Generalization or Memorization: Data Contamination and Trustworthy Evaluation for Large Language Models

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu[†], Mengfei Yang[‡], and Ge Li[*]
Key Laboratory of High Confidence Software Technologies (Peking University),
Ministry of Education; School of Computer Science, Peking University, Beijing, China
[†]Beijing Institute of Control Engineering, [‡]China Academy of Space Technology
{dongyh, jiangxue}@stu.pku.edu.cn, {zhijin, lige}@pku.edu.cn

Abstract

Recent statements about the impressive capabilities of large language models (LLMs) are usually supported by evaluating on open-access benchmarks. Considering the vast size and wide-ranging sources of LLMs' training data, it could explicitly or implicitly include test data, leading to LLMs being more susceptible to data contamination. However, due to the opacity of training data, the black-box access of models, and the rapid growth of synthetic training data, detecting and mitigating data contamination for LLMs faces significant challenges. In this paper, we propose CDD, which stands for **C**ontamination **D**etection via output **D**istribution for LLMs. CDD necessitates only the sampled texts to detect data contamination, by identifying the peakedness of LLM's output distribution. To mitigate the impact of data contamination in evaluation, we also present TED: **T**rustworthy **E**valuation via output **D**istribution, based on the correction of LLM's output distribution. To facilitate this study, we introduce two benchmarks, i.e., DETCON and COMIEVAL, for data contamination detection and contamination mitigation evaluation tasks. Extensive experimental results show that CDD achieves the average relative improvements of 21.8%-30.2% over other contamination detection approaches in terms of Accuracy, F1 Score, and AUC metrics, and can effectively detect implicit contamination. TED substantially mitigates performance improvements up to 66.9% attributed to data contamination across various contamination setups. In real-world applications, we reveal that ChatGPT exhibits a high potential to suffer from data contamination on HumanEval benchmark.[1]

1 Introduction

In recent years, LLMs have revolutionized the fields of natural language processing (NLP), artificial intelligence, and software engineering. To evaluate LLMs' capabilities in various downstream tasks, such as automatic question answering, natural language reasoning, and code generation, people conduct extensive tests for LLMs based on enormous benchmark datasets (Chen et al., 2021; Cobbe et al., 2021). The results indicate that LLMs exhibit superior performance on these tasks. While marveling at the powerful capabilities of LLMs, people usually want to determine whether an LLM's excellent performance is due to the genuine understanding of tasks to achieve generalization, or merely because it has seen the test data to form memorization, i.e., suffering from data contamination.

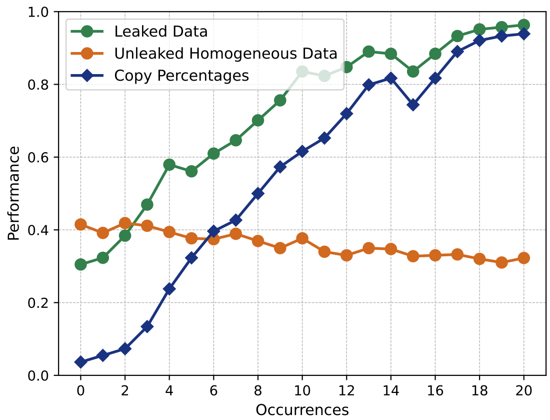

Figure 1: An example of data contamination affecting LLMs' performance, where CodeLlama is fine-tuned on HumanEval (as leaked data) + 50K StarCoder data excluding MBPP (as unleaked homogeneous data).

Data contamination, also known as data leakage, refers to the scenario where the test data has been included in the model's training data (Magar et al., 2022; Dickson, 2023), leading to the model performing exceptionally well on these leaked test data. Owing to the vast size and wide-ranging sources of the pre-trained datasets for LLMs, they are more susceptible to data contamination, which can be primarily categorized into two situations:

[*]Corresponding author.
[1]https://github.com/YihongDong/CDD-TED4LLMs

Findings of the Association for Computational Linguistics ACL 2024, pages 12039–12050

1) For existing benchmark datasets, they are more easily leaked because of massive text quotes, code reuse, and synthetic data in LLMs' training data. 2) For upcoming benchmark datasets, newly constructed test data may already exist in the continuously evolving training data of LLMs since people are usually unaware of the specifics of LLMs' training data. Consequently, it becomes formidable to prevent data contamination for LLMs.

Data contamination exerts a profound and deleterious impact on LLMs (Zhou et al., 2023; Jacovi et al., 2023; Roberts et al., 2023). As shown in Figure 1, with LLMs continuing to learn on contaminated data (i.e., both leaked data and other training data), their performance keeps improving on leaked data but stagnates and even degrades on similar data. This example reflect that data contamination can lead to a substantial overestimation of models' performance, thus affecting the trustworthiness and effectiveness of LLMs in practical applications. Furthermore, data contamination may also conceal the potential flaws of models, presenting major obstacles for people to identify and improve upon LLMs' shortcomings. Therefore, it is crucial for LLMs to detect data contamination and ensure trustworthy evaluation.

Although acknowledged the significance, data contamination detection and trustworthy evaluation for LLMs still persist as open and challenging issues (Yang et al., 2023; Huang et al., 2023). The difficulties of data contamination detection can be essentially attributed to three factors: 1) Opaque Training Data. It is usually non-public and comprehensive, while continuously evolving for new LLMs. 2) Black Box Models. The parameters and output probabilities of LLMs may not be available, such as ChatGPT and GPT-4 (OpenAI, 2023). 3) Proliferation of Synthetic Data. It could implicitly introduce the variants[2] of test data to training data. Further, the evaluation to mitigate the impact of data contamination has hardly been studied.

In this paper, we overcome the preceding challenges by proposing CDD: **C**ontamination **D**etection via output **D**istribution for LLMs. CDD uses the sampled texts to identify the peakedness of LLM's output distribution for data contamination detection. We follow a hypothesis that training is likely to alter the model's output distribution, resulting in a more peaked output distribution for training data, thereby tending the model towards specific outputs on these data. On this basis, we also present TED: **T**rustworthy **E**valuation via output **D**istribution, which is designed to mitigate the impact of data contamination in evaluation by correcting LLM's output distribution.

[2]These variants may include, but are not limited to, translations into other languages, additions of explanations or intermediate processes, and provisions of alternate solutions.

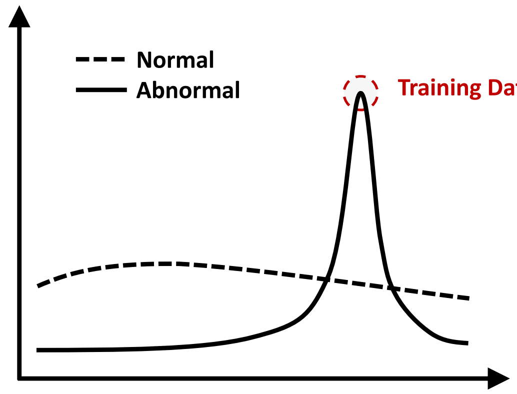

Figure 2: The illustration of LLMs' output distribution.

We construct two new datasets, i.e., DetCon and ComiEval, for data contamination detection and contamination mitigation evaluation tasks, respectively. Experimental results demonstrate that CDD achieves state-of-the-art (SOTA) performance and is also suitable for identifying implicit contamination, i.e., existing the variants of test data in training data. TED successfully mitigates the impact of data contamination in evaluation across various scenarios. Furthermore, we also provide strong evidence that ChatGPT suffers from data contamination on HumanEval dataset.

2 Motivation Example

A powerful LLM that transcends memorization has the capability to generate diverse outputs in response to a given input. Considering the huge vocabulary size of LLMs, which encompasses a good number of tokens with analogous semantics, the output distribution sampled from LLMs ought to not exhibit peakedness. However, when LLMs solely form memorization via training, LLMs are prone to generate outputs that abnormally resemble their training data, as shown in Figure 2. From a statistical perspective, assuming that the average probability of LLM's output tokens is 0.95, the likelihood of sampling two outputs that contain the same 100 consecutive tokens is about $0.005 < 0.01$, which is an extremely improbable event. Therefore, if an LLM consistently outputs some identical or highly similar texts through sampling, it is most

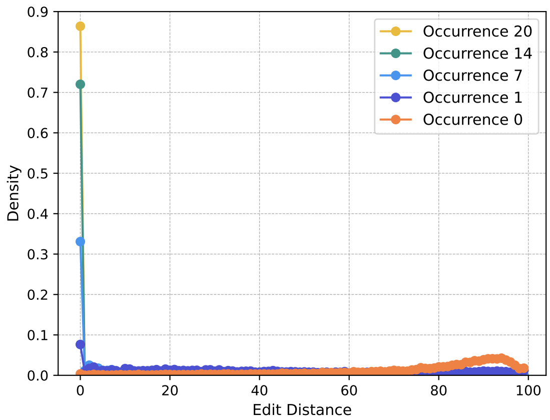

Figure 3: The output distributions of LLMs as modeled by edit distance across varying degrees of data contamination (with the same setting as Figure 1).

likely caused by memorization.

Figure 3 displays an example of how the LLM's output distribution changes as the degree of data contamination varies. We model the LLM's output distribution by computing the edit distances of sampled texts, referred to as edit distance distribution (§ 3.1). As shown in Figure 3, in the absence of data contamination (i.e., occurrence 0), the density of zero edit distance stands at 0.0035, where zero edit distance means that sampled texts exactly match. However, upon the LLM being exposed to the leaked data even once (i.e., occurrence 1) during training, the density of zero edit distance escalates sharply to more than 20 times larger than the original, showing the peakedness. Therefore, the impact of data contamination on the LLM's output distribution is substantial.

In this paper, to the best of our knowledge, we are the first to consider from the standpoint of LLMs' output distribution to address the challenges associated with data contamination detection and contamination mitigation evaluation, employing only the sampled texts without access to the output probability and training data.

3 Methodology

In this section, we first establish the edit distance distribution (§ 3.1), and then on this basis, we design CDD for data contamination detection (§ 3.2) and TED for contamination mitigation evaluation (§ 3.3).

3.1 Edit Distance Distribution

Edit distance (Levenshtein et al., 1966) is a measure of similarity between two strings, which is defined as the minimum number of operations required to transform one string into the other. The operations typically include insertion, deletion, or substitution of a single character.

Considering the generation of LLMs is based on tokens instead of characters, we adopt token-level edit distance in this paper. Given two strings a and b, token-level edit distance is calculated as:

$$\mathrm{ED}(a,b) = \begin{cases} \mathrm{Len}(a) & \text{if } \mathrm{Len}(b)=0, \\ \mathrm{Len}(b) & \text{if } \mathrm{Len}(a)=0, \\ \mathrm{ED}(\mathrm{Tail}(a),\mathrm{Tail}(b)) & \text{if } \mathrm{Head}(a)=\mathrm{Head}(b), \\ 1+\min\begin{cases} \mathrm{ED}(\mathrm{Tail}(a),b) \\ \mathrm{ED}(a,\mathrm{Tail}(b)) \\ \mathrm{ED}(\mathrm{Tail}(a),\mathrm{Tail}(b)) \end{cases} & \text{otherwise}, \end{cases} \tag{1}$$

where $\mathrm{Len}(a)$ means the length of tokenized a, $\mathrm{Head}(a)$ means the first token of tokenized a, $\mathrm{Tail}(a)$ means the string consists of all tokens of tokenized a following $\mathrm{Head}(a)$. We use dynamic programming to speed up calculations and rolling arrays to reduce space overhead.

Given an LLM, we can model its output distribution by computing the edit distances of sampled texts $S = \{s_1, s_2, ..., s_n\}$, where n is the number of samples. Specifically, we define the density function ρ as:

$$\rho(d) = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \mathbb{I}(\mathrm{ED}(s_i,s_j)=d)}{n*(n-1)/2}, \tag{2}$$

where $d \in \mathbb{Z}_{\geq 0}$ and $\mathbb{I}(\cdot)$ is the indicator function that outputs 1 if the condition is true, otherwise 0.

3.2 CDD for Data Contamination Detection

Given a test data $\{x, y\}$ consisting of a prompt x and the corresponding answer y, we aim to detect if this data has been trained by the model $\mathcal{M}$.

We sample S from $\mathcal{M}$ with the input x to calculate ρ. For data contamination detection, the calculation of ρ can be simplified as:

$$\rho'(d) = \frac{\sum_{i=1}^{n} \mathbb{I}(\mathrm{ED}(s_i,y)=d)}{n}. \tag{3}$$

However, $\rho'(d)$ assumes that test data must be explicitly leaked in its original form $\{x, y\}$, and does not take into account the possible implicit contamination of the variant form, i.e., $\{x, \hat{y}\}$.

Through observation, we find that the copy percentage of model outputs increases as the degree of data contamination increases, as shown in Figure 1. Therefore, we approximate y by the model's output texts and finally choose to replace y with the model's greedy search text $s_{t=0}$, which can be easily achieved by setting temperature $t = 0$ when sampling. Thus,

$$\rho^*(d) = \frac{\sum_{i=1}^{n} \mathbb{I}(\mathrm{ED}(s_i, s_{t=0}) = d)}{n}. \quad (4)$$

In this work, we employ $\rho^*(d)$ to measure edit distance distribution by default.

Further, we define the peakedness of edit distance distribution as

$$\mathrm{Peak}(\mathcal{M}; x) = F(d \leq \alpha \cdot l) = \sum_{d=0}^{\alpha \cdot l} \rho^*(d), \quad (5)$$

where F is the cumulative distribution function, $\alpha \in [0, 1]$ is a hyper-parameter to control the similarity, and and l is defined as:

$$l = \max(\{\mathrm{Len}(s) \mid s \in S\}). \quad (6)$$

Through identifying the peakedness, CDD can detect data contamination on test data as:

$$\mathrm{CDD}(\mathcal{M}; x) = \begin{cases} \text{Leaked} & \text{if } \mathrm{Peak}(\mathcal{M}; x) > \xi, \\ \text{Unleaked} & \text{if } \mathrm{Peak}(\mathcal{M}; x) \leq \xi, \end{cases} \quad (7)$$

where $\xi \in [0, 1]$ is hyper-parameter to control the threshold. The pseudocode of CDD for data contamination detection is shown in Algorithm 1.

Algorithm 1 The pseudocode of CDD

Require: LLM $\mathcal{M}$, the prompt of test data x, and hyper-parameter $\alpha = 0.05, \xi = 0.01$.
Ensure: Contamination status cs.
1: Sample S from $\mathcal{M}$ with the input x.
2: Model $\rho^*(d)$ via Eq. (4)
3: Compute $\mathrm{Peak}(\mathcal{M}; x)$ via Eq. (5).
4: Detect cs via Eq (7)
5: **return** cs.

3.3 TED for Contamination Mitigation Evaluation

We achieve contamination mitigation evaluation using TED, which includes two rules to correct the LLM's output distribution, i.e., exclude peakedness and remove duplicates.

1) Exclude Peakedness. We hope to restore the uncontaminated sampling results by excluding the peakedness in the LLM's output distribution, while excluding the greedy text $s_{t=0}$ which is most likely to represent the leaked data potentially memorized by the LLM.

$$S_e = \{s \mid s \in S \wedge \mathrm{ED}(s, s_{t=0}) > \tau\}, \quad (8)$$

where $\tau \in [0, +\infty)$ is a hyper-parameter to control the difference.

2) Remove Duplicates. It aims to remove the duplicate sampling results, especially those differing from $s_{t=0}$, which are also less likely to duplicately occur in the uncontaminated sampling results.

$$S_r = \{s_i | s_i \in S \wedge \forall j < i, s_j \neq s_i\}. \quad (9)$$

In the evaluation phase, an evaluation metric $\mathcal{E}$ using TED to mitigate the impact of data contamination can be defined as:

$$\mathcal{E}_{\mathrm{TED}}(\mathcal{M}; x) \equiv \mathcal{E}_{\mathrm{TED}}(S; x) = \mathcal{E}(S_e \wedge S_r; x), \quad (10)$$

The pseudocode of TED for contamination mitigation evaluation is shown in Algorithm 2.

Algorithm 2 The pseudocode of TED.

Require: LLM $\mathcal{M}$, the prompt of test data x, evaluation metric $\mathcal{E}$, and hyper-parameter $\tau = 2$.
Ensure: Evaluation performance ep.
1: Sample S from $\mathcal{M}$ with the input x.
2: Exclude peakedness to compute S_e via Eq. (8).
3: Remove duplicates to compute S_r via Eq. (9).
4: Obtain ep based on $\mathcal{E}$ via Eq. (10).
5: **return** ep.

4 Experiment

In this section, we first introduce two datasets, DetCon and ComiEval, tailored for the tasks of data contamination detection and contamination mitigation evaluation, respectively (§ 4.1). We then evaluate the efficacy of CDD on DetCon dataset (§ 4.2). Following this, we assess the performance of TED on ComiEval dataset (§ 4.3). Finally, we demonstrate the application results of both CDD and TED in real-world scenarios (§ 4.4).

[3] We rephrase leaked data and each problem in the variant of leaked data has another correct solution different from the original solution, where the majority is generated by ChatGPT and about 10% is generated by ChatGPT-assisted humans.

Table 1: Detailed statistics of simulating different data contamination scenarios of LLMs.

Domain	Leaked Dataset	Base LLMs	Other Training Data	Mixing Ratio	Learning Rate	Occurrences	Contamination Form
Code Generation	HumanEval	{CodeLlama, CodeGen}	StarCoder data	1 : {0, 0.1K, 1K, 10K}	{1e-3, 2e-4, 4e-8}	[0, 20]	{Explicit, Implicit[3]}
Logical Reasoning	GSM8K	{Llama2, Bloom}	RedPajama data				

Table 2: The differences between CDD and other contamination detection approaches, where N-gram and LLM Decontaminator are designed to detect the contamination of training data rather than models.

Approach	Not Need Prob.	Not Need Param.	Not Need Other LLM	Consider Implicit Contamination
N-gram (Brown et al., 2020)	✔	✔	✔	✗
Embedding similarity	✔	✗	✔	✗
Perplexity (Li, 2023)	✗	✔	✔	✗
Min-k% Prob (Shi et al., 2023)	✗	✔	✔	✗
LLM Decontaminator (Yang et al., 2023)	✔	✔	✗	✔
CDD	✔	✔	✔	✔

Table 3: Comparison of CDD and other contamination detection approaches, where † denotes the application of the approach needs additional conditions as shown in Table 2 and the ***bold italic*** indicates the highest value other than CDD, which is also the baseline of the relative improvement.

Approach	DETCON (Code Generation)			Average	DETCON (Logical Reasoning)			Average
	Accuracy	F1 Score	AUC		Accuracy	F1 Score	AUC	
N-gram (char-level)	0.484	0.593	-	0.538	0.564	0.67	-	0.617
N-gram (token-level)	0.541	0.302	-	0.422	0.656	0.498	-	0.577
Embedding similarity†	0.524	0.569	0.571	0.554	0.592	0.645	0.668	***0.635***
Perplexity†	0.513	0.593	0.491	0.532	0.497	0.664	0.699	0.620
Min-k% Prob†	0.563	0.524	0.565	0.550	0.527	0.677	0.698	0.634
LLM Decontaminator†	0.535	0.578	-	***0.556***	0.509	0.433	-	0.471
CDD	**0.715**	**0.694**	**0.761**	**0.724** (↑ 30.2%)	**0.706**	**0.765**	**0.846**	**0.773** (↑ 21.8%)

4.1 Dataset

Considering the absence of datasets for data contamination detection and contamination mitigation evaluation tasks, we dedicate more than 2100 hours to constructing the DETCON and COMIEVAL datasets, utilizing two A6000 GPUs (48GB $\times$ 2).

We simulate data contamination by training LLMs using benchmark data. To cover various scenarios of data contamination, we consider different settings, including two domain benchmarks leaked on four LLMs, two contamination form (i.e. explicit and implicit leaked data), using three different learning rates during training, four mixing ratios of leaked data with other training data, and 21 degrees of contamination (i.e., occurrences). The detailed statistics can be found in Table 1. Due to the high cost of large-scale pre-training, we employ LoRA (Hu et al., 2022) to fine-tune the base models on these various settings. On this basis, we construct the DETCON and COMIEVAL datasets.

DETCON contains 2224 data contamination detection tasks, covering two domains (code generation and logical reasoning) and two contamination forms (explicit and implicit), which need to detect whether a specific LLM has contamination on a particular data. We randomly select the data from the leaked dataset and the LLM from the settings in Table 1, where occurrence 0 refers to 'uncontaminated' and the others denote 'contaminated'.

COMIEVAL contains 560 contamination mitigation evaluation tasks, consisting of a randomly selected contaminated model from Table 1 and the corresponding uncontaminated model, which need to evaluate the performance of the contaminated model and try to mitigate the impact of data contamination to approach the performance of the uncontaminated model.

The detailed statistics and introductions of DETCON and COMIEVAL datasets can be found in Appendix A.

4.2 Data Contamination Detection

Experimental Setup. We compare CDD with baselines, including 1) **N-gram**: We employ widely-used 13-gram for both char-level and token level; 2) **Embedding Similarity**: Use the embed-

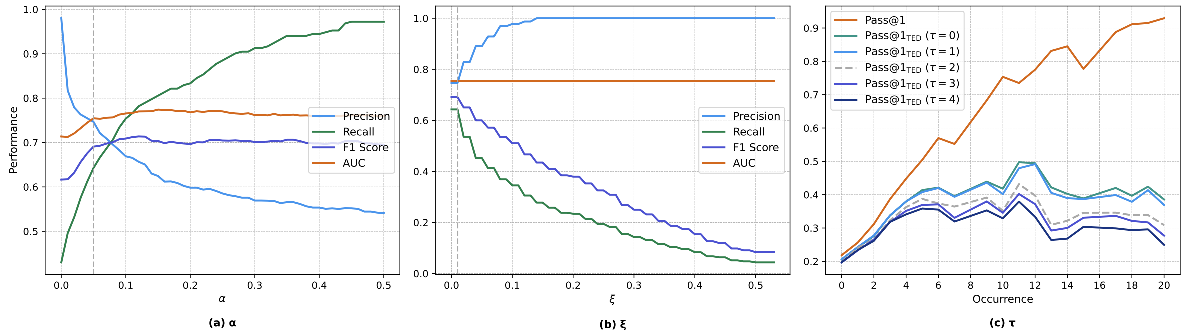

Figure 4: The influence of hyper-parameters, where α and ξ serve for CDD, τ is used for TED, and we use the gray dashed line to represent the employed hyper-parameters.

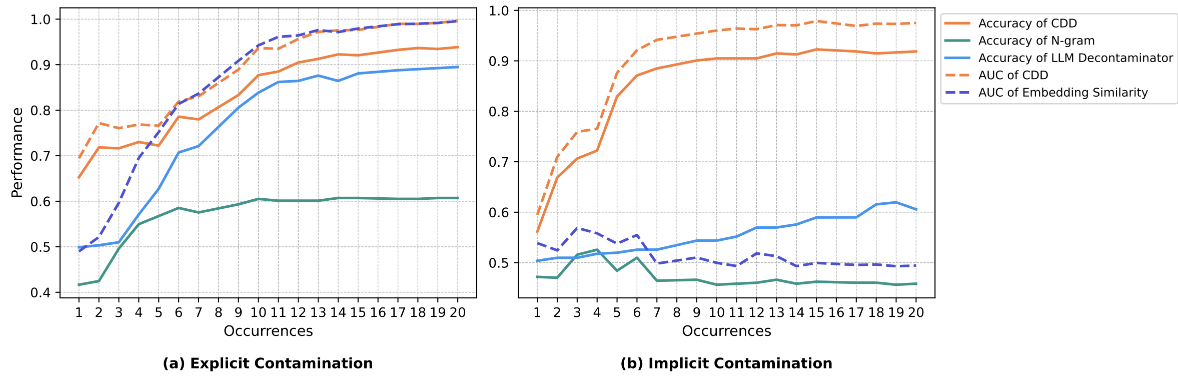

Figure 5: The effectiveness of CDD for data contamination detection in explicit and implicit contamination forms.

ding of the base model to compute similarity; 3) **Perplexity**: Compute the perplexity of the original answer given the prompt; 4) **Min-k% Prob**: Compute the minimum k% probability of the original answer given the prompt, and 5) **LLM Decontaminator**: Use other LLM to determinate the similarity and we employ ChatGPT as this LLM. The differences between CDD and baselines are shown in Table 2. For hyper-parameters, we set $\alpha = 0.05$, $\xi = 0.01$, the cap of l as 100 for CDD by default, and baselines follow the settings in their paper.

The Effect of CDD. As presented in Table 3, compared with other contamination detection approaches, CDD attains SOTA performance in both code generation and logic reasoning domains. CDD exhibits steady improvements across the Accuracy, F1 Score, and AUC metrics, with the average relative improvement ranging between 21.8% and 30.2%. Moreover, the advantage of CDD is that it only requires the sampled texts of LLMs to detect data contamination, without the need for additional conditions in Table 2.

We compare CDD with the two best-performing approaches besides CDD (i.e., Embedding Similarity and LLM Decontaminator) in Table 3, alongside the most commonly used n-gram in two contamination forms, as shown in Figure 5. In the cases of explicit contamination, as the degree of contamination increases, the detection effectiveness across all approaches improves. CDD outperforms the other approaches at lower contamination degrees, which are more challenging to detect. In contrast, in the cases of implicit contamination, CDD alone maintains robust performance, whereas the other approaches encounter significant limitations.

We fix the hyper-parameter α and ξ intuitively for CDD in the experiments. In Figure 4 (a) and (b), we analyze the influence of α and ξ empirically on DetCon dataset by changing itself and fixing another hyper-parameter. The results indicate that there is still room for further improvements with the better hyper-parameter setup of α and ξ.

4.3 Contamination Mitigation Evaluation

Experimental Setup. We evaluate the effectiveness of TED for contamination mitigation in different learning rates, base LLMs, mixing ratios, contamination forms, and occurrences on ComiEval. We set the hyper-parameter $\tau = 2$ and use Pass@1

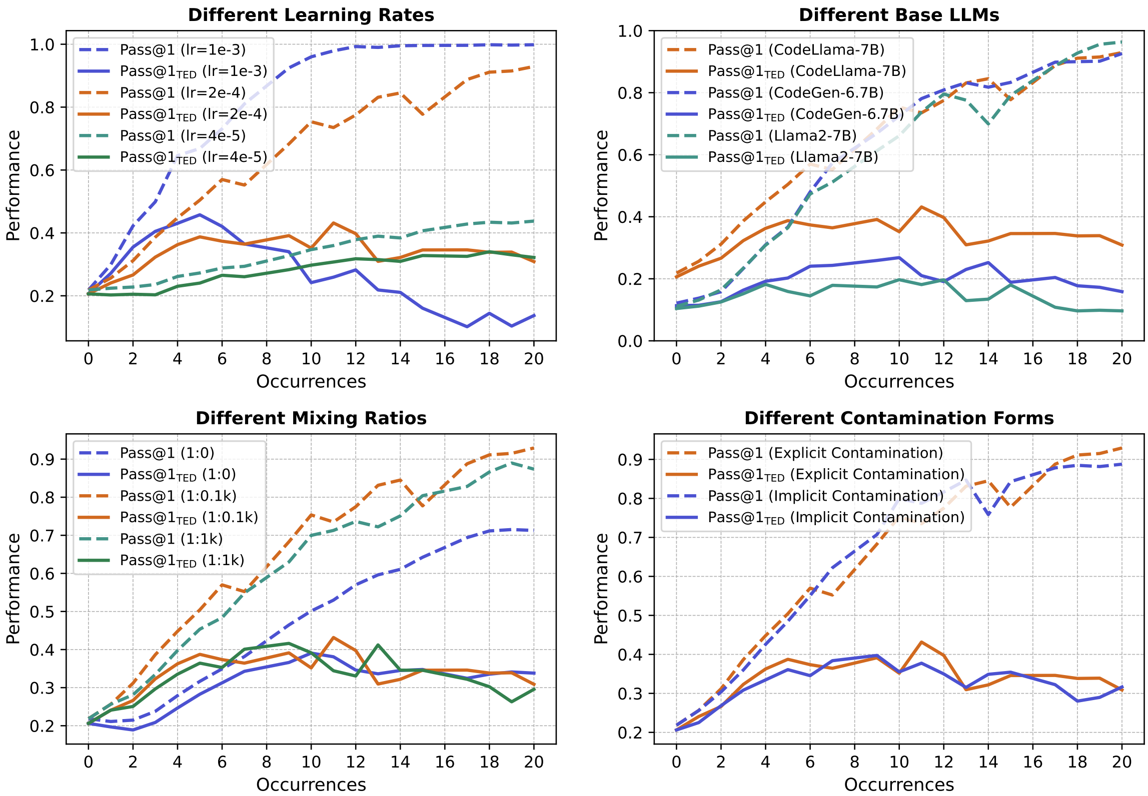

Figure 6: The effect of our TED on model performance as the degree of data contamination increases under different settings. The legend displays the settings for specific data leakage scenarios.

(Chen et al., 2021) as the evaluation metric $\mathcal{E}$.

Effect of TED. TED can steadily mitigate the performance improvements across different settings and occurrences in data contamination scenarios, as shown in Figure 6. Moreover, the advantage of TED is that the performance influence of TED on the uncontaminated model (i.e. 0 occurrences) is small and almost negligible. However, as contamination degrees continue to increase, the performance influence of TED becomes apparent in all of the different settings.

Table 4: Ablation Study of TED, where RD and EP mean the rules of remove duplicates and exclude peakedness in TED, respectively.

Approach	Occurrences				
	0	1	7	14	20
Pass@1	0.219	0.257	0.553	0.846	0.930
+ RD	0.212	0.244	0.486	0.740	0.831 (↓ **10.7%**)
+ EP	0.212	0.242	0.371	0.335	0.320 (↓ **65.5%**)
Pass@1_{TED}	0.209	0.241	0.364	0.321	0.308 (↓ **66.9%**)

We analyze the effects of each component in TED, as shown in Table 4. The main function is provided by the rule of exclude peakedness, followed by the rule of remove duplicates. Both components are beneficial to TED and are also effective when employed alone.

As illustrated in Figure 4 (c), an increase in the hyperparameter τ for TED leads to a more pronounced suppression of performance improvements attributable to data contamination. Meanwhile, it also marginally decreases the performance of the uncontaminated model.

4.4 Real-World Application

Experimental Setup. In real-world applications, we apply CDD and TED for ChatGPT and construct two new datasets to assist evidence: 1) CodeForces2305 comprises 90 of the easiest level programming problems collected from the CodeForces website since May 2023, which is after the most recent update deadline of ChatGPT's training data, i.e., April 2023. 2) HumanEval_R is reconstructed on HumanEval, which replaces its function signature, translates its requirements into German, French, and Chinese, selects different public test cases from the work (Dong et al., 2023a) to prompt, and remains the private test cases for testing. To enhance the detection precision, we set the hyper-parameters α to 0 and ξ to a larger value of 0.2 for CDD. We keep τ at the default value of 2 for TED.

Table 5: Data contamination detection and contamination mitigation evaluation for ChatGPT, where we call ChatGPT's API with the fixed version '1103', Avg. Peak means the average of the peakedness of output distribution computed via Eq. 5, and CR means the ratio of contaminated tasks detected by CDD in the benchmark.

Benchmark	Pass@1	Avg. Peak	CR	Pass@1_{TED}
HumanEval	0.7248	0.2326	0.4147	0.5964
HumanEval_R	0.4684	0.0594	0.1097	0.4171
CodeForces2305	0.0790	0.0063	0	0.0785

Data contamination for ChatGPT. As shown in Table 5, on HumanEval dataset, ChatGPT exhibits a high Avg. Peak and Leak Ratio. Considering the implementation of more stringent α and ξ, it is posited that ChatGPT is likely to suffer from data contamination on HumanEval dataset. This hypothesis is further evidenced through evaluations conducted on HumanEval_R and CodeForces2305 datasets. HumanEval_R indicates their high Avg. Peak and Leak Ratio are not easily attributable to the difficulty of problems. By modifying prompt forms through a process of reconstruction, all of the performance, Avg. Peak, and Leak Ratio of ChatGPT are significantly reduced. On CodeForces2305 dataset, which is unlikely to be involved in data contamination, ChatGPT's performance was markedly lower than anticipated, with the Avg. Peak at less than 0.01 and Leak Ratio of 0. Moreover, TED demonstrates significant effectiveness on both the contaminated HumanEval and HumanEval_R.

5 Related Work

Data contamination detection. The concept of data contamination for LLMs can be derived from the context of GPT-3 (Brown et al., 2020). Due to the vastness of the pre-training corpus of GPT-3, it inevitably overlapped with some evaluation benchmarks. Therefore, GPT-3 adopted 13-gram overlap detection to remove the data in the training set that conflicts with the test set of benchmarks.

Some work (Pan et al., 2020; Zhou et al., 2023; Jacovi et al., 2023; Dodge et al., 2021) exposed the serious consequences of data contamination and urged attention to this problem. However, most currently released LLMs did not open their pre-training corpus, which poses a new challenge for data contamination detection. Recent work tried to detect contamination without access to the pre-training corpus (Oren et al., 2023; Deng et al., 2023; Golchin et al., 2023). Min-k% Prob (Shi et al., 2023) calculated the average of the k% smallest probabilities of generated tokens and considered it as contaminated if it exceeded a certain threshold. The work (Li, 2023) assumed that data leaked into the training set tends to exhibit lower perplexity and utilizes perplexity analysis for detection. However, they often require other model outputs (e.g. probability) in addition to text, presenting challenges in detecting closed-source LLMs like ChatGPT, and they ignore the potential implicit contamination from variants of test data.

Recent investigations (Huang et al., 2023; Yang et al., 2023) have suggested that filtering training data based on n-grams may not effectively address the issue of data contamination, especially concerning semantically equivalent sentence rephrasing. To this end, LLM Decontaminator (Yang et al., 2023) detected the similarity of test data and training data based on other advanced LLMs.

Our work requires only sampled texts to detect LLM's data contamination via output distribution and considers the potential implicit contamination.

Contamination Mitigation Evaluation. To mitigate the impact of data contamination and ensure trustworthy evaluations, several approaches focus on constructing new evaluation benchmarks (Golchin et al., 2023). The work (Zhu et al., 2023) employs an LLM to paraphrase the contaminated dataset for evaluations. However, LLM's synthetic data is widely used for training, which already contains lots of paraphrased data (Yang et al., 2023). The work (Li et al., 2023b) leverages temporal information to construct a benchmark beginning from January 2023. However, building a high-quality benchmark is costly and time-consuming, and unfortunately, the training data deadline for ChatGPT and GPT-4 has been updated from September 2021 to April 2023 and continues to be delayed.

Our work achieves contamination mitigation evaluation from the standard of LLM's output distribution and is orthogonal to the preceding works.

6 Conclusion

In this paper, we have proposed two novel approaches, namely CDD and TED, to deal with data contamination detection and contamination mitigation evaluation for LLMs, considering the LLM's output distribution. We construct two corresponding datasets, i.e., DETCON and COMIEVAL, for these two tasks. Extensive experimental results indicate the superiority and versatility of CDD and TED. Moreover, we also discover that ChatGPT is likely to suffer from data contamination on HumanEval dataset. We hope to shed light on this direction and call more attention to data contamination issues.

7 Limitations

Our work has several limitations, which we aim to address in our future work:

First, the validation of our work is mainly focused on benchmarks for code generation and logical reasoning, which are highly representative and widely adopted. In the future, we will further validate our approaches on other benchmarks.

Second, our approaches require multiple samplings to compute the output distribution, and the more samplings conducted, the better the effect. We can use parallel sampling techniques to speed up sampling, thereby reducing time overhead.

Third, considering the limitation of computational resources, we employ a popular parameter-efficient fine-tuning approach, i.e., LoRA, instead of full-parameter fine-tuning to simulate data contamination for LLMs. In future work, we plan to attempt full-parameter fine-tuning.

Finally, in constructing our datasets, we assume that the four base LLMs used do not suffer from data contamination on the selected benchmarks. However, in reality, these LLMs may have slight data contamination. To completely avoid this issue, it might be necessary to retrain an LLM from scratch on a training set known to be entirely free of test data. However, undertaking such a process would be prohibitively costly.

8 Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grant No.62192730, 62192733, 61832009, 62192731, 62072007, the Key Program of Hubei under Grant JD2023008.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *CoRR*.

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

Together Computer. 2023. Redpajama: An open source recipe to reproduce llama training dataset.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2023. Investigating data contamination in modern benchmarks for large language models. *CoRR*, abs/2311.09783.

Ben Dickson. 2023. Why data contamination is a big issue for llms.

Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *EMNLP (1)*, pages 1286–1305. Association for Computational Linguistics.

Yihong Dong, Jiazheng Ding, Xue Jiang, Zhuo Li, Ge Li, and Zhi Jin. 2023a. Codescore: Evaluating code generation by learning code execution. *CoRR*, abs/2301.09043.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. 2023b. Self-collaboration code generation via chatgpt. *CoRR*, abs/2304.07590.

Yihong Dong, Ge Li, and Zhi Jin. 2023c. CODEP: grammatical seq2seq model for general-purpose code generation. In *ISSTA*.

Shahriar Golchin and Mihai Surdeanu. 2023. Time travel in llms: Tracing data contamination in large language models. *CoRR*, abs/2308.08493.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*. OpenReview.net.

Yiming Huang, Zhenghao Lin, Xiao Liu, Yeyun Gong, Shuai Lu, Fangyu Lei, Yaobo Liang, Yelong Shen, Chen Lin, Nan Duan, and Weizhu Chen. 2023. Competition-level problems are effective LLM evaluators. *CoRR*, abs/2312.02143.

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. 2023. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. In *EMNLP*, pages 5075–5084. Association for Computational Linguistics.

Xue Jiang, Yihong Dong, Lecheng Wang, Qiwei Shang, and Ge Li. 2023. Self-planning code generation with large language model. *CoRR*, abs/2303.06689.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Jia Li, Ge Li, Yunfei Zhao, Yongmin Li, Zhi Jin, Hao Zhu, Huanyu Liu, Kaibo Liu, Lecheng Wang, Zheng Fang, Lanshen Wang, Jiazheng Ding, Xuanming Zhang, Yihong Dong, Yuqi Zhu, Bin Gu, and Mengfei Yang. 2024. Deveval: Evaluating code generation in practical software projects. *CoRR*, abs/2401.06401.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Moustafa-Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023a. Starcoder: may the source be with you! *CoRR*, abs/2305.06161.

Yucheng Li. 2023. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation. *CoRR*, abs/2309.10677.

Yucheng Li, Frank Guerin, and Chenghua Lin. 2023b. Latesteval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. *CoRR*, abs/2312.12343.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *ACL (2)*, pages 157–165. Association for Computational Linguistics.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. Codegen: An open large language model for code with multi-turn program synthesis. In *ICLR*. OpenReview.net.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Yonatan Oren, Nicole Meister, Niladri S. Chatterji, Faisal Ladhak, and Tatsunori B. Hashimoto. 2023. Proving test set contamination in black box language models. *CoRR*, abs/2310.17623.

Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *SP*, pages 1314–1331. IEEE.

Manley Roberts, Himanshu Thakur, Christine Herlihy, Colin White, and Samuel Dooley. 2023. Data contamination through the lens of time. *CoRR*, abs/2310.10628.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. 2022. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100.

Sijie Shen, Xiang Zhu, Yihong Dong, Qizhi Guo, Yankun Zhen, and Ge Li. 2022. Incorporating domain knowledge through task augmentation for front-end javascript code generation. In *ESEC/SIGSOFT FSE*, pages 1533–1543. ACM.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. Detecting pretraining data from large language models. *CoRR*, abs/2310.16789.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

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

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. Don't make your LLM an evaluation benchmark cheater. *CoRR*, abs/2311.01964.

Wenhong Zhu, Hongkun Hao, Zhiwei He, Yunze Song, Yumeng Zhang, Hanxu Hu, Yiran Wei, Rui Wang, and Hongyuan Lu. 2023. CLEAN-EVAL: clean evaluation on contaminated large language models. *CoRR*, abs/2311.09154.

Table 6: The statistics of DETCON and COMIEVAL datasets, where each text is equipped with the probability.

Dataset	Task Nums	Inputs (optional) per task	Outputs per task
DETCON	1112 / 1112	a prompt, the original answer, 51 sampled texts, and the model parameter	'contaminated' / 'uncontaminated'
COMIEVAL	560	leaked dataset, 51 sampled texts of each leaked data, and model parameters	evaluation performance

A Details of Dataset Construction

In this section, we further describes the different data contamination scenarios, as well as how we collect and process data from these scenarios to construct the dataset.

First, we prepare the data and models:

1. **Test Data.** We choose the HumanEval (Chen et al., 2021) dataset for code generation and the GSM8K (Cobbe et al., 2021) dataset for logical reasoning.

2. **LLMs.** For code generation tasks, we use CodeLlama-7B (Rozière et al., 2023) and CodeGen-6.7B (Nijkamp et al., 2023); for logical reasoning tasks, we select Llama2-7B (Touvron et al., 2023) and Bloom-7B (Scao et al., 2022).

3. **Training Data.** Code generation tasks use the training data from StarCoder (Li et al., 2023a), while logical reasoning tasks use RedPajama (Computer, 2023).

Next, we construct the dataset DETCON for data contamination detection, starting with the construction of uncontaminated samples. We directly use the outputs generated by LLMs on the test data, representing uncontaminated data. Then, we construct contaminated samples by simulating different contamination scenarios:

1. **Explicit and Implicit Contamination**. Explicit contamination refers to the direct use of test data for training, while implicit contamination refers to training with variants of the test data.

2. **Proportion in the training data**. We use different amounts of training data mixed with test data to train LLMs. The proportions of the test dataset mixed with training data include 1:0, 1:0.1k, 1:1k, 1:10k.

3. **Different learning rates**. Considering the effect of learning rate on model training, we chose three different learning rates: 1e-3, 2e-4, and 4e-8.

4. **Degree of data contamination**. Training LLMs with contaminated data for more epochs indicates a higher degree of contamination. Epochs range from 0 to 20, where 0 means no training of LLM, indicating no contamination, reserved for constructing uncontaminated samples.

By combining these four different scenarios, we can construct a variety of composite data contamination scenarios. For each piece of test data, we randomly select the results generated by LLMs under one of the contamination scenarios as the contaminated samples. Following the previous works (Jiang et al., 2023; Dong et al., 2023b,c; Li et al., 2024), in generating these samples, we also record the outputs of greedy search with a temperature parameter of 0 (1 sample) and 50 samples obtained by sampling with a temperature of 0.8.

This construction approach aims to comprehensively cover possible data contamination scenarios, ensuring we can accurately assess the performance of LLMs in the face of different types and degrees of data contamination.

Finally, we construct the dataset COMIEVAL for contamination mitigation evaluation. We selected 560 LLMs and their generated outputs from all the constructed contaminated LLMs as the task inputs. Then, we used the performance of the corresponding uncontaminated LLMs on these two test datasets as the target output, serving as the evaluation criterion. Table 6 demonstrates the statistics of DETCON and COMIEVAL, respectively.