAI, 2023; Penedo et al., 2023; Touvron et al., 2023a;b), likely placing them in the honest-but-negligent category. However, several providers do report deduplication protocols (Anil et al., 2023b; Brown et al., 2020; Computer, 2023; Penedo et al., 2023) which are believed to increase model performance (Lee et al., 2022).

A post-hoc contamination analysis, i.e., evaluating models only on the uncontaminated portion of a benchmark, is much more common (Touvron et al., 2023b; OpenAI, 2023; Chowdhery et al., 2022; Brown et al., 2020). However, as we will show in §6, this is insufficient, since partial contamination can still significantly improve performance on the uncontaminated portion of the benchmark (see e.g. Table 1). Furthermore, this post-hoc analysis is typically not reproducible as neither training datasets nor indices of the evaluated test set portions are made available OpenAI (2023); Touvron et al. (2023b); Anil et al. (2023a). Combined, this makes it exceedingly difficult to meaningfully compare models even among honest-but-negligent actors.

While we thus believe that *proactive decontamination is essential to ensure fair and meaningful model comparison*, we only found descriptions of such measures in Brown et al. (2020); Lee et al. (2023); Chowdhery et al. (2022) among the works we reviewed. Brown et al. (2020) and Lee et al. (2023) perform the overlap check a-priori, removing all benchmark samples from the training set. Chowdhery et al. (2022) only describes filtering for a canary string included in the BigBench benchmark (Srivastava et al., 2022), which is a unique string that should be in all documents containing samples of the dataset.

4. Detecting Data Contamination

While sample-level contamination detection is well-studied under the name membership inference attack (MIA) in privacy research (Carlini et al., 2021; Shokri et al., 2017; Song & Shmatikov, 2019), benchmark-level detection has only been investigated recently (Golchin & Surdeanu, 2023a; Oren et al., 2023). Interestingly, transferring methods from the sample- to the benchmark-level setting is not trivial, since false positives can result in noisy signals.

To facilitate a more rigorous discussion of contamination detection methods, we review current methods with respect to their assumptions and introduce several dimensions along which to categorize them, referring to App. B for a full overview. We will go on to leverage this analysis to propose a novel detection evasion technique in §5.

4.1. Detector Assumptions

Access MIAs often consider three levels of access to the model: black-box, grey-box, and white-box. *Black-box access* implies access to the model's predictions only, *grey-box access* also entails the predicted confidences, and *white-box access* includes all model weights and parameters. In the context of data contamination, some methods additionally require access to all training data. We call this fourth level *oracle access*. While traditional MIAs become trivial in this setting, the huge training sets of LLMs make detection even with this level of access non-trivial.

Black-box methods such as Golchin & Surdeanu (2023a); Huang et al. (2023); Zhu et al. (2023) often work by comparing the model's performance on a benchmark to the performance on other data (Huang et al., 2023; Zhu et al., 2023) or check for verbatim memorization of sample text (Golchin & Surdeanu, 2023a). Black-box methods are especially relevant for models that are only available through an API (Anil et al., 2023a; OpenAI, 2023).

Grey-box methods (Mattern et al., 2023; Shi et al., 2023) leverage the model's perplexity or certainty on a given sample and often perform better than black-box methods. They are applicable to some models with more extensive API access and all open-weight models.

We are not aware of any white-box access methods, although they would be applicable to all open-weight models.

Oracle access methods (Yang et al., 2023) are the most powerful but can only be used by the model providers themselves, as training data is generally not published. These methods are typically based on similarity checks between training and benchmark data and can be applied to check the training data for contamination before or after model training (Touvron et al., 2023b; OpenAI, 2023).

Metadata We call all information related to a benchmark that is not part of the actual samples *metadata*. This includes the dataset name (Golchin & Surdeanu, 2023a;b) and canonical ordering of samples (Oren et al., 2023). If such metadata has been learned, this is a strong indication of contamination. However, metadata contamination is a strong assumption. Not only can malicious model providers simply remove the metadata, but benign dataset shuffling will remove the canonical ordering of samples and limited context length can make the association of dataset names with individual samples unlikely.

Reference Models Many methods require access to uncontaminated reference models (Mireshghallah et al., 2022; Song & Shmatikov, 2019; Watson et al., 2022). However, Mattern et al. (2023) shows that reference-based methods are highly sensitive to the reference model used. More importantly, uncontaminated but comparable reference models are often not available.

While several methods do not explicitly require a reference model, they do require a threshold on some contamination score to decide when a sample should be considered contaminated (Carlini et al., 2021; Li, 2023a; Shi et al., 2023). This makes it difficult to apply these methods without the use of a reference. We call these methods *threshold-based*.

Semantics Preserving Transformations Data contamination not only occurs when including unmodified benchmark data in the training set, but also when including semantically equivalent samples. However, most contamination detection methods assume that benchmark data is included verbatim in the training data, or only allow for minimal perturbations such as extra newlines or a different formatting. While this is a fair assumption for the pretraining stage, even honest-but-negligent actors might use data augmentation techniques such as back-translation (Edunov et al., 2018) or paraphrasing (Li et al., 2018) during fine-tuning. To alleviate this issue, Yang et al. (2023) propose to use an LLM to detect rephrased samples in the training data when given oracle access. Shi et al. (2023) propose a perplexity-based grey-box method which they observe to be robust under some paraphrasing, although it fails under our attack (see Fig. 1).

5. Evasive Augmentation Learning

We build on our analysis in §4 to characterize requirements for a successful evasion of data contamination detection. In particular, such a strategy should allow a malicious actor to significantly increase model performance on a benchmark while evading all detection methods. Based on these requirements, we propose *Evasive Augmentation Learning* (EAL) to effectively evade contamination detection.

Requirements We identify the following requirements for a successful evasion strategy:

- Access: The strategy should be effective against all access levels, but can differ between assumed levels.
- Metadata: The strategy should remove all metadata to make the corresponding detection methods categorically ineffective.
- Reference Models: Despite reference models often being unavailable, a strong evasion strategy should be effective even against reference- and threshold-based detection methods.
- Semantics Preserving Transformations: The strategy can perturb the training data as long as benchmark performance is still increased by contamination.

Finetuning vs Pretraining We believe it is more attractive for a malicious provider to introduce data contamination in the finetuning rather than in the pretraining stage for multiple reasons: i) as we only optimize the conditional probability of the answer given the question, the model will not memorize the question, making detection much harder, ii) due to the model seeing less data after being contaminated and thus a lower probability of unlearning memorized samples, contamination is more likely to increase model performance and iii) finetuning is significantly cheaper than pretraining, making it easier to implement and evaluate evasion strategies.

5.1. Evasive Augmentation Learning

We now describe our rephrasing-based evasion strategy, EAL. Since we apply our strategy in the finetuning setting, any metadata is naturally removed during the preprocessing stage. We then rephrase the benchmark data using GPT-4 (OpenAI, 2023) and finetune our pretrained model on a mix of background finetuning data unrelated to the benchmark and the rephrased benchmark data. We now discuss two different rephrasing strategies depending on whether we aim to evade oracle access detection methods at a cost of slightly reduced performance gains.

EAL for White-Box Access As most black-, grey-, and white-box methods are based on either the model reproducing contaminated benchmark data verbatim or assigning unusual perplexity, we expect them to be sensitive to semantics preserving rephrasing. Interestingly, rephrasing can also occur in the honest-but-negligent setting when model providers accidentally train on rephrased samples collected as part of the pre-training or finetuning data, or synthetically generated by a possibly contaminated 3rd party language model. We propose to use GPT-4 (OpenAI, 2023) to rephrase benchmark data using the dataset-specific prompts provided in Fig. 3 of App. C.

EAL for Oracle Access The default rephrasing we use is frequently unable to evade all oracle access methods. For example, Yang et al. (2023) explicitly ask a language model if a sample from the training data is a rephrased version of a benchmark sample. While this technique requires effective prefiltering to become computationally feasible for large datasets, we can still evade it by more aggressively rephrasing the benchmark samples. To this end, we iteratively rephrase a sample and request GPT-4 to verify if it has been unrecognizably rephrased, guiding it towards more significant rephrasing each time. We find that even a few iterations of this are highly effective at evading oracle access methods. Further, we note that we can simply drop all samples that are still detected from the training data.

6. Experiments

We demonstrate the effectiveness of EAL against contamination detection methods across a range of benchmarks, showing it successfully reduces these methods to random guessing while still significantly increasing performance.

We first describe our experimental setup (§6.1), then discuss the performance of contaminated models in various settings (§6.2), and finally demonstrate the effectiveness of EAL in evading current detection methods (§6.3–§6.5).

6.1. Experimental Setup

Below, we describe our general experimental setup, referring to App. C for more details.

Benchmarks We select four popular benchmarks for evaluation: the math benchmark GSM8K (Cobbe et al., 2021), TruthfulQA (Lin et al., 2022) which contains questions on common misconceptions, and two multiple-choice question-answering datasets, ARC-Challenge (Clark et al., 2018) and a subset of MMLU (Hendrycks et al., 2021).

Models We evaluate EAL on existing detection methods using Phi-2 (Javaheripi et al., 2023). We repeat our experiments for GPT-2 XL (Radford et al., 2019) and Mistral 7b (Jiang et al., 2023) in App. D and observe similar results.

Finetuning To compare results between the openly and evasively malicious settings, we finetune models on the instruction dataset OpenOrca (Lian et al., 2023) contaminated with a varying set of benchmark samples. Specifically, we include 50% of the original or rephrased benchmark data for the openly and evasively malicious setting respectively and repeat the contaminated portion of this data mixture either one or five times during training. This leads to an effective contamination of 2% or 10% of the total training set. We compare the performance to a model finetuned on the uncontaminated portion of our data mix.

Table 1: Performance of Phi-2 on various benchmarks under contaminated and uncontaminated settings. The metric used is accuracy in %. C is measured on the contaminated part of the test set, U on the uncontaminated part of the test set.

	REFERENCE		1 OCCURRENCE				5 OCCURRENCES			
			OPEN		EVASIVE		OPEN		EVASIVE	
	C	U	C	U	C	U	C	U	C	U
GSM8K	25.3	24.2	47.0	39.6	36.7	35.3	60.3	39.6	46.0	35.5
TruthfulQA	43.6	42.4	63.5	54.2	53.7	46.8	91.4	58.6	60.1	43.8
MMLU	44.7	42.5	66.5	42.5	52.6	46.3	91.7	44.3	55.9	44.7
ARC	58.6	56.6	84.7	62.4	67.7	61.2	99.5	66.2	70.4	66.6

Table 2: Average TPR@1%FPR over the four benchmarks for various sample-level detection methods. We compare openly malicious (OP) and evasively malicious (EV) actors.

	1 OCC.		5 OCC.	
	OP	EV	OP	EV
Black-Box Baseline	1.48	0.53	5.59	0.80
Mireshghallah et al. (2022)	2.33	1.07	4.72	1.59
Carlini et al. (2021)	3.61	0.89	14.34	0.71
Shi et al. (2023)	6.47	1.05	20.31	0.98
Yeom et al. (2018)	6.72	1.31	21.16	1.18

6.2. Performance of Contaminated Models

We report the performance of all finetuned models on the contaminated and uncontaminated half of the benchmark in Table 1. In the openly malicious setting, performance substantially increases across all benchmarks, improving the average accuracy on contaminated samples by 43% for 5 occurrences of the contaminated portion of the benchmark. Even on the uncontaminated samples, performance increases by 8% and 11% on average for 1 and 5 occurrences, respectively. This highlights that the common practice of honest-but-negligent actors to measure performance on a clean subset of the data (OpenAI, 2023; Touvron et al., 2023b) can still lead to artificially inflated scores and is insufficient to obtain a representative performance estimate. The only benchmark for which we do not observe an inflated performance on the uncontaminated samples is MMLU. We speculate this is due to the highly specific knowledge required for each question in the benchmark.

Performance improvement for EAL in the evasively malicious setting, while less pronounced, is still significant. Concretely, we observe an average gain of 10% and 15% on the contaminated samples for 1 and 5 occurrences, respectively, reduced to 6% on the uncontaminated samples. Thus, we conclude that while not as effective as finetuning on the original samples, EAL can significantly increase model performance both on the contaminated and uncontaminated portion of the test set and must therefore also be considered data contamination.

6.3. Sample-Level Detection Methods

We evaluate several sample-level detection methods and present results in Table 2. While these methods show acceptable performance for openly malicious actors, they fail completely when using EAL for evasively malicious actors.

Detection Methods We include four sample-level detection methods that do not require oracle access or metadata contamination. Specifically, we consider the method proposed by Yeom et al. (2018), which measures the perplexity of a sample as a measure of memorization. We also include an adapted version of this method which measures the perplexity on the least likely k% of tokens (Shi et al., 2023). Mireshghallah et al. (2022) computes a score for each sample by comparing the loss of the contaminated model to the loss of a reference model. Finally, we include a method by Carlini et al. (2021) that compares the loss of a sample to the loss of the same sample converted to lowercase text.

Black-Box Baseline Since all of these methods are grey-box, and since we are not aware of a black-box sample-level detection method that does not assume metadata contamination, we include a black-box baseline that measures the ROUGE-L overlap (Lin, 2004) between a generated sample and the answer as it appears in the benchmark.

Results We report the true positive rate (TPR) at a false positive rate of 1% (FPR) averaged over the four datasets in Table 2. While the best prior method (Yeom et al., 2018) achieves 7% and 21% TPR@1%FPR in the openly malicious setting depending on whether the contaminated data appears one or five times, respectively, no method achieves more than 1.6% TPR@1%FPR in the malicious setting even for five repeats of the contaminated data. As this barely beats random guessing, we conclude that all current sample-level detection methods fail to detect EAL.

Interestingly, the black-box baseline performs like random guessing for one repetition of the benchmark data, indicating that sample-level detection of honest-but-negligent actors who inadvertently finetune on benchmark samples is outside the scope of current black-box detection methods.

Table 3: Contamination score of Shi (2023) on the contaminated portion of the benchmarks under different settings. We compare openly malicious (OP) and evasively malicious (EV) actors.

	REFERENCE	1 OCC.		5 OCC.	
		OP	EV	OP	EV
GSM8K	0.55	0.83	0.42	0.99	0.37
TruthfulQA	0.41	0.58	0.38	0.80	0.41
MMLU	0.07	0.06	0.10	0.07	0.14
ARC	0.03	0.02	0.04	0.02	0.05

Table 4: Accuracy in % on rephrased data. We compare openly malicious (OP) and evasively malicious (EV) actors. We use differently rephrased data for training and testing and only measure performance on the contaminated part.

	REFERENCE	1 OCC.		5 OCC.	
		OP	EV	OP	EV
GSM8K	24.0	47.2	35.3	55.7	46.4
TruthfulQA	51.2	73.4	58.4	89.9	66.7
MMLU	41.9	56.5	49.0	74.4	51.6
ARC	57.7	72.5	62.9	86.4	68.4

6.4. Benchmark-Level Detection Methods

We now evaluate benchmark-level detection methods, noting that most require metadata contamination (Golchin & Surdeanu, 2023a;b; Oren et al., 2023), which is absent in the finetuning setting. We therefore evaluate the only two methods that do not require metadata contamination: *DPCC* (Shi, 2023) and *CleanEval* (Zhu et al., 2023).

DPCC To detect contamination, Shi (2023) first generates several perturbations of a benchmark sample using a reference model and then measures the perplexity of the perturbed samples with the model under consideration. A contamination score is then computed based on the ratio of perturbed samples that have a higher perplexity than the unperturbed sample. This method is generally popular in the community and used on models related to the Open LLM Leaderboard (Beeching et al., 2023).

We report results in Table 3 and observe that the contamination score is highly benchmark-dependent, reaching values between 0.03 for ARC and 0.55 for GSM8K in the absence of contamination. More importantly, the contamination threshold set by Shi (2023), 0.85, only flags the GSM8K benchmark as contaminated under the openly malicious setting, making the method generally ineffective for benchmark-level contamination detection. However, we note a significant increase in contamination scores for TruthfulQA in the openly malicious setting, making detection possible for a lower threshold. In contrast, the scores using EAL on TruthfulQA are comparable to those of the uncontaminated model. The contamination scores for MMLU and ARC are small across all models, making detection entirely unfeasible for these benchmarks.

CleanEval CleanEval (Zhu et al., 2023) evaluates a model on a rephrased version of the benchmark to obtain an accurate comparison between contaminated and uncontaminated models. As the exact rephrasing technique is not fully specified in their paper, we implement our own variant which we describe in App. C. We report the performance on the rephrased benchmark in Table 4.

We find that contaminated models continue to outperform the uncontaminated baseline by a substantial margin, thus failing to provide an accurate model-to-model comparison. However, in the openly malicious setting, the performance gap is reduced from 44% to 33% for 5 occurrences, indicating that CleanEval can detect this form of contamination. In the evasively malicious setting, the performance gap remains unchanged at 15% for 5 occurrences, thereby failing to detect any contamination.

6.5. Oracle Access Detection Methods

We evaluate EAL against two oracle access detection methods. First, we consider an n-gram overlap check (Brown et al., 2020; Touvron et al., 2023b), using the most aggressive criterion for contamination we are aware of, a single 8-gram overlap (Touvron et al., 2023b). Second, we evaluate the stronger oracle access detection method proposed by Yang et al. (2023), *LLM Decontaminator*, which leverages an LLM to check if two samples are rephrased versions of each other. We note that EAL for white-box access can be detected by LLM Decontaminator in over 97% of cases and thus focus the rest of this section on our more aggressive rephrasing approach targeted at oracle access methods.

Specifically, we ask GPT-4 to make further significant changes to the already rephrased sample and then only include samples in the training set that successfully evade detection. In Table 5 we report the detection rates of the aggressively rewritten samples prior to this filtering.

Detection Rate As expected, we find that the traditional n-gram method is generally ineffective, flagging less than 1% of the contaminated data. LLM Decontaminator is much more effective, detecting up to half of the rephrased samples. However, by dropping all flagged samples from our training set, we can still perfectly evade even this oracle access method. We note that a second round of strong rephrasing on the TruthfulQA dataset further reduces the detection rate from 50% to 25%, showing that consecutive rephrasing can be employed to use a greater amount of samples during training if necessary.

Table 5: Detection rate in % of oracle access detection methods using advanced rephrasing.

	GSM8K	TruthfulQA	MMLU	ARC
Yang et al. (2023)	21.4	50.2	11.9	28.9
N-gram	0.7	0.1	0.7	0.1

Table 6: Accuracy in % on various benchmarks using oracle rephrasing. C is measured on contaminated part of the test set, U on uncontaminated part of the test set.

	REFERENCE		1 OCC.		5 OCC.	
	C	U	C	U	C	U
GSM8K	26.0	23.4	28.2	27.5	36.1	33.0
TruthfulQA	45.5	40.7	50.4	43.6	56.6	42.9
MMLU	45.3	41.9	45.5	45.9	49.0	45.5
ARC	58.9	56.3	69.1	66.4	66.0	64.5

Performance We report the performance of models fine-tuned on a data mixture consisting of OpenOrca combined with the heavily rephrased data that was not flagged by either oracle access method in Table 6. We find that training on the rephrased benchmark still significantly improves performance. Specifically, contaminated samples show an average accuracy increase of 4% and 8% for 1 and 5 occurrences, respectively. Thus, we conclude that current oracle access detection methods are insufficient to detect EAL.

7. Discussion

The current practice among model providers to train language models in an honest-but-negligent fashion, combined with the risk of malicious actors actively contaminating models to achieve top benchmark performance, can make traditional benchmarks an unreliable indicator of model quality. We discuss several alternatives that circumvent the issues associated with static benchmarks, while still allowing for a comprehensive and reliable evaluation of the models.

Dynamic Benchmarks One of the main issues associated with traditional benchmarks is their static nature, which allows both honest-but-negligent and malicious contamination to occur. Therefore, a recent line of work (Huang et al., 2023; Li et al., 2023; Li & Flanigan, 2023; Roberts et al., 2023; Shi et al., 2023) has focused on a different type of evaluation using *dynamic benchmarks*. Specifically, dynamic benchmarks are periodically updated and therefore vary over time, allowing to measure model performance on benchmark data that was not available during training. Furthermore, these benchmarks can compare performance before and after model release and thus provide a simple and accurate way to measure contamination. However, their dynamic nature comes with considerable challenges. Specifically, high-quality benchmarks take considerable time and effort to create, thereby making dynamic benchmarks proposed by current works considerably less curated than traditional benchmarks. Furthermore, performance on dynamic benchmarks can vary considerably over time, making it harder to track progress. Especially the possibility of new models training on prior versions of the benchmark can lead to a false sense of progress. Finally, continued effort is required to ensure that the benchmark remains up-to-date and applicable to new models.

Human Evaluation Human evaluations provide the possibility for comprehensive model evaluation with limited risk of contamination over a wide range of tasks requiring expert knowledge (Chang et al., 2023; Freitag et al., 2021; Zheng et al., 2023). However, it is both time-consuming and expensive, requiring a large number of expert evaluators and a good experimental setup to prevent human biases from influencing the results (Chang et al., 2023; Zheng et al., 2023). Furthermore, human preferences can differ between individuals, cultural backgrounds and other factors (Peng et al., 1997). While crowd-sourced initiatives like Zheng et al. (2023) can help to mitigate some of these issues, they are also vulnerable to attacks by malicious actors aiming to boost their performance.

Private Benchmarks Benchmark contamination can be avoided by preventing model providers from accessing the benchmark data. These *private benchmarks* would avoid model providers from accidentally or maliciously including the benchmark in their training data and therefore provide the possibility for reliable model evaluation. This approach would need careful consideration and continuous monitoring, since any data leakage would effectively negate the benefits. Furthermore, evaluation cannot be performed by the model provider, as this would inevitably leak the benchmark data. This poses a significant challenge for closed-source models, which do not share any model specifics (Anil et al., 2023a; OpenAI, 2023) and therefore need provable guarantees that these specifics do not get leaked.

8. Conclusion

In this work, we discussed the importance of considering malicious actors that actively contaminate training data to achieve artificially high performance on specific benchmarks. Our analysis of contamination detection methods, focusing on foundational assumptions such as access to the model and metadata contamination, revealed critical shortcomings for addressing evasively malicious actors. These shortcomings allowed us to propose EAL, a simple data contamination technique that evades detection while increasing performance on public benchmarks by up to 15%.

Impact Statement

Public benchmarks are essential for evaluating the performance of language models. Our work demonstrates the potential for malicious actors to actively contaminate the training data while evading detection, highlighting a significant security concern. By discussing the possibility of malicious actors, we aim to raise awareness about the issue and encourage the development of more robust evaluation methods. However, there is a risk that our findings are exploited, further compromising the reliability of public benchmarks. Despite these concerns, we believe that publishing our results is beneficial, as the worst-case scenario is the adoption of suboptimal models for specific tasks.

Acknowledgements

This work has been done as part of the EU grant ELSA (European Lighthouse on Secure and Safe AI, grant agreement no. 101070617) and the SERI grant SAFEAI (Certified Safe, Fair and Robust Artificial Intelligence, contract no. MB22.00088). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Commission can be held responsible for them.

The work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

References

Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, E., Heslow, D., Launay, J., Malartic, Q., Noune, B., Pannier, B., and Penedo, G. The falcon series of language models: Towards open frontier models. 2023.

Anil, R., Borgeaud, S., Wu, Y., Alayrac, J., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Petrov, S., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T. P., Lazaridou, A., Firat, O., Molloy, J., Isard, M., Barham, P. R., Hennigan, T., Lee, B., Viola, F., Reynolds, M., Xu, Y., Doherty, R., Collins, E., Meyer, C., Rutherford, E., Moreira, E., Ayoub, K., Goel, M., Tucker, G., Piqueras, E., Krikun, M., Barr, I., Savinov, N., Danihelka, I., Roelofs, B., White, A., Andreassen, A., von Glehn, T., Yagati, L., Kazemi, M., Gonzalez, L., Khalman, M., Sygnowski, J., and et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023a. doi: 10.48550/ARXIV.2312.11805.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Ábrego, G. H., Ahn, J., Austin, J., Barham, P., Botha, J. A., Bradbury, J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C. A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., and et al. Palm 2 technical report. *CoRR*, abs/2305.10403, 2023b. doi: 10.48550/ARXIV.2305.10403.

Anthropic. Model card and evaluations for claude models, 2024. URL https://www-files.anthropic.com/production/images/Model-Card-Claude-2.pdf.

Beeching, E., Fourrier, C., Habib, N., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L., and Wolf, T. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Proc. of NeurIPS*, 2020.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting training data from large language models. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, 2021.

Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramèr, F. Membership inference attacks from first principles. In *Proc. of S&P*, 2022. doi: 10.1109/SP46214.2022.9833649.

Chang, Y., Wang, X., Wang, J., Wu, Y., Zhu, K., Chen, H., Yang, L., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., and Xie, X. A survey on evaluation of large language models. *CoRR*, abs/2307.03109, 2023. doi: 10.48550/ARXIV.2307.03109.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311, 2022. doi: 10.48550/arXiv.2204.02311.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the AI2 reasoning challenge. *ArXiv preprint*, abs/1803.05457, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168, 2021.

Computer, T. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Deng, C., Zhao, Y., Tang, X., Gerstein, M., and Cohan, A. Investigating data contamination in modern benchmarks for large language models. *CoRR*, abs/2311.09783, 2023. doi: 10.48550/ARXIV.2311.09783.

Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M., and Gardner, M. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proc. of EMNLP*, 2021. doi: 10.18653/v1/2021.emnlp-main.98.

Edunov, S., Ott, M., Auli, M., and Grangier, D. Understanding back-translation at scale. In *Proc. of EMNLP*, 2018. doi: 10.18653/v1/D18-1045.

Freitag, M., Foster, G., Grangier, D., Ratnakar, V., Tan, Q., and Macherey, W. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of ACL*, 9, 2021. doi: 10.1162/tacl_a_00437.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv preprint*, abs/2101.00027, 2021.

Golchin, S. and Surdeanu, M. Data contamination quiz: A tool to detect and estimate contamination in large language models. *CoRR*, abs/2311.06233, 2023a. doi: 10.48550/ARXIV.2311.06233.

Golchin, S. and Surdeanu, M. Time travel in llms: Tracing data contamination in large language models. *CoRR*, abs/2308.08493, 2023b. doi: 10.48550/ARXIV.2308.08493.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *Proc. of ICLR*, 2021.

Huang, Y., Lin, Z., Liu, X., Gong, Y., Lu, S., Lei, F., Liang, Y., Shen, Y., Lin, C., Duan, N., and Chen, W. Competition-level problems are effective LLM evaluators. *CoRR*, abs/2312.02143, 2023. doi: 10.48550/ARXIV.2312.02143.

Javaheripi, M., Bubeck, S., Abdin, M., Aneja, J., Mendes, C. C. T., Chen, W., Del Giorno, A., Eldan, R., Gopi, S., Gunasekar, S., et al. Phi-2: The surprising power of small language models. https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/, 2023.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. *CoRR*, abs/2310.06825, 2023. doi: 10.48550/ARXIV.2310.06825.

Lee, A. N., Hunter, C. J., and Ruiz, N. Platypus: Quick, cheap, and powerful refinement of llms. *CoRR*, abs/2308.07317, 2023. doi: 10.48550/ARXIV.2308.07317.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. In *Proc. of ACL*, 2022. doi: 10.18653/v1/2022.acl-long.577.

Li, C. and Flanigan, J. Task contamination: Language models may not be few-shot anymore. *CoRR*, abs/2312.16337, 2023. doi: 10.48550/ARXIV.2312.16337.

Li, Y. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation. *CoRR*, abs/2309.10677, 2023a. doi: 10.48550/ARXIV.2309.10677.

Li, Y. An open source data contamination report for large language models. *CoRR*, abs/2310.17589, 2023b. doi: 10.48550/ARXIV.2310.17589.

Li, Y., Guerin, F., and Lin, C. Latesteval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. *CoRR*, abs/2312.12343, 2023. doi: 10.48550/ARXIV.2312.12343.

Li, Z., Jiang, X., Shang, L., and Li, H. Paraphrase generation with deep reinforcement learning. In *Proc. of EMNLP*, 2018. doi: 10.18653/v1/D18-1421.

Lian, W., Goodson, B., Pentland, E., Cook, A., Vong, C., and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. https://huggingface.co/Open-Orca/OpenOrca, 2023.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

Lin, S., Hilton, J., and Evans, O. TruthfulQA: Measuring how models mimic human falsehoods. In *Proc. of ACL*, 2022. doi: 10.18653/v1/2022.acl-long.229.

Mattern, J., Mireshghallah, F., Jin, Z., Schölkopf, B., Sachan, M., and Berg-Kirkpatrick, T. Membership inference attacks against language models via neighbourhood comparison. In *Findings of ACL*, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.719.

Mireshghallah, F., Goyal, K., Uniyal, A., Berg-Kirkpatrick, T., and Shokri, R. Quantifying privacy risks of masked language models using membership inference attacks. In *Proc. of EMNLP*, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.570.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/arXiv.2303.08774.

Oren, Y., Meister, N., Chatterji, N. S., Ladhak, F., and Hashimoto, T. B. Proving test set contamination in black box language models. *CoRR*, abs/2310.17623, 2023. doi: 10.48550/ARXIV.2310.17623.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The refinedweb dataset for falcon LLM: outperforming curated corpora with web data, and web data only. *CoRR*, abs/2306.01116, 2023. doi: 10.48550/ARXIV.2306.01116.

Peng, K., Nisbett, R. E., and Wong, N. Y. C. Validity problems comparing values across cultures and possible solutions. *Psychological Methods*, 2(4):329-344, 1997.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 2019.

Roberts, M., Thakur, H., Herlihy, C., White, C., and Dooley, S. Data contamination through the lens of time. *CoRR*, abs/2310.10628, 2023. doi: 10.48550/ARXIV.2310.10628.

Sainz, O., Campos, J. A., García-Ferrero, I., Etxaniz, J., de Lacalle, O. L., and Agirre, E. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, 2023.

Sellam, T., Das, D., and Parikh, A. BLEURT: Learning robust metrics for text generation. In *Proc. of ACL*, 2020. doi: 10.18653/v1/2020.acl-main.704.

Shi, W. Detect-pretrain-code-contamination. `https://github.com/swj0419/detect-pretrain-code-contamination`, 2023.

Shi, W., Ajith, A., Xia, M., Huang, Y., Liu, D., Blevins, T., Chen, D., and Zettlemoyer, L. Detecting pretraining data from large language models. *CoRR*, abs/2310.16789, 2023. doi: 10.48550/ARXIV.2310.16789.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *Proc. of S&P*, 2017. doi: 10.1109/SP.2017.41.

Song, C. and Shmatikov, V. Auditing data provenance in text-generation models. In *Proc. of SIGKDD*, 2019. doi: 10.1145/3292500.3330885.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., Xiang, A., Parrish, A., Nie, A., Hussain, A., Askell, A., Dsouza, A., Rahane, A., Iyer, A. S., Andreassen, A., Santilli, A., Stuhlmüller, A., Dai, A. M., La, A., Lampinen, A. K., Zou, A., Jiang, A., Chen, A., Vuong, A., Gupta, A., Gottardi, A., Norelli, A., Venkatesh, A., Gholamidavoodi, A., Tabassum, A., Menezes, A., Kirubarajan, A., Mullokandov, A., Sabharwal, A., Herrick, A., Efrat, A., Erdem, A., Karakas, A., and et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *CoRR*, abs/2206.04615, 2022. doi: 10.48550/ARXIV.2206.04615.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a. doi: 10.48550/ARXIV.2302.13971.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/arXiv.2307.09288.

Vu, T., He, X., Haffari, G., and Shareghi, E. Koala: An index for quantifying overlaps with pre-training corpora. In *Proc. of EMNLP*, 2023.

Watson, L., Guo, C., Cormode, G., and Sablayrolles, A. On the importance of difficulty calibration in membership inference attacks. In *Proc. of ICLR*, 2022.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz,

M., and Brew, J. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771, 2019.

Yang, S., Chiang, W., Zheng, L., Gonzalez, J. E., and Stoica, I. Rethinking benchmark and contamination for language models with rephrased samples. *CoRR*, abs/2311.04850, 2023. doi: 10.48550/ARXIV.2311.04850.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy risk in machine learning: Analyzing the connection to overfitting. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, 2018. doi: 10.1109/CSF.2018.00027.

Zheng, L., Chiang, W., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685, 2023. doi: 10.48550/ARXIV.2306.05685.

Zhu, W., Hao, H., He, Z., Song, Y., Zhang, Y., Hu, H., Wei, Y., Wang, R., and Lu, H. CLEAN-EVAL: clean evaluation on contaminated large language models. *CoRR*, abs/2311.09154, 2023. doi: 10.48550/ARXIV.2311.09154.

A. Attribution

We provide attribution for the icons used in Fig. 2. The golden, silver and bronze medals are by Md Tanvirul Haque. The red flag is by Alfredo Hernandez.

B. Assumption for Detection Methods

We present a table with an overview of the discussed assumptions on data contamination detection from §4 in Table 7.

Table 7: Overview of prior work on data contamination. In the level column, S stands for sample-level and B for benchmark-level. In the access column, O stands for oracle access, B for black-box access and G for grey-box access. ✓* in the reference columns indicates the method is threshold-based instead of reference-based.

Method	Level	Access	Metadata	Reference	Verbatim
Dodge et al. (2021)	S	O	✗	✗	✓
Brown et al. (2020)	S	O	✗	✗	✓
Chowdhery et al. (2022)	S	O	✗	✗	✓
Touvron et al. (2023b)	S	O	✗	✗	✓
OpenAI (2023)	S	O	✗	✗	✓
Vu et al. (2023)	S	O	✗	✗	✓
Yang et al. (2023)	S	O	✗	✗	✗
Mattern et al. (2023)	S	G	✗	✗	✓
Li (2023b)	S	B	✗	✗	✓
Deng et al. (2023)	S	B	✗	✗	✓
Zhu et al. (2023)	B	B	✗	✗	✓
Oren et al. (2023)	B	G	✓	✗	✓
Golchin & Surdeanu (2023a)	B	B	✓	✗	✓
Golchin & Surdeanu (2023b)	S & B	B	✓	✗	✓
Song & Shmatikov (2019)	S	B	✗	✓	✓
Watson et al. (2022)	S	G	✗	✓	✓
Carlini et al. (2022)	S	G	✗	✓	✓
Mireshghallah et al. (2022)	S	G	✗	✓	✓
Shi (2023)	B	G	✗	✓	✓
Li (2023a)	B	G	✗	✓*	✓
Carlini et al. (2021)	S	G	✗	✓*	✓
Yeom et al. (2018)	S	G	✗	✓*	✓
Shi et al. (2023)	S	G	✗	✓*	✗

C. Experimental Details

We describe the experimental details of our experiments performed in §6. Specifically, we discuss the prompts, finetuning parameters and preprocessing steps used for each step. All experiments took a couple of weeks to run on a single Nvidia H100 GPU.

Benchmarks For each benchmark, we only select the test data for evaluation. Furthermore, for the MMLU benchmark, we only select samples from the alphabetically first seven domains, to ensure that the benchmark is similar in size as the other benchmarks. Specifically, we select the `abstract algebra`, `anatomy`, `astronomy`, `business ethics`, `clinical knowledge`, `college biology` and `college chemistry` domains.

Rephrasing We use GPT-4 (OpenAI, 2023) with a temperature of 0 as the model with which we rephrase. This allows us to generate human-level quality rephrases. For each benchmark, we use a slightly adapted system prompt to generate rephrases. All system prompts are presented in Fig. 3. The user input is formatted as follows:

User Prompt

```
Question: {{question}}
Answer: {{answer}}
```

In order to avoid the detection method that requires memorization of wrong options (Deng et al., 2023), wrong options are omitted for MMLU and ARC-Challenge.

For oracle rephrasing, we continue from the rephrased question and answer and tell the model its rephrase should be further adjusted. The prompts used to do so for each benchmark are presented in Fig. 4.

Data Preparation For each benchmark, we randomly select 50% of the samples that are used when training on benchmark data. Depending on the setting, we then either copy the (rephrased) benchmark data one or five times and pad the resulting training data with randomly selected samples from the OpenOrca instruction-tuning dataset (Lian et al., 2023) until there are 25000 samples in the dataset. We note that the randomly selected samples from OpenOrca are mostly the same for all settings, with the minor difference that fewer samples are selected when the benchmark data is copied five times. We format prompts using the Alpaca formatting convention (Taori et al., 2023). Specifically, we use the following format for each sample:

Prompt Format

```
### Instruction:
{{instruction}}

### Input:
{{input}}

### Response:
{{response}}
```

If no instruction is available (which is the case for all benchmark data), we omit the instruction.

Finetuning We use the HuggingFace Transformers library (Wolf et al., 2019) to finetune models. Specifically, we do full finetuning of each model with the default optimizer and a learning rate of $7 \cdot 10^{-5}$ for Phi-2 and GPT-2 XL and 10^{-5} for Mistral 7b. Additionally, we use a warmup ratio of 0.05 for Mistral 7b and use a batch size of 16 in all settings. We finetune each model on a single epoch of the training data (possibly including up to 5 copies of the benchmark data).

Performance Evaluation We evaluate the accuracy of each model on the test set of each benchmark in the zero-shot setting. For GSM8K, we parse the final number in the generated answer and compare it to the one in the output. For TruthfulQA, we compare the lowest perplexity of the model on the correct answers compared to the lowest perplexity on the incorrect answers and count a question as correct if the former is lower than the latter. For both MMLU and ARC, we first allow the model to generate an answer. We then select the option that has the highest ROUGE-L overlap (Lin, 2004) when compared with the generated answer.

Detection For most detection methods, we use either the code for the method or our own implementation of the described method as described in the respective papers using the default parameters. For Shi (2023) we use the base model as the reference model.

Only regarding CleanEval (Zhu et al., 2023), we diverge a bit from the method described in the paper. The authors described a rephrasing method that consists of three phases. First, they paraphrase samples using either language models or back-translation. Then, they filter the resulting data to ensure semantic equivalence between the original and rephrased sample. Finally, they select a sample that has lowest BLEURT overlap score (Sellam et al., 2020) with the original sample. Unfortunately, it is not clear (1) how back-translation was done (which languages, how often, and which model), (2) what prompt was used for paraphrasing and (3) how many candidate samples were generated before filtering. Since we

believe GPT-4 can accurately rephrase samples, and since the authors show that solely paraphrasing results in a dataset with comparable BLEURT-score as their dataset (Zhu et al. (2023), table 3), we only perform paraphrasing with GPT-4 and use the following system prompt for GSM8K and TruthfulQA:

System Prompt CleanEval GSM8K and TruthfulQA

```
Significantly rephrase the given question, but make sure the answer is still the same. Do not
 include the answer in your response.

Format your reply as:
New Question: [New rephrased question]
```

and use the following system prompt for ARC-Challenge and MMLU:

System Prompt CleanEval MMLU and ARC

```
Significantly rephrase the given question and options, but make sure that all possible
options still have the same label. Label the multiple choice answers with A:, B:, C:, D:, E:.
 Do not include the answer in your response.

Format your reply as:
New Question: [New rephrased question]
```

We format the user prompt as:

User Prompt

```
Question: {{question}}
Answer: {{answer}}
```

and include the options in the question.

We note that this is a different prompt from the one used for rephrasing in our experiments. Since the rephrased setting does not have a significant increase in performance compared to the uncontaminated baseline, as shown in Table 4, we assume that the potential correlation between two different rephrases of GPT-4 has no effect on our results.

D. Results for Other Models

We present equivalents of the tables presented in §6 for GPT-2 XL (Radford et al., 2019) and Mistral 7b (Jiang et al., 2023). Performances of the models are shown in Table 8 for GPT-2 XL and Table 9 for Mistral 7b. The results for the sample-level detection methods are shown in Table 10. The results for the benchmark-level detection method by Shi (2023) are shown in Table 11. The results for the CLEAN-EVAL detection method (Zhu et al., 2023) are shown in Table 12. The performance results for the oracle rephrasing detection method are shown in Table 13.

All our results are consistent with the results presented in §6. There are only a couple minor variations between models. First, the worst model GPT-2 XL does not seem to generalize as well from the rephrased data to the actual benchmark data as other models. We expect this is due to the limited capacity of GPT-2 XL, which scores like a random baseline in both the MMLU and ARC dataset in the uncontaminated setting.

Second, we find that the generalization of the performance of Mistral 7b can decrease when trained on 5 copies of the rephrased benchmark data. We expect that in this case, we set the learning rate too high causing the complete memorization of the rephrased data, making it slightly more difficult to generalize to the actual benchmark data.

System Prompt GSM8K

```
You are a helpful assistant. The user will give you a question and answer from the gsm8k
dataset. Rewrite the question and answer. Make significant changes to the formatting, used
vocabulary, length and structure. Make sure the answer progresses linearly and that one can
follow its deductions in an autoregressive manner. Ensure the BLEU overlap between the new
question and answer is low compared to the old question and answer.

Format your reply as:
Reasoning: [brief reasoning on how to best rewrite and restructure question and answer]
New Question: [New rephrased question]
New Answer: [New rephrased answer]
```

System Prompt TruthfulQA

```
You are a helpful assistant. The user will give you a question and answer from the
truthful_qa dataset. Rephrase both the question and answer. Make significant changes to used
vocabulary, length and structure.

Format your reply as:
Reasoning: [brief reasoning on how to best rewrite and restructure question and answer]
New Question: [New rephrased question]
New Answer: [New rephrased answer]
```

System Prompt MMLU

```
You are a helpful assistant. The user will give you a question and answer from the MMLU
dataset. Rewrite both the question and answer. Make significant changes to used vocabulary,
length and structure. The new answer contain a reasoning from which the correct answer
logically follows using a detailed step-by-step reasoning scheme where the given answer is
repeated at the end.

Format your reply as:
Reasoning: [brief reasoning on how to best rewrite and restructure question and answer]
New Question: [New rephrased question]
New Answer: [New rephrased answer]
```

System Prompt ARC

```
You are a helpful assistant. The user will give you a question and answer from the ARC-
Challenge dataset. Rephrase both the question and answer. Make significant changes to used
vocabulary, length and structure.

Format your reply as:
Reasoning: [brief reasoning on how to best rewrite and restructure question and answer]
New Question: [New rephrased question]
New Answer: [New rephrased answer]
```

Figure 3: System prompts used for rephrasing.

User Prompt GSM8K

```
Rewrite the question and answer further such that the background story, names and numbers are
 completely different. Make sure it is difficult to recognize that one is a rewrite of the
other. Use the same reply format.
```

User Prompt TruthfulQA

```
A human could still detect that the new question and answer are based on the original ones.
Make significant changes to the question and change the discussed misconception in order to
make such an observation impossible. Use the same format.
```

User Prompt MMLU

```
A human could still detect that the new question and answer are based on the original ones.
Make very significant changes to the question and answer to make such an observation
completely impossible. Change numbers, background story and all you can change to make this
happen. Use the same format.
```

User Prompt ARC

```
A human could still detect that the new question and answer are based on the original ones.
Make very significant changes to the question and answer to make such an observation
completely impossible. Change numbers, background story and all you can change to make this
happen. Use the same format.
```

Figure 4: User prompts used for further rephrasing of each benchmark.

Table 8: Performance of GPT-2 XL on various benchmarks under contaminated and uncontaminated settings. The metric for all datasets is accuracy in %. C is measured on the contaminated part of the test set, U on the uncontaminated part of the test set.

	Reference		1 Occurrence				5 Occurrences			
			open		evasive		open		evasive	
	C	U	C	U	C	U	C	U	C	U
GSM8K	2.4	2.3	1.8	1.8	1.1	1.7	13.9	4.1	3.7	2.4
TruthfulQA	33.7	33.7	53.9	46.1	39.4	37.7	90.6	53.2	51.0	40.4
MMLU	24.2	25.8	51.6	27.6	25.6	26.4	90.7	27.0	28.0	23.4
ARC	23.9	28.5	54.5	22.5	25.8	25.2	94.8	28.0	27.3	25.7

Table 9: Performance of Mistral 7b on various benchmarks under contaminated and uncontaminated settings. The metric for all datasets is accuracy in %. C is measured on the contaminated part of the test set, U on the uncontaminated part of the test set.

	Reference		1 Occurrence				5 Occurrences			
			open		evasive		open		evasive	
	C	U	C	U	C	U	C	U	C	U
GSM8K	9.1	11.7	33.5	28.5	30.3	22.8	92.7	25.1	48.7	19.8
TruthfulQA	44.1	46.1	79.6	59.1	65.0	55.9	93.3	58.9	55.7	46.8
MMLU	50.2	43.9	81.9	48.6	52.0	46.7	96.7	42.3	52.4	44.7
ARC	61.9	58.8	91.1	66.6	70.6	62.1	99.7	59.3	68.9	58.0

Table 10: Average TPR@1%FPR over the four benchmarks for various sample-level detection methods and models. We compare openly malicious (Op) and evasively malicious (Ev) actors.

	GPT-2 XL				Mistral 7b			
	1 Occ.		5 Occ.		1 Occ.		5 Occ.	
	Op	Ev	Op	Ev	Op	Ev	Op	Ev
Black-Box Baseline	0.2	0.7	5.0	1.3	1.8	1.2	23.9	0.9
Mireshghallah et al. (2022)	2.2	1.9	5.0	2.8	1.2	1.8	7.5	2.2
Carlini et al. (2021)	5.1	1.3	24.0	1.4	2.8	1.0	21.7	1.2
Shi et al. (2023)	7.1	1.6	36.2	1.5	3.8	1.0	26.4	1.3
Yeom et al. (2018)	8.6	1.3	40.6	1.3	3.8	1.1	27.3	1.5

Table 11: Contamination score of Shi (2023) on the contaminated portion of the benchmarks under different settings. A benchmark is considered contaminated when the score is higher than 0.85. We compare openly malicious (Op) and evasively malicious (Ev) actors.

	GPT-2 XL					Mistral 7b				
	Reference	1 Occ.		5 Occ.		Reference	1 Occ.		5 Occ.	
		Op	Ev	Op	Ev		Op	Ev	Op	Ev
GSM8K	0.56	0.98	0.54	1.00	0.51	0.89	1.00	0.92	1.00	0.91
TruthfulQA	0.39	0.58	0.43	0.79	0.45	0.60	0.83	0.65	0.86	0.59
MMLU	0.08	0.08	0.11	0.07	0.15	0.23	0.21	0.34	0.19	0.42
ARC	0.03	0.04	0.04	0.04	0.06	0.10	0.09	0.13	0.11	0.15

Table 12: Accuracy in % on rephrased data for different models. We compare openly malicious (OP) and evasively malicious (EV) actors. We use differently rephrased data for training and testing and only measure performance on the contaminated part.

	GPT-2 XL					MISTRAL 7B				
	REFERENCE	1 OCC.		5 OCC.		REFERENCE	1 OCC.		5 OCC.	
		OP	EV	OP	EV		OP	EV	OP	EV
GSM8K	1.83	2.59	1.83	6.54	3.20	10.96	34.55	30.59	64.08	49.77
TruthfulQA	41.87	65.27	50.74	83.99	57.88	51.23	83.00	67.49	91.87	60.59
MMLU	28.25	37.80	28.46	52.64	31.91	46.75	73.58	46.34	85.77	50.20
ARC	29.73	31.44	24.74	45.02	25.95	60.14	84.19	68.04	88.49	67.87

Table 13: Accuracy in % on various benchmarks using oracle rephrasing for various models. C is measured on contaminated part of the test set, U on uncontaminated part of the test set.

	GPT-2 XL						MISTRAL 7B					
	REFERENCE		1 OCC.		5 OCC.		REFERENCE		1 OCC.		5 OCC.	
	C	U	C	U	C	U	C	U	C	U	C	U
GSM8K	2.13	2.59	1.83	0.91	2.13	2.28	9.89	10.96	24.05	21.00	24.51	18.42
TruthfulQA	36.36	31.38	39.74	35.60	44.16	40.98	43.90	46.14	54.29	51.29	46.75	47.54
MMLU	24.59	25.41	28.86	26.42	27.03	24.19	48.37	45.73	47.97	47.36	47.36	50.00
ARC	20.96	31.39	26.29	27.96	25.26	26.42	62.20	58.49	66.32	57.46	58.76	56.95