Does Data Contamination Detection Work (Well) for LLMs? A Survey and Evaluation on Detection Assumptions

Yujuan Velvin Fu[1], Özlem Uzuner[2], Meliha Yetişgen[1], Fei Xia[1]
[1]University of Washington, [2]George Mason University
{velvinfu, melihay, fxia}@uw.edu
ouzuner@gmu.edu

Abstract

Large language models (LLMs) have demonstrated great performance across various benchmarks, showing potential as general-purpose task solvers. However, as LLMs are typically trained on vast amounts of data, a significant concern in their evaluation is data contamination, where overlap between training data and evaluation datasets inflates performance assessments. Multiple approaches have been developed to identify data contamination. These approaches rely on specific assumptions that may not hold universally across different settings. To bridge this gap, we systematically review 50 papers on data contamination detection, categorize the underlying assumptions, and assess whether they have been rigorously validated. We identify and analyze eight categories of assumptions and test three of them as case studies. Our case studies focus on detecting direct, instance-level data contamination, which is also referred to as Membership Inference Attacks (MIA). Our analysis reveals that MIA approaches based on these three assumptions can have similar performance to random guessing, on datasets used in LLM pretraining, suggesting that current LLMs might learn data distributions rather than memorizing individual instances. Meanwhile, MIA can easily fail when there are data distribution shifts between the seen and unseen instances [1].

1 Introduction

Large language models (LLMs) have achieved remarkable performance across various benchmarks, signaling their potential to revolutionize numerous technical domains as general-purpose problem solvers (Achiam et al., 2023; Meta AI, 2024). However, a key concern in accurately evaluating those LLMs is the possibility of **data contamination**, where the LLM's training data overlaps with the evaluation dataset (Balloccu et al., 2024). Evaluating LLMs on contaminated benchmarks leads to inflated performance assessments (Balloccu et al., 2024; Sainz et al., 2023a; Li and Flanigan, 2024), and creates a misleading perception of their capabilities. Therefore, multiple detection approaches have been developed to identify data contamination in LLMs, and these approaches can also be deployed to identify the use of copyrighted or sensitive content in LLM training (Xu et al., 2024; Meeus et al., 2024b).

All existing approaches for detecting data contamination in language models (LMs) rely on specific assumptions regarding the LMs and datasets, which may not be universally applicable across different settings[2]. While previous surveys have focused on detection and mitigation techniques, to our best knowledge, there is currently no comprehensive analysis that surveys and validates the assumptions underlying these approaches (Xu et al., 2024; Ishihara, 2023; Hu et al., 2022).

To bridge this gap, we (1) systematically review 50 papers on data contamination detection for LMs, (2) present the formal, mathematical definitions for different levels of data contamination, (3) categorize the underlying requirements and assumptions associated with each approach and critically assess whether these assumptions have been rigorously validated, and (4) demonstrate through case studies that some unverified assumptions can be wrong in multiple scenarios.

2 Literature Evaluation

To systematically investigate approaches for data contamination detection, we implement a three-step literature review process with 3 sources of papers: (1) four key survey papers (Xu et al., 2024; Ishihara, 2023; Hu et al., 2022; Deng et al., 2024a)

[1]Links to all relevant papers and the code for the case study are available on our project GitHub: https://github.com/velvinnn/LLM_MIA.

[2]This work surveys data contamination detection approaches for LMs of all sizes, not limited to LLMs.

Findings of the Association for Computational Linguistics: NAACL 2025, pages 5250–5271
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

and papers from the 1st Workshop on Data Contamination at ACL 2024[3]; (2) relevant papers cited by the papers from Source (1); and (3) relevant papers cited by the papers from Source (2).

From the above three sources, we collect 81 relevant papers on data contamination. We additionally filter these papers according to the inclusion criterion: the paper must propose and/or evaluate detection approaches for data contamination in text datasets and LMs. Further more, eight studies solely discussing risks and mitigation strategies for data contamination do not meet the inclusion criteria and are excluded.

Consequently, our review includes a total of 50 papers. Among them, we systematically summarize those detection approaches and present formal mathematical representations for their underlying requirements and assumptions. We then evaluate whether these underlying assumptions are true under different scenarios, as described below.

3 Levels of Data Contamination

Data contamination can occur at instance or dataset levels and the detection approaches for them can be different. To facilitate the discussion, we would like to first provide formal mathematical definitions of data contamination at these levels, based on the descriptive definitions from previous research (Xu et al., 2024; Balloccu et al., 2024; Ishihara, 2023).

3.1 Instance-Level Contamination

In this study, we focus on text datasets and define a language instance x as a sequence of word tokens. Originally, **direct** instance-level contamination is defined as the presence of an instance x within an LM M's training set, D_M, i.e. $x \in D_M$ (Xu et al., 2024). However, LMs often do not publish their exact training corpus, but instead refer to multiple datasets, as subsets of D_M (Zhao et al., 2023). These datasets typically undergo various pre-processing steps, such as de-duplication, filtering, masking, and removing noise. Consequently, LMs are trained on slightly different versions of the same dataset (Palavalli et al., 2024). Meanwhile, there is also **indirect** instance-level contamination from greater variations of the dataset, such as machine paraphrasing (Yang et al., 2023).

To account for such minor differences and indirect contamination, in our Definition D1 below, we introduce a **Binary Indicator Function for Instance-Level Contamination**, $b(x, x')$, which returns *True* (1) if two instances are considered to be the same and *False* (0) otherwise. Researchers can determine what instances are considered to be the same by defining $b(x, x')$ accordingly.

Definition D1. Instance-Level contamination: Let D_M be the training data of an LM M. The binary function $f(M, x)$ is defined as follows:

$$f(M,x) = \begin{cases} 1 & \text{if } \exists x' \in D_M, b(x,x') = 1 \\ 0 & \text{if } \forall x' \in D_M, b(x,x') = 0 \end{cases} \quad (1)$$

We define an instance x to be **seen** by M, or M is **contaminated** by x, iff $f(M, x) = 1$. Conversely, we define an instance x as **clean** or **unseen** by M iff $f(M, x) = 0$.

The detection of instance-level contamination is commonly referred to as **membership inference attack (MIA)**. The goal of MIA is to determine the probability of an instance being used to train an LM, namely, $\widehat{f}(M, x)$ (Hu et al., 2022).

3.2 Dataset-Level Contamination

Prior research implicitly refers to dataset-level contamination at two degrees: partial dataset contamination and full dataset contamination.

Definition D2. Full Dataset Contamination: A dataset D is fully contaminated (**fully seen**) by an LM, if every instance within this dataset is contaminated. Namely, $\forall x \in D, f(M, x) = 1$.

When creating benchmarks for detecting data contamination, previous work typically generates the fully contaminated split. For example, Maini et al. (2024b) created contaminated and clean datasets, respectively from the training and validation splits of the LM's pretraining corpus. Shi et al. (2023) focused on LMs which disclosed that they used Wikipedia event data for training, and created the contaminated dataset from the Wikipedia event data which were published before the LMs' release.

Definition D3. Partial Dataset Contamination: A dataset D is partially contaminated (**paritially seen**) by an LM M, if at least one instance within D is seen. Namely, $\exists x \in D, f(M, x) = 1$.

In practice, especially when reporting contamination from benchmark datasets (Dong et al., 2024) or detecting copyrighted content (Karamolegkou et al., 2023; Chang et al., 2023), people focus more on evaluating partial dataset contamination.

[3] https://conda-workshop.GitHub.io/.

Detection Approach	Require-ments (ID)	Assump-tions (ID)	Detection Rearch	Critiques on Those Approaches
Instance Similarity (9 papers)	Disclose (R1) & Release (R2)	None	Dodge et al. (2021), Elangovan et al. (2021), Li et al. (2024), Riddell et al. (2024), Deng et al. (2024b), Yang et al. (2023), Piktus et al. (2023), Lee et al. (2022), Marone and Van Durme (2023)	
Prob. Analysis (16 papers)	None	Absolute Prob. (A1)	Song and Shmatikov (2019), Shi et al. (2023), Meeus et al. (2024b), Maini et al. (2024b), Wei et al. (2024), Srivastava et al. (2023), Li (2023)	Dekoninck et al. (2024), Duan et al. (2024), Maini et al. (2024b), Cao et al. (2024), Meeus et al. (2024c)
	Perturbed Instance (R3)	Ref. Prob. (A2)	Mattern et al. (2023), Maini et al. (2024b) Oren et al. (2024)	Duan et al. (2024), Maini et al. (2024b), Meeus et al. (2024c)
	Ref. LM (R4)	Ref. Prob. (A3)	Carlini et al. (2021), Maini et al. (2024b), Mireshghallah et al. (2022), Zanella-Béguelin et al. (2020) Meeus et al. (2024a)	Dekoninck et al. (2024), Duan et al. (2024), Cao et al. (2024), Maini et al. (2024b), Meeus et al. (2024c)
	Other	Other	Jagannatha et al. (2021), Zhang et al. (2023)	
Instance Gen. & Instance Select. (20 papers)	None	Verbatim Mem. (A4)	Carlini et al. (2022), Kandpal et al. (2022), Magar and Schwartz (2022), Duarte et al. (2024)*, Tirumala et al. (2022), Schwarzschild et al. (2024), Golchin and Surdeanu (2023a)*	
	Key Info. (R5)	Key Info. Gen. (A5)	Deng et al. (2024b), Ranaldi et al. (2024), Chang et al. (2023), Pan et al. (2020), Carlini et al. (2021), Carlini et al. (2019), Liu et al. (2024), Golchin and Surdeanu (2023b), Golchin and Surdeanu (2023a)	
	None	Gen. Variation (A6)	Dong et al. (2024)	
	Metadata (R6)	Metadata Mem. (A7)	Sainz et al. (2023b)* Karamolegkou et al. (2023)*	Dekoninck et al. (2024)
Answer Mem. (5 papers)	Instance Perturb. (R7)	Answer Change (A8)	Liu et al. (2024)*, Mehrbakhsh et al. (2024)*, Yim et al. (2024)*, Zong et al. (2023)*, Razeghi et al. (2022)*	

Table 1: Existing detection approaches for direct data contamination, their requirements and assumptions, and critiques they received. Some papers cover multiple detection approaches with different assumptions. Most detection methods apply to both instance- and dataset-level contamination, while * denotes those limited to dataset-level contamination. In this study, we show that the underlined assumptions may not be often satisfied.

Definition D4. Unseen/Clean Dataset: A dataset is clean (**unseen**) by an LM, if none of its instances is contaminated. Namely, $\forall x \in D, f(M, x) = 0$.

4 Detection of Direct Data Contamination

Direct data contamination is the most common and well-researched type of data contamination. In this section, we categorize the existing detection approaches, their requirements, assumptions, and the critiques they received (see Table 1). The requirements are defined as the preliminary conditions necessary for conducting certain detection approaches. The assumptions are what the authors of detection approaches assume to be true; the assumptions either are explicitly stated by the authors or can be inferred from the detection approaches.

Most detection methodologies for direct contamination are primarily developed to address instance-level contamination; however, they can be adapted to account for dataset-level contamination. Consequently, unless specified otherwise, this section will concentrate on instance-level contamination.

The performance of a detection method depends on how well its requirements are met and the reliability of its assumptions. Therefore, we group the detection approaches based on their assumptions and requirements.

4.1 Instance Similarity

When D_M is known, detection approaches based on **instance similarity** directly deploy Equation 1, by proposing a similarity function to measure $b(\cdot, \cdot)$

and comparing a new instance with every $x \in D_M$.

Previous research focuses on developing a better or more efficient similarity function. Examples of similarity calculation can be conducted through exact match (Dodge et al., 2021), fuzzy match (Piktus et al., 2023; Lee et al.), automatic NLG evaluation metrics (Elangovan et al., 2021; Deng et al., 2024b), and another LM (Yang et al., 2023). Tools have also been developed to allow efficient search within a large D_M, such as Data Portraits (Marone and Van Durme, 2023) and ROOTS Search Tool (Piktus et al., 2023).

Although this approach does not rely on underlying assumptions, it has two requirements:

Requirement R1. D_M needs to be disclosed.

Requirement R2. D_M must be accessible, which is often hindered by legal, privacy constraints, and expired website links.

Case Study: To examine how often these two requirements are met, we analyzed the top 10 LMs on the Vellum LLM leaderboard [4]. We found that *none* of the LMs fulfilled R1, the most basic requirement, let alone R2 (see Appendix A.3 for details).

4.2 Probability Analysis

When the training dataset D_M is unavailable, but the LM M's output token probabilities are known, probability analysis has been used to detect potential instance-level contamination. We group those detection approaches by their assumptions, and unless specified otherwise, they have no requirements.

4.2.1 Absolute Probability

Given an instance x, probability analysis measures instance-level contamination through $P_M(x)$, the probability of the instance x based on an LM M.

Assumption A1. Seen instances will have higher probabilities than unseen ones, and there exists a threshold, ξ_p, that separates seen instances from unseen ones:

$$P_M(x) \begin{cases} \geq \xi_p & \text{if } f(M,x) = 1 \\ < \xi_p & \text{if } f(M,x) = 0 \end{cases} \quad (2)$$

Previous research measures $P_M(x)$ through perplexity (Carlini et al., 2021; Li, 2023) or approximates it through LM loss (Wei et al., 2024), which can be impacted by the instance domain and simplicity. To improve upon this assumption, Shi et al. (2023) evaluates only the average token probability of top $p\%$ least likely tokens in an instance (**Min p% Token**), assuming that unseen instances contain more low-probability outliers in Wikipedia events and books. Similarly, Song and Shmatikov (2019) assesses probabilities of the k most frequent tokens.

Likewise, Srivastava et al. (2023), Wei et al. (2024), and Meeus et al. (2024b) proposed inserting special strings as watermarks into the training data, using the probability of these watermarks to detect data contamination.

However, Maini et al. (2024b) and Duan et al. (2024) have demonstrated that the perplexity and min top p probabilities are close to random in detecting direct instance-level data contamination across different splits of the Pile dataset. Maini et al. (2024b) suggests that shifts in perplexity and infrequent word probabilities may be attributed to temporal events on platforms like Wikipedia, rather than contamination. Similarly, Cao et al. (2024) highlighted that perplexity and token probability approaches are ineffective for code generation tasks.

4.2.2 Reference Probability by An Instance

Instead of assuming the probabilities of all the seen instances are higher than the probabilities of all the unseen instances, this approach compares the probabilities of similar instances.

Requirement R3. There exists an algorithm which, given an instance x and an LM M, can automatically generate a similar unseen instance, x'.

Assumption A2. If x and x' are similar and M has seen x but not x' , the probability of x should be much higher than that of x' based on M:

$$P_M(x) \begin{cases} \gg P_M(x') & \text{if } f(M,x) = 1 \\ \not\gg P_M(x') & \text{if } f(M,x) = 0 \end{cases} \quad (3)$$

Utilizing this assumption, Mattern et al. (2023) construct the similar, reference instance x' by replacing individual words in x with their synonyms. However, in practice, the observation that $P_M(x) \geq P_M(x')$ might result from replacement with rare words. This assumption has been proven false by Maini et al. (2024b) on different splits of the Pile dataset (Gao et al., 2020). In addition to this synonym-based perturbation, Maini et al. (2024b) demonstrate the ineffectiveness of other perturbation approaches, including white space, characters, random deletion, and case changes.

Another study, Oren et al. (2024), constructs the reference instance by randomly shuffling (exchanging) the order of sentences in the original instance.

[4] https://www.vellum.ai/llm-leaderboard#model-comparison. Accessed on Oct 6, 2024.

They make another assumption of the *exchangeability*, positing that all orderings of all the instances in an exchangeable benchmark should be equally likely if uncontaminated. This assumption might not be valid for coding and reasoning tasks.

4.2.3 Reference Probability by Another LM

This type of approach compares the probability of an instance based on two LMs.

Requirement R4. Given an instance x, we can find another LM M' such that x is unseen by M'.

Assumption A3. If x is seen by M but not M', then $P_M(x)$ should be much higher than $P_{M'}(x)$:

$$P_M(x) \begin{cases} \gg P_{M'}(x) & \text{if } f(M,x)=1 \\ \not\gg P_{M'}(x) & \text{if } f(M,x)=0 \end{cases} \quad (4)$$

Previous research has utilized term frequency (Meeus et al., 2024a), the zlib entropy (Carlini et al., 2021), and another LM (Carlini et al., 2021; Mireshghallah et al., 2022) as the reference model. However, Maini et al. (2024b) and Duan et al. (2024) have demonstrated that those reference models perform close to random guessing across various domains. Cao et al. (2024) also show the zlib entropy does not work for code generation tasks. Instead of using reference probabilities at the sentence level, Zanella-Béguelin et al. (2020) deploy a reference model for both individual token probability and its probability rank within the vocabulary.

4.3 Instance Generation and Instance Selection

In this section, we investigate underlying requirements and assumptions for detection approaches based on instance generation and instance selection.

Instance generation detects contamination by treating x as a prefix-suffix pair, $x = (x_p, x_s)$. These approaches evaluate the LM M's generated output, $M(x_p)$, conditioned on x_p. If $M(x_p)$ is similar or identical to x_s, x will be predicted as seen. Based on this core intuition, instance generation approaches vary in their assumptions regarding input-output pairs and language generation approaches. Unless specified otherwise, those approaches below focus on instance generation.

For instance selection, instead of directly generating answers, the LM is tasked with selecting the most likely x_s from a set of candidate options in a multi-choice format. However, detection approaches relying on instance selection face a fundamental limitation: even if x is unseen, the LM might still choose the correct x_s by accident. Consequently, these approaches are generally not employed to detect instance-level contamination but rather to assess the probability of full dataset contamination.

4.3.1 Verbatim Memorization

This type of approach assumes LMs can memorize their training data, to certain extent.

Assumption A4. Given an prefix-suffix pair $x = (x_p, x_s)$, if x has been seen by an LM M, x_s can be generated (memorized) by M through greedy decoding, when given the input x_p.

$$M_g(x_p) \begin{cases} = x_s & \text{if} f(M,x)=1 \\ \neq x_s & \text{if} f(M,x)=0 \end{cases} \quad (5)$$

Duarte et al. (2024) and Golchin and Surdeanu (2023a) define x_s as sentences or passages, and create similar instances to x_s by paraphrasing x_s using another LM. They use instance selection, and assume that the contaminated LM will be more likely to select the verbatim option.

However, instance-level contamination does not always lead to verbatim memorization. Utilizing instance generation, Kandpal et al. (2022), Carlini et al. (2019), Carlini et al. (2022), and Tirumala et al. (2022) demonstrate that verbatim memorization requires repeated exposures to this instance x during training, and a larger LM and longer input length x_p can result in better memorization. Schwarzschild et al. (2024) used the minimum length of x_p needed to generate the desired output x_s to define the degree of memorization.

Similarly, Kandpal et al. (2022) and Carlini et al. (2021) study a relaxed version of this assumption, where the LM can generate x_s through different sampling strategies in decoding, such as top-*k* or top-*p* (Nucleus) sampling (Holtzman et al., 2020). They reach a similar conclusion that data contamination does not necessarily lead to memorization.

4.3.2 Key Information Generation

This type of approach assumes that, if an LM has seen an instance, it can generate x's key information based on its context.

Requirement R5. An instance x can be paraphrased into a slot-filling, context-key pair, $x = (x_c, x_k)$. The key x_k is usually a representative sub-span of x, such as dates and names. The rest tokens in x compose the context, x_c.

Assumption A5. If x is seen, M will be able to produce similar output to x_k when given x_c.

$$S(M(x_c), x_k) \begin{cases} \geq \tau_s & \text{if } f(M, x) = 1 \\ < \tau_s & \text{if } f(M, x) = 0 \end{cases} \quad (6)$$

Here, $M(x_c)$ denotes the output of the LM M through a certain decoding method. $S(\cdot, \cdot)$ is a text similarity function, and τ_s is the contamination threshold. One can use the similarity functions described in Section 4.1.

Leveraging this assumption, prior studies have masked key information within specific datasets, including input questions in NLP benchmarks (Deng et al., 2024b; Golchin and Surdeanu, 2023b; Liu et al., 2024), column names in SQL code generation questions (Ranaldi et al., 2024), character names in books (Chang et al., 2023), and labels in NLI and SST tasks (Magar and Schwartz, 2022).

4.3.3 Generation Variation

This type of approach explores how an LM's outputs vary if it has seen an instance during training.

Assumption A6. Suppose an instance x can be represented as a prefix-suffix pair, $x = (x_p, x_s)$. If an LM M has seen x, then given x_p, M will generate something identical or similar to x_s under different sampling strategies:

$$\text{Var}(\{M.(x_p)\}) \begin{cases} < \xi_v & \text{if } f(M, x_p) = 1 \\ \geq \xi_v & \text{if } f(M, x_p) = 0 \end{cases} \quad (7)$$

where $Var(\{M.(x_p)\})$ measures the variations of outputs from M under diverse, different sampling strategies when given x_p; ξ_v is a threshold, based on the type of input x_p and sampling strategies.

Dong et al. (2024) defines the metric $Var(\cdot)$ as 'Contamination Detection via output Distribution' (CDD), and utilizes this assumption to detect memorization in coding and reasoning benchmarks. However, this assumption can lead to false positives for other tasks, such as multiple choices, where the output is more constrained and has less variation.

4.3.4 Metadata-based Memorization

This type of approach determines whether an LM has seen a dataset D by using D's metadata.

Requirement R6. Given a dataset D, we can construct an input prompt x_m including D's metadata m, such as dataset name, split, and format.

Assumption A7. If an LM M has seen a dataset D, when given D's metadata, M is able to generate instances that are very similar to some $x \in D$.

$$\begin{cases} \exists x \in D, S(M(x_m), x) \geq \tau_m & \text{if } D \text{ is seen} \\ \forall x \in D, S(M(x_m), x) < \tau_m & \text{if } D \text{ is unseen} \end{cases} \quad (8)$$

Here, $M(x_m)$ is the set of instances that M generates when given D's metadata m; $S(M(x_m), x)$ represents the highest similarity between x and any instance x' as a subsequence of $M(x_m)$; τ_m is the contamination threshold for $S(\cdot, \cdot)$.

Sainz et al. (2023b) and Golchin and Surdeanu (2023b) utilized this assumption to demonstrate that OpenAI systems memorized many instances from widely used benchmarks. However, this approach can have false negatives if the LM's training phase does not preserve the linkage between D's metadata and instances (Dekoninck et al., 2024).

4.4 Answer Memorization

Answer memorization is usually conducted at the dataset level. It introduces perturbations to the original dataset, measures the LM's performance change, and aims to detect whether the LM's high performance is due to memorizing its answer.

Requirement R7. Given an LM M and an evaluation dataset D, one can generate a similar dataset D' that is unseen by M, by modifying every $x \in D$.

Assumption A8. Suppose datasets D and D' are similar and an LM M has seen D but not D', M's performance on D (Eval(M, D)) will be much higher than its performance on D' (Eval(M, D')).

$$\text{Eval}(M, D) \begin{cases} \gg \text{Eval}(M, D') & \text{if } D \text{ is seen} \\ \not\gg \text{Eval}(M, D') & \text{if } D \text{ is unseen} \end{cases} \quad (9)$$

Previous research evaluates answer memorization in multiple-choice (MC) tasks by introducing variations such as altering numbers in mathematical tasks (Mehrbakhsh et al., 2024), changing the order of MC options, etc. (Yim et al., 2024; Zong et al., 2023). Razeghi et al. (2022) show that multiple LMs perform better on numerical reasoning tasks involving frequently occurring numerical variables in their pretraining data. Similar to this assumption, Liu et al. (2024) detects if the LM can still predict the correct answer, after removing all MC options.

5 Other Types of Contamination

Besides direct data contamination, previous research also investigates indirect data contamination (6 papers) and task contamination (5 papers).

5.1 Indirect Data Contamination

Indirect data contamination occurs when an instance x is not seen by an LM M, but something (x') derived from x is. For instance, x' can be a paraphrase or a summary of x (Yang et al., 2023).

Indirect data contamination is often hard to track and trace (Balloccu et al., 2024). For example, OpenAI uses online user conversations for training, which could include variations of benchmark datasets[5]. Another example involves knowledge distillation, where an LM utilizes instances x_k generated by another LM M' during training, and these instances x_k may resemble instances from the training set $D_{M'}$ of M' (Veselovsky et al., 2023).

5.1.1 Detection Approaches for Indirect Data Contamination

Compared to direct contamination, indirect data contamination is much more challenging to detect. Dekoninck et al. (2024) and Cao et al. (2024) show that many probability-based detection approaches are ineffective for indirect data contamination.

However, prior research showed that three approaches may still be applicable: (1) the instance similarity measured by another LM (Yang et al., 2023), (2) the CDD metric (Dong et al., 2024), which leverages Assumption A6 by measuring output variations rather than directly comparing with original instances, and (3) directly tracking the disclosed usage of datasets. For example, Balloccu et al. (2024) reviewed the datasets evaluated using OpenAI APIs.

5.2 Task Contamination

Task contamination occurs when any instance of the same task is seen by an LM (Li and Flanigan, 2024). Detecting task contamination is crucial for assessing an LM's generalizability to unseen tasks (Chung et al., 2024). Tasks can include applications such as machine translation, summarization, and mathematical calculation. Task contamination is a broader concept than data contamination: if some labeled instances from a dataset are seen by an LM, the associated task is contaminated, but task contamination doesn't always imply the dataset has been seen.

Task contamination generally evaluates an LM's performance on a particular task at the dataset level. The idea is that if an LM has previously seen the task, its performance will be much higher compared to unseen tasks of similar difficulty.

For example, as noted by Aiyappa et al. (2023), LLMs show improved performance on the same benchmark after model updates, which may be influenced by data contamination during LLMs' continuous training. Ranaldi et al. (2024) and Li and Flanigan (2024) also find that OpenAI models perform significantly better on benchmarks released before the model's release than on those released later, when task difficulty is controlled or performance is normalized using a baseline model. To ensure fair comparisons across tasks, Li and Flanigan (2024) control task difficulty using a baseline system. However, Cao et al. (2024) also note that LMs do not necessarily perform worse on more recent code generation benchmarks.

6 Case Study

Besides the case study in Section 4.1, we aim to evaluate whether the assumptions outlined in Table 1 are universally applicable across different domains, for direct and instance-level MIA.

6.1 Assumptions to Evaluate

As shown in Table 1, prior research has verified that 4 out of 8 assumptions can fail under certain conditions. Meanwhile, some assumptions have specific requirements, and their applicability depends on how well these requirements are met. Therefore, we focus on two unverified assumptions that have no such requirements for evaluation, limiting confounding factors and deferring the testing of other assumptions to future studies. We also validate one verified assumption (Assumption A1) to confirm the consistency of our findings with prior research.

Assumption A1: Absolute Probability. In the assumption that seen instances will have a lower perplexity and fewer low-likely (outlier) tokens, we measure perplexity by an instance's first k tokens (**PPL_**k) (Carlini et al., 2021); **Min** p**% Token** by the average token probabilities among $p\%$ least likely tokens (Shi et al., 2023).

Assumption A4: Verbatim Memorization. We expand this assumption from the instance level to the token level, assuming an LM will memorize some tokens in seen instances. We measure the percentage of tokens in an instance ranked as the k most likely in casual language modeling (**Mem** k). The k value of 1 represents greedy decoding, and larger than 1 simulates the decoding with top

[5] https://help.openai.com/en/articles/5722486-how-your-data-is-used-to-improve-model-performance. Accessed on Oct 6, 2024.

LM	# Params	Training Phase	# Epochs	Trainset Size	Batch Size	Seen & Unseen Datasets Used in Our Case Study
Pythia (Biderman et al., 2023)	70M - 12B	Pretraining	≈ 1.5	825 GiB	2M	Pile (Gao et al., 2020)
OLMo-2 (OLMo et al., 2024)	7B		≈ 2	22.4 TB	4M	AlgebraicStack (Azerbayev et al., 2023)
Zephyr-7B-β(*Tunstallet al.*, 2023)	7B	Supervised Fine-tuning	1-3	9.3 GB	512	UltraChat (Ding et al., 2023)
BioMistral-NLU (Fu et al., 2025)	7B		2	3.6 GB	64	Medical-NLU (Fu et al., 2025)

Table 2: LMs and datasets used in the case study. Except for the UltraChat dataset, each dataset contains multiple subsets from different domains. The trainset refers to the whole trainset used in each LM's corresponding training phase, as described in their original paper, which is a superset of seen & unseen datasets used in our case study.

Training Phase		Pretaining				Supervised Fine-tuning		
Model		Pythia-6.9B		OLMO-2-7B		Zephyr-7B-β	BioMistral-NLU-7B	
Assumptions & Metric		Youtube-Subtitles	ArXiv	Github-Coq	Github-Isabelle	Ultra Chat	DC-MTSample	RE-2012temp
A1	PPL_50	50.7	50.7	47.1	50.9	55.3	51.9	**62.5**
	PPL_100	50.4	51.1	48.2	53.1	59.1	60.0	**95.3**
	PPL_200	49.6	50.9	48.5	51.6	**60.1**	58.9	**99.4**
	Min 5% token	48.5	51.3	49.4	54.0	**63.6**	47.4	**92.9**
	Min 15% token	48.6	51.3	49.3	53.6	**63.0**	51.7	**93.3**
	Min 25% token	48.5	51.4	49.3	53.1	**61.5**	53.3	**93.4**
A4	Mem 5	49.1	52.7	52.1	48.7	52.2	47.9	41.4
	Mem 15	48.6	51.9	50.6	58.6	53.1	49.1	45.2
	Mem 25	48.2	51.6	49.4	59.2	53.1	50.3	48.9
A6	Entropy 5	49.3	52.0	47.2	52.3	54.3	55.4	**93.2**
	Entropy 15	49.0	52.0	47.8	52.1	54.1	54.8	**93.1**
	Entropy 25	48.9	51.9	48.6	52.5	54.0	54.1	**93.1**
Average AUC		49.0	51.8	49.0	53.3	55.9	52.0	**74.1**
PPL	Seen	13.2+-16.0	7.9+-3.7	10.5+-8.1	8.4+-5.2	5.3+-4.6	1.7+-0.2	1.4+-0.1
	Unseen	12.7+-10.6	8.0+-3.6	9.9+-7.2	9.2+-6.5	6.2+-4.1	1.8+-0.2	3.0+-1.6

Table 3: Average MIA AUC for different LMs. For LMs evaluated on multiple subsets (domains) of the same dataset, we present the results from the subsets with the lowest and highest average AUC. The last two rows, marked as 'PPL_200', represent the average perplexity $\pm$ STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

k token sampling.

Assumption A6: Generation Variation. We evaluate the assumption that, given a seen prefix sequence, the LM exhibits less variation (i.e., higher certainty) in predicting the next token under different token sampling strategies. Since lower entropy indicates greater certainty, we measure entropy over the top k most likely tokens (**Entropy** k) (see Appendix A.4 for details).

6.2 Experiment Design

To enhance the generalizability of our results, we evaluate these assumptions using four different types of LLMs, with datasets used in different training phases, shown in Table 2. We also investigate the impact of model size on MIA performance, using seven Pythia models with parameter sizes ranging from 70M to 13B.

Except for the UltraChat dataset, each dataset consists of multiple smaller subsets from different domains. For our experiments, we randomly sample 9 subsets from each, and consider each subset as an individual dataset. To minimize distribution shifts between seen and unseen datasets, we randomly select 1,000 instances from the training split (seen) and 1,000 instances from the test split (unseen) within each dataset. If a test split is unavailable, we sample from the validation split. If there are fewer than 1,000 unseen instances, we use the entire test split, ensuring that each split contains at least 100 instances.

Following prior work (Shi et al., 2023), we evaluate MIA performance using the area under the ROC curve (AUC) at the instance level, representing the probability that a seen instance has a better score (higher or lower) than an unseen instance (Google).

6.3 Results

6.3.1 Within-Domain MIA

Table 3 shows the AUC for each MIA method across representative subsets (domains) of each dataset. Complete results for all datasets and domains are available in Appendix A.5.1.

On pretraining datasets, all metrics perform close to random guessing, with AUC close to 50. We also observed the same pattern with different

sizes of Pythia LMs. Our results for Assumption A1 are consistent with critiques they received (see Section 4.2.1). We suspect that during pretraining, LMs are more likely to learn underlying data distributions, instead of memorizing specific instances.

On fine-tuning datasets, we observed a great variation in MIA AUC across domains. The best-performing metric, PPL_200 on the RE-2012temp dataset, can have an AUC as high as 99.4. This suggests that data contamination from memorizing training instances remains a risk during the fine-tuning phase. Overall, the performance of the perplexity-based metric improves as the number of tokens increases. This trend is linked to the fine-tuning process, where tokens at the beginning of the training instance serve as input prompts but are not explicitly learned during training.

6.3.2 Cross-Domains MIA with Data Distribution Shifts

Within the same domain, the similar average PPL between seen and unseen instances indicates that they have similar underlying distributions, but also a high variation (STD). However, PPL differs a lot across domains. We therefore examine the impact of distribution shifts from different domains on the MIA performance, with the scenario where seen and unseen instances are from different domains.

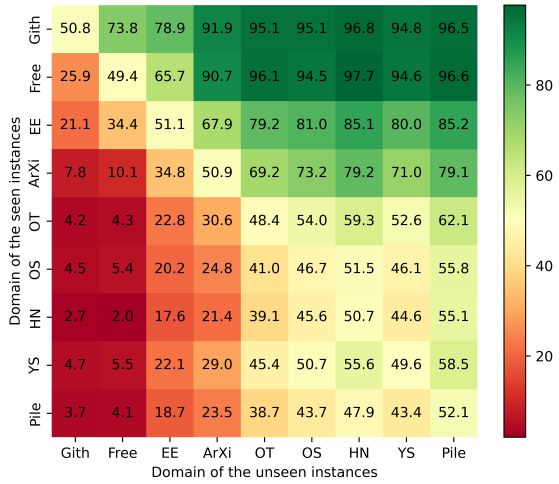

Figure 1: Average MIA AUC for the Pythia-6.9b model with PPL_200, when the seen and unseen instances are from different domains. The abbreviations represent the domains in Table 12 in the Appendix A.5.1.

We present the AUC with PPL_200 in Figure 1. The AUC is high in the top-right corner when seen instances are from a domain with lower average PPLs and unseen instances are from a domain with higher average PPLs. Conversely, the bottom-left corner has low AUCs. This indicates that the accuracy of PPL_200-based MIA highly depends on the domain difference, instead of the seen vs. unseen distinction. More information about the PPL_200 distribution within and across domains is in Appendix A.5.2. A similar trend is observed with other metrics (see Appendix A.5.2 & A.5.3).

7 Discussion

In our case study, we observed the evaluated MIA approaches perform well only on certain domains of datasets used during the fine-tuning phase, but not during the pretraining phase. This discrepancy may be attributed to the significantly larger dataset and batch sizes employed in the pretraining phase.

Together, our case study and prior research show that 6 out of the 8 assumptions listed in Table 1 can often be invalid under certain conditions. The other two unverified assumptions, key information memorization (A5) and answer change due to memorization (A8), depend on very specific requirements, complicating their evaluation. Overall, our findings suggest that MIA remains a challenging task.

The limited effectiveness of MIA in pretraining phases suggests detecting data contamination in benchmarks remains an important challenge for LLM evaluation. While poor MIA performance may indicate a lack of direct instance memorization, models could still learn underlying distributional patterns of benchmark data, enabling artificially high performance on the benchmark datasets (Dekoninck et al., 2024). On the other hand, privacy and copyright risks persist, as LLMs might learn from sensitive/proprietary data (e.g., patented concepts) without triggering MIA alarms.

8 Conclusion

In this study, we present a comprehensive survey of 50 studies focused on data contamination detection and their underlying assumptions. Our theoretical analysis reveals that these assumptions may not apply consistently across different contexts.

Our case studies showed that 3 out of the 8 assumptions are not universally applicable across all training phases and dataset domains, especially for datasets used in the pretraining stage. Our cross-domain MIA experiments additionally show that many assumptions measure an LM's goodness of fit, which is not necessarily the result of instance memorization due to data contamination. Thus, detecting data contamination remains challenging.

9 Limitations

In this study, we reviewed 50 papers on data contamination detection, but there may be additional relevant studies not captured by our collection methods. We primarily focus on data contamination detection approaches for English LMs. Other detection approaches for non-English LMs, data modalities beyond text, and other machine learning techniques may exist and could potentially be transferable to English LMs.

Several factors influence an LM's ability to memorize an instance. The relationship between MIA performance and the training dataset (e.g., domain, size, batch size), as observed in our case study, may vary for other LMs and their respective datasets.

10 Ethical Considerations

In this work, we employed multiple LMs and their training sets, which may contain sensitive and/or identifiable information. For example, the Pile (Gao et al., 2020) includes content crawled from the Internet. The BioMistral-NLU-7B and its training set contain datasets derived from de-identified clinical notes (Fu et al., 2025). Therefore, we only downloaded the necessary instances and published the numerical results from our experiments. In our project GitHub, we only release our code for data sampling and MIA approaches to ensure reproducibility; we do not publish any actual data instances or LMs. We recommend the community to check the corresponding regulations before deploying the datasets and LMs for other purposes.

11 Acknowledgements

This work was supported by the National Institutes of Health (NIH)—National Cancer Institute (Grant Nos. 1R01CA248422-01A1), National Library of Medicine (Grant No. 2R15LM01320902A1), and National Center for Advancing Translational Sciences of the National Institutes of Health (Grant No. UL1TR002319). The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

We thank Chenxi Li, Tianmai Zhang, and the anonymous reviewers for their feedback during the paper revision.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rachith Aiyappa, Jisun An, Haewoon Kwak, and Yong-yeol Ahn. 2023. Can we trust the evaluation on ChatGPT? In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 47–54, Toronto, Canada. Association for Computational Linguistics.

Anthropic. 2024a. Claude 3 haiku: our fastest model yet. Accessed on Oct 6, 2024.

Anthropic. 2024b. Introducing claude 3.5 sonnet. Accessed on Oct 6, 2024.

Anthropic. 2024c. Introducing the next generation of claude. Accessed on Oct 6, 2024.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *Preprint*, arXiv:2310.10631.

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

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, pages 2397–2430.

Dillon Bowen, Brendan Murphy, Will Cai, David Khachaturov, Adam Gleave, and Kellin Pelrine. 2024. Scaling laws for data poisoning in llms. *Preprint*, arXiv:2408.02946.

Jialun Cao, Wuqi Zhang, and Shing-Chi Cheung. 2024. Concerned with data contamination? assessing countermeasures in code language model. *arXiv preprint arXiv:2403.16898*.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, Santa Clara, CA. USENIX Association.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Kent Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023. Speak, memory: An archaeology of books known to chatgpt/gpt-4. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7312–7327.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Jasper Dekoninck, Mark Niklas Müller, Maximilian Baader, Marc Fischer, and Martin Vechev. 2024. Evading data contamination detection for language models is (too) easy.

Chunyuan Deng, Yilun Zhao, Yuzhao Heng, Yitong Li, Jiannan Cao, Xiangru Tang, and Arman Cohan. 2024a. Unveiling the spectrum of data contamination in language model: A survey from detection to remediation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16078–16092, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024b. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8698–8711.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Jesse Dodge, Maarten Sap, Ana Marasović, 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 *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. *arXiv preprint arXiv:2402.15938*.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do membership inference attacks work on large language models? In *Conference on Language Modeling (COLM)*.

André Vicente Duarte, Xuandong Zhao, Arlindo L. Oliveira, and Lei Li. 2024. DE-COP: Detecting copyrighted content in language models training data. In *Forty-first International Conference on Machine Learning*.

Aparna Elangovan, Jiayuan He, and Karin Verspoor. 2021. Memorization vs. generalization: Quantifying data leakage in nlp performance evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1325–1335.

Yujuan Velvin Fu, Giridhar Kaushik Ramachandran, Namu Park, Kevin Lybarger, Fei Xia, Ozlem Uzuner, and Meliha Yetisgen. 2025. Biomistral-nlu: Towards more generalizable medical language understanding through instruction tuning. *AMIA 2025 Informatics Summit*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Shahriar Golchin and Mihai Surdeanu. 2023a. Data contamination quiz: A tool to detect and estimate contamination in large language models. *CoRR*, abs/2311.06233.

Shahriar Golchin and Mihai Surdeanu. 2023b. Data contamination quiz: A tool to detect and estimate contamination in large language models. *CoRR*, abs/2311.06233.

Machine Learning Education Google. AUC (area under the roc curve). Accessed: 2025-02-10.

Jacob Haimes, Cenny Wenner, Kunvar Thaman, Vassil Tashev, Clement Neo, Esben Kran, and Jason Schreiber. 2024. Benchmark inflation: Revealing llm performance gaps using retro-holdouts. *arXiv preprint arXiv:2410.09247*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37.

Shotaro Ishihara. 2023. Training data extraction from pre-trained language models: A survey. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 260–275.

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 *Proceedings of*

the 2023 Conference on Empirical Methods in Natural Language Processing, pages 5075–5084, Singapore. Association for Computational Linguistics.

Abhyuday Jagannatha, Bhanu Pratap Singh Rawat, and Hong Yu. 2021. Membership inference attack susceptibility of clinical language models. *arXiv preprint arXiv:2104.08305*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7403–7412, Singapore. Association for Computational Linguistics.

Wojciech Kusa, Harrisen Scells, Moritz Staudinger, and Allan Hanbury. 2024. Leveraging cochrane systematic literature reviews for prospective evaluation of large language models. *The 1st Workshop on Data Contamination (CONDA)*.

Ariel Lee, Cole Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445.

Changmao Li and Jeffrey Flanigan. 2024. Task contamination: Language models may not be few-shot anymore. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18471–18480.

Yucheng Li. 2023. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation. *arXiv preprint arXiv:2309.10677*.

Yucheng Li, Yunhao Guo, Frank Guerin, and Chenghua Lin. 2024. An open-source data contamination report for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 528–541.

Chuang Liu, Renren Jin, Mark Steedman, and Deyi Xiong. 2024. Evaluating Chinese large language models on discipline knowledge acquisition via memorization and robustness assessment. In *Proceedings of the 1st Workshop on Data Contamination (CONDA)*, pages 1–12, Bangkok, Thailand. Association for Computational Linguistics.

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

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. 2024a. Tofu: A task of fictitious unlearning for llms.

Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. 2024b. Llm dataset inference: Did you train on my dataset? *The 1st Workshop on Data Contamination (CONDA)*.

Marc Marone and Benjamin Van Durme. 2023. Data portraits: Recording foundation model training data. In *Advances in Neural Information Processing Systems*, volume 36, pages 15121–15135. Curran Associates, Inc.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343.

R Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. 2023. How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven. *Transactions of the Association for Computational Linguistics*, 11:652–670.

Matthieu Meeus, Shubham Jain, Marek Rei, and Yves-Alexandre de Montjoye. 2024a. Did the neurons read your book? document-level membership inference for large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2369–2385.

Matthieu Meeus, Igor Shilov, Manuel Faysse, and Yves-Alexandre de Montjoye. 2024b. Copyright traps for large language models. In *Forty-first International Conference on Machine Learning*.

Matthieu Meeus, Igor Shilov, Shubham Jain, Manuel Faysse, Marek Rei, and Yves-Alexandre de Montjoye. 2024c. Sok: Membership inference attacks on llms are rushing nowhere (and how to fix it). *arXiv preprint arXiv:2406.17975*.

Behzad Mehrbakhsh, Dario Garigliotti, Fernando Martínez-Plumed, and Jose Hernandez-Orallo. 2024. Confounders in instance variation for the analysis of data contamination. In *Proceedings of the 1st Workshop on Data Contamination (CONDA)*, pages 13–21, Bangkok, Thailand. Association for Computational Linguistics.

Meta AI. 2024. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3/. Accessed: 2024-04-18.

Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. 2022. Quantifying privacy risks of masked language models using membership inference attacks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8332–8347, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. 2 olmo 2 furious.

OpenAI. Models - openai api. Accessed on Oct 6, 2024.

OpenAI. 2024a. Gpt-4o mini: advancing cost-efficient intelligence. Accessed on Oct 6, 2024.

OpenAI. 2024b. Hello gpt-4o. Accessed on Oct 6, 2024.

Yonatan Oren, Nicole Meister, Niladri S. Chatterji, Faisal Ladhak, and Tatsunori Hashimoto. 2024. Proving test set contamination in black-box language models. In *The Twelfth International Conference on Learning Representations*.

Medha Palavalli, Amanda Bertsch, and Matthew Gormley. 2024. A taxonomy for data contamination in large language models. In *Proceedings of the 1st Workshop on Data Contamination (CONDA)*, pages 22–40, Bangkok, Thailand. Association for Computational Linguistics.

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

Sundar Pichai and Demis Hassabis. 2024. Our next-generation model: Gemini 1.5. Accessed on Oct 6, 2024.

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

Federico Ranaldi, Elena Sofia Ruzzetti, Dario Onorati, Leonardo Ranaldi, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, and Fabio Massimo Zanzotto. 2024. Investigating the impact of data contamination of large language models in text-to-sql translation. *arXiv preprint arXiv:2402.08100*.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854.

Martin Riddell, Ansong Ni, and Arman Cohan. 2024. Quantifying contamination in evaluating code generation capabilities of language models. *arXiv preprint arXiv:2403.04811*.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023a. 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*, pages 10776–10787.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, and Eneko Agirre. 2023b. Did chatgpt cheat on your test? Accessed: 2024-09-09.

Avi Schwarzschild, Zhili Feng, Pratyush Maini, Zachary C Lipton, and J Zico Kolter. 2024. Rethinking llm memorization through the lens of adversarial compression. *The 1st Workshop on Data Contamination (CONDA)*.

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. In *NeurIPS 2023 Workshop on Regulatable ML*.

Congzheng Song and Vitaly Shmatikov. 2019. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 196–206, New York, NY, USA. Association for Computing Machinery.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *Preprint*, arXiv:2310.16944.

Veniamin Veselovsky, Manoel Horta Ribeiro, and Robert West. 2023. Artificial artificial artificial intelligence: Crowd workers widely use large language models for text production tasks. *arXiv preprint arXiv:2306.07899*.

Johnny Wei, Ryan Wang, and Robin Jia. 2024. Proving membership in LLM pretraining data via data watermarks. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13306–13320, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, et al. 2024. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*.

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

Wen-wai Yim, Yujuan Fu, Asma Ben Abacha, and Meliha Yetisgen. 2024. To err is human, how about medical large language models? comparing pretrained language models for medical assessment errors and reliability. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16211–16223, Torino, Italia. ELRA and ICCL.

Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor R¨uhle, Andrew Paverd, Olga Ohrimenko, Boris K¨opf, and Marc Brockschmidt. 2020. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 363–375.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2023. Counterfactual memorization in neural language models. *Advances in Neural Information Processing Systems*, 36:39321–39362.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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. *arXiv preprint arXiv:2311.01964*.

Wenhong Zhu, Hongkun Hao, Zhiwei He, Yun-Ze Song, Jiao Yueyang, Yumeng Zhang, Hanxu Hu, Yiran Wei, Rui Wang, and Hongyuan Lu. 2024. CLEAN–EVAL: Clean evaluation on contaminated large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 835–847, Mexico City, Mexico. Association for Computational Linguistics.

Yongshuo Zong, Tingyang Yu, Bingchen Zhao, Ruchika Chavhan, and Timothy Hospedales. 2023. Fool your (vision and) language model with embarrassingly simple permutations. *arXiv preprint arXiv:2310.01651*.

A Appendix

A.1 Table of Notations

We present the notations used in this manuscript in Table 4.

A.2 Risks and Mitigation Approaches for Data Contamination

During our paper collection, we identify the relevant research on the risks and mitigation strategies for data contamination, which does not involve proposing or evaluating existing detection approaches for data contamination. While excluding those papers from the main text of this manuscript, we provide their citations in Table 5 in this Appendix.

A.3 Case Study for Instance Similarity

To assess the applicability of instance similarity-based detection approaches (see Section 4.1), we analyzed how frequently their requirements, R1 and R2, are met. We reviewed the top 10 models from the Vellum LLM leaderboard[6]. As demonstrated in Table 6, none of the models fulfilled R1, the most basic requirement. However, some models disclose their cut-off date for collecting training data (Achiam et al., 2023; OpenAI).

[6] https://www.vellum.ai/llm-leaderboard#model-comparison. Accessed on Oct 6, 2024.

Notation	Definition
x	A language instance, as a sequence of tokens.
(x_p, x_s)	A prefix-suffix instance pair, which is a common data format in natural language generation (NLG) tasks.
(x_c, x_k)	A context-key instance pair from slot-filling tasks, as defined in Requirement R5.
M	A language model.
$M.(x)$	M's output respect to an input x, given a decoding setup $\cdot$. If $\cdot$ is not specified, we consider it as a fixed, but unknown decoding state.
D	A dataset, as a set of language instances.
D_M	M's training set.
$b(x, x')$	Binary indicator function for instance-level contamination, which takes two instances x and x' as inputs, and returns *False* (0) or *True* (1), based on the instance similarity.
$S(x, x')$	A function accessing the similarity between two instances, x and x' and outputs a real value.
$f(M, x)$	Gold standard for instance-level contamination, as defined in Equation 1.
$P_M(x)$	The probability of the instance x given an LM M.
Min p% Token	The average probabilities of top p% least likely tokens in an instance x, based on a given LM M.
τ	The contamination threshold for functions.
Var($\{M.(x_p)\}$)	The measure of variations of outputs produced by M under diverse, different sampling strategies. given x_p
PPL_k	The perplexity from an instance's first k tokens.
Entropy k	The entropy of the top k most likely tokens for the next position, defined by Equation 10.
Eval(M, D)	The evaluation result of an LM M on a dataset D.

Table 4: Table of notations.

Model	Citation	Meet Require.	
		R1	R2
Claude 3.5 Sonnet	Anthropic (2024b)	No	-
Claude 3 Opus	Anthropic (2024c)	No	-
Gemini 1.5 Pro	Pichai and Hassabis (2024)	No	-
GPT-4	Achiam et al. (2023)	No	-
Llama 3 Instruct - 70B	Meta AI (2024)	No	-
Claude 3 Haiku	Anthropic (2024a)	No	-
GPT-3.5	OpenAI	No	-
Mixtral 8x7B	Jiang et al. (2024)	No	-
GPT-4o	OpenAI (2024b)	No	-
GPT-4o mini	OpenAI (2024a)	No	-

Table 6: None of the top 10 LMs, in the LLM Leaderboard by Vellum meet the requirements of disclosing pre-training corpora (R1).

A.4 Entropy Calculation

In this section, we explain the procedure for verifying Assumption A6. In the context of casual language modeling, we consider an LM M with a given prefix sequence x_p. Assumption A6 assumes that given x_p, if M has seen an instance with the same prefix, it will generate similar responses, regardless of the sampling strategy used. Since the verification of this assumption can be influenced by various sampling strategies, we quantify the variance in the model's output by measuring the entropy of the token probabilities, which indicates the model's certainty about the next token generation.

To do this, we first compute the probability distribution of the next token over the model's vocabulary. Given that LLMs may contain vocabularies with over 50,000 tokens (Biderman et al., 2023), most tokens have a very low likelihood of being sampled. Therefore, we focus on the Entropy among the top k most likely tokens (**Entropy** k).

At every token position x_p, we calculate the entropy based on the probabilities of the top k tokens, using the following formula:

$$\text{Entropy}_k(M, x_p) = -\sum_{i=1}^{k} P_i(M(x_p)) \log P_i(M(x_p)) \tag{10}$$

Given an instance x with N tokens, the Entropy k for x is the average $\text{Entropy}_k(M, x_p)$ across all tokens x_p in x:

$$\text{Entropy}_k(M, x) = (\sum_{p=1}^{N} \text{Entropy}_k(M, x_p))/N \tag{11}$$

A.5 More Case Study Results

A.5.1 Within-Domain Detection with Different LMs

In this section, we present the detailed detection AUCs for all models: (1) different sizes of Pythia Models: Pythia-70m (Table 7), Pythia-160m (Table 8), Pythia-410m (Table 9), Pythia-1.4b (Table 10), Pythia-2.8b (Table 11), Pythia-6.9b (Table 12), Pythia-12b (Table 13); (2) OLMo-2-7B (Table 14); and (3) BioMistral-NLU-7B (Table 15).

Similar to the results in Table 3, we observed close-to-random performance in the detection AUCs for all Pythia models and dataset domains.

Citation	Content
Zhou et al. (2023)	Impact of direct data contamination on test performance
Dekoninck et al. (2024)	Impact of indirect data contamination on test performance
Jacovi et al. (2023)	Strategies to prevent contamination in benchmark datasets
Zhu et al. (2024)	Strategies to mitigate contamination in benchmark datasets
Haimes et al. (2024)	Mitigating data contamination in benchmarks through retrospectively creating held-out datasets.
Kusa et al. (2024)	Proposing an evaluation pipeline in Systematic Literature Review to mitigate data contamination.
McCoy et al. (2023)	Evaluating the novelty of LM-generated text
Maini et al. (2024a)	Studying unlearning methods to make LMs forget specific training data
Bowen et al. (2024)	Studying contributing factors behind data poisoning, with corrupted or malicious training data.
Mitchell et al. (2023)	Differentiates human vs. machine-generated text using probability curvature

Table 5: Seleted relevant work to risks and mitigation approaches for data contamination.

Assumptions & Metric		Github	FreeLaw	Enron-Emails	ArXiv	OpenWeb-Text2	Open-Subtitles	Hacker-News	Youtube-Subtitles	Pile-CC
A1	PPL_50	49.4	49.3	50.5	51.3	51.7	47.4	48.9	52.2	49.9
	PPL_100	50.3	50.2	51.5	50.8	50.5	46.2	48.1	52.0	50.8
	PPL_200	50.3	50.1	53.5	50.5	49.9	44.3	51.1	50.8	51.5
	Min 5% token	51.7	49.9	49.9	50.6	48.9	45.0	50.4	49.9	51.1
	Min 15% token	51.5	49.5	51.0	50.5	49.5	45.4	50.3	49.6	51.6
	Min 25% token	51.4	50.1	51.0	50.9	49.9	45.5	49.5	49.8	51.3
A4	Mem 5	47.6	49.2	49.5	51.5	50.8	48.8	49.5	50.0	51.2
	Mem 15	49.6	49.5	48.7	50.1	50.4	49.0	49.3	50.4	51.2
	Mem 25	49.6	49.9	48.3	50.6	50.3	48.2	49.6	49.6	51.0
A6	Entropy 5	50.8	49.7	48.2	52.1	49.9	47.4	49.3	50.4	50.2
	Entropy 15	50.6	49.8	48.9	52.4	49.4	47.5	49.3	50.5	50.3
	Entropy 25	50.6	49.9	49.2	52.4	49.0	47.7	49.3	50.2	50.2
Average AUC		50.1	49.7	49.8	51.2	50.1	47.3	49.4	50.3	50.9
PPL_{200}	Seen	11.1±12.7	13.1±8.9	29.5±21.4	28.9±13.5	46.0±23.5	36.7±21.1	42.9±16.7	45.3±41.3	50.2±33.7
	Unseen	11.5±20.9	14.2±25.3	31.7±22.4	29.0±13.1	45.4±22.5	33.7±18.2	43.7±18.8	44.3±28.7	51.5±35.6

Table 7: Average contamination detection AUC for the **pythia-70m** model, under different domains within the Pile dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

A.5.2 Metric Distribution in Histogram

In this section, we present the distributions of different metrics both within domain and across domains.

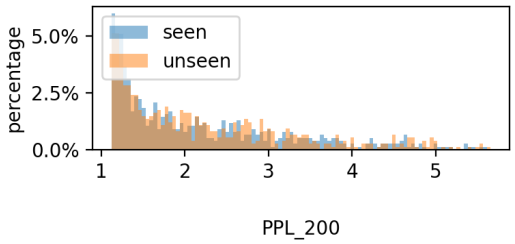

Figure 2: The density plot of **PPL_200** from the Pythia-6.9b model, when both seen and unseen instances are from the **Github** domain.

We compare the MIA performance between the GitHub and Pile-CC domains, from the Pythia-6.9b model. As shown in Figure 2 & 3, when the seen and unseen instances are from the same domain, their PPL_200 distributions are very similar. However, as shown in Figure 4 & 5, when the seen and unseen instances are from different domains, their PPL_200 distributions are very different. This indicates that the PPL_200 relates more to domain shifts, instead of the contamination status of individual instances.

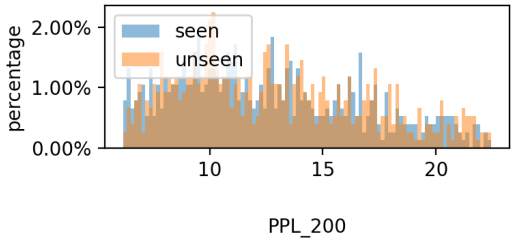

Figure 3: The density plot of **PPL_200** from the Pythia-6.9b model, when both seen and unseen instances are from the **Pile-CC** domain.

Assumptions & Metric		Github	FreeLaw	Enron-Emails	ArXiv	OpenWeb-Text2	Open-Subtitles	Hacker-News	Youtube-Subtitles	Pile-CC
A1	PPL_50	49.7	49.0	50.3	51.2	51.9	47.6	48.6	52.3	50.0
	PPL_100	50.3	49.4	51.2	51.0	50.4	46.1	47.6	51.7	50.7
	PPL_200	50.1	49.4	53.0	50.7	49.6	44.2	50.5	50.7	51.7
	Min 5% token	51.9	49.9	51.0	50.5	48.6	45.0	48.6	49.4	51.3
	Min 15% token	51.4	49.8	50.9	50.8	49.5	45.1	50.1	49.3	51.4
	Min 25% token	51.3	49.7	51.3	51.3	49.6	45.6	49.4	49.2	51.4
A4	Mem 5	48.3	49.1	50.4	52.8	51.2	52.6	50.1	49.9	51.1
	Mem 15	48.2	49.2	50.1	51.4	51.5	48.0	49.1	49.3	50.0
	Mem 25	48.4	49.0	49.9	50.5	51.4	46.2	49.2	49.4	50.2
A6	Entropy 5	51.3	49.1	48.9	52.1	49.8	47.0	48.8	50.1	50.6
	Entropy 15	51.1	49.3	49.4	52.2	49.7	47.5	48.9	50.1	50.6
	Entropy 25	51.0	49.4	49.6	52.1	49.4	47.7	48.9	50.0	50.6
Average AUC		50.1	49.3	50.4	51.5	50.3	47.7	49.2	49.9	50.8
$PPL_2$00	Seen	7.4±8.5	8.2±5.6	18.8±13.5	18.6±8.6	29.9±31.5	45.8±598.9	29.0±11.0	30.9±28.2	33.3±25.3
	Unseen	7.6±13.5	8.8±16.7	20.3±14.8	18.7±8.3	28.5±12.8	24.8±11.4	29.6±13.1	30.4±23.0	34.4±25.9

Table 8: Average contamination detection AUC for the **pythia-160m** model, under different domains within the Pile dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

Assumptions & Metric		Github	FreeLaw	Enron-Emails	ArXiv	OpenWeb-Text2	Open-Subtitles	Hacker-News	Youtube-Subtitles	Pile-CC
A1	PPL_50	49.5	49.5	50.1	50.7	51.0	47.4	48.8	51.5	50.0
	PPL_100	50.5	49.2	51.3	50.9	49.7	45.5	47.6	51.0	50.3
	PPL_200	50.3	49.3	53.0	50.6	48.8	44.6	50.9	50.2	52.0
	Min 5% token	52.2	49.8	50.3	50.7	49.0	45.5	49.3	48.9	51.4
	Min 15% token	51.6	49.6	50.9	51.0	49.3	46.0	50.2	49.0	51.2
	Min 25% token	51.4	49.4	51.0	51.3	48.9	46.4	50.2	48.9	51.0
A4	Mem 5	47.5	49.2	51.3	53.0	50.5	50.3	49.4	49.7	50.3
	Mem 15	47.6	49.2	51.3	52.6	51.8	50.3	49.7	50.1	50.4
	Mem 25	48.2	49.2	51.1	51.2	51.9	52.6	50.5	49.2	51.2
A6	Entropy 5	51.2	49.1	49.4	52.2	50.5	47.4	49.0	49.5	50.2
	Entropy 15	51.0	49.3	49.9	52.0	49.7	48.0	48.9	49.5	50.2
	Entropy 25	50.9	49.4	50.1	51.9	49.4	48.2	48.9	49.2	50.3
Average AUC		50.0	49.4	50.8	51.9	50.1	48.4	49.5	49.7	50.7
$PPL_2$00	Seen	4.7±5.0	5.5±3.4	11.8±8.2	12.7±6.0	18.8±9.7	20.0±9.1	19.8±7.5	20.6±22.2	22.3±17.0
	Unseen	4.8±7.2	5.9±11.4	12.9±9.6	12.7±5.7	18.2±8.3	18.5±7.9	20.4±9.3	20.0±14.8	23.0±17.8

Table 9: Average contamination detection AUC for the **pythia-410m** model, under different domains within the Pile dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

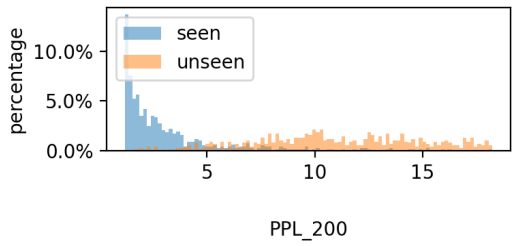

Figure 4: The density plot of **PPL_200** from the Pythia-9.6b model, when the seen instances are from the **Github** domain, and the unseen instances are from the **Pile-CC** domain.

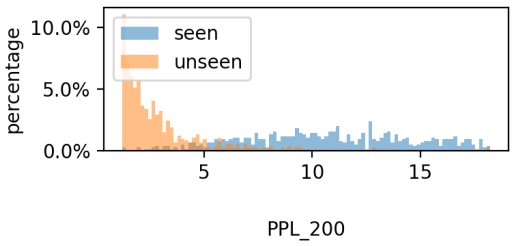

Figure 5: The density plot of **PPL_200** from the Pythia-9.6b model, when the seen instances are from the **Pile-CC** domain, and the unseen instances are from the **Github** domain.

We observe a similar trend for other metrics: Min 25% Prob (Figure 6, 7, 8, 9), Mem 25 (Figure 10, 11, 12, 13), Entropy 25 (Figure 14, 15, 16, 17).

Assumptions & Metric		Github	FreeLaw	Enron-Emails	ArXiv	OpenWeb-Text2	Open-Subtitles	Hacker-News	Youtube-Subtitles	Pile-CC
A1	PPL_50	49.5	49.0	49.8	50.7	50.7	48.1	48.5	51.0	49.8
	PPL_100	50.7	49.5	50.9	51.0	49.2	45.6	47.4	50.9	50.2
	PPL_200	50.8	49.8	51.9	50.9	48.7	44.4	50.9	49.9	52.0
	Min 5% token	51.8	50.5	49.7	51.0	49.2	46.2	48.9	48.4	51.1
	Min 15% token	51.5	49.7	50.3	51.3	49.6	47.0	50.1	48.7	51.1
	Min 25% token	51.4	49.4	50.5	51.5	49.1	47.8	50.0	48.6	51.2
A4	Mem 5	48.8	49.1	50.9	53.0	51.0	51.5	49.6	49.1	50.5
	Mem 15	48.0	49.5	50.8	51.4	51.2	51.7	49.8	48.9	50.4
	Mem 25	48.7	49.5	51.1	51.0	51.5	50.9	49.7	48.7	51.0
A6	Entropy 5	51.2	49.0	49.9	52.0	49.9	46.6	48.7	49.3	50.2
	Entropy 15	51.0	49.1	50.1	51.9	49.5	48.0	48.7	48.9	50.2
	Entropy 25	51.0	49.2	50.1	51.8	49.3	48.5	48.8	48.7	50.3
Average AUC		50.2	49.4	50.5	51.7	50.1	48.6	49.2	49.1	50.6
PPL_{200}	Seen	3.6±3.8	4.4±2.4	8.5±5.9	9.8±4.6	14.2±7.4	16.0±6.9	15.3±5.7	16.3±19.2	17.1±12.5
	Unseen	3.6±4.9	4.7±8.6	9.3±7.3	9.9±4.4	13.7±6.4	14.8±6.1	15.7±7.2	15.7±12.3	17.8±13.9

Table 10: Average contamination detection AUC for the **pythia-1.4b** model, under different domains within the Pile dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

Assumptions & Metric		Github	FreeLaw	Enron-Emails	ArXiv	OpenWeb-Text2	Open-Subtitles	Hacker-News	Youtube-Subtitles	Pile-CC
A1	PPL_50	49.4	48.6	49.4	50.7	50.5	47.8	48.0	51.1	49.9
	PPL_100	50.5	49.2	50.6	50.9	49.0	45.9	47.5	51.0	50.4
	PPL_200	50.6	49.6	51.7	50.8	48.6	45.3	50.8	49.9	52.2
	Min 5% token	51.6	49.8	49.9	51.2	49.5	47.2	48.2	48.6	51.6
	Min 15% token	51.5	49.6	50.3	51.4	49.5	48.1	49.4	48.7	51.3
	Min 25% token	51.2	49.4	50.1	51.4	48.9	48.8	49.7	48.6	51.0
A4	Mem 5	48.7	49.1	50.7	52.8	50.9	51.5	49.5	50.2	50.9
	Mem 15	48.2	48.9	50.4	51.0	51.3	50.4	48.9	49.7	50.2
	Mem 25	48.9	48.5	50.5	50.8	51.4	50.0	48.7	49.3	50.2
A6	Entropy 5	50.9	49.1	49.5	51.9	50.0	47.5	48.9	49.0	50.6
	Entropy 15	50.8	49.2	49.8	51.9	49.5	48.7	49.0	48.9	50.5
	Entropy 25	50.8	49.2	50.0	51.8	49.3	49.1	49.0	48.8	50.6
Average AUC		50.1	49.2	50.2	51.6	50.0	48.8	49.0	49.4	50.8
PPL_{200}	Seen	3.2±3.3	3.9±2.1	7.2±5.1	8.7±4.1	12.5±6.5	14.0±6.2	13.3±4.9	14.4±17.3	15.0±10.2
	Unseen	3.2±4.4	4.2±7.8	7.9±6.3	8.7±3.9	12.1±5.8	13.1±5.5	13.6±6.1	13.9±11.3	15.7±12.2

Table 11: Average contamination detection AUC for the **pythia-2.8b** model, under different domains within the Pile dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

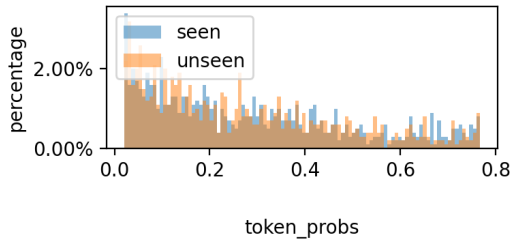

Figure 6: The density plot of **Min 25% Prob** when both seen and unseen instances are from the **Github** domain.

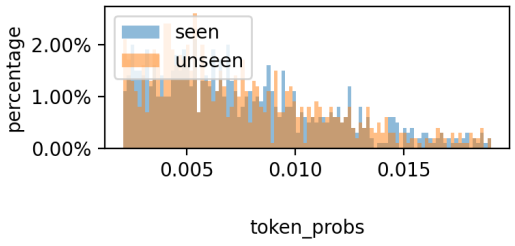

Figure 7: The density plot of **Min 25% Prob** when both seen and unseen instances are from the **Pile-CC** domain.

Assumptions & Metric		Github	FreeLaw	Enron-Emails	ArXiv	OpenWeb-Text2	Open-Subtitles	Hacker-News	Youtube-Subtitles	Pile-CC
A1	PPL_50	49.7	48.9	49.5	50.7	50.3	47.7	48.5	50.7	49.8
	PPL_100	50.7	49.4	50.3	51.1	48.8	46.2	47.4	50.4	50.4
	PPL_200	50.8	49.4	51.1	50.9	48.4	46.7	50.7	49.6	52.1
	Min 5% token	51.7	49.8	49.2	51.3	49.7	48.2	47.8	48.5	50.9
	Min 15% token	51.6	49.7	49.8	51.3	49.1	49.5	49.7	48.6	51.1
	Min 25% token	51.3	49.3	50.0	51.4	48.8	50.2	49.8	48.5	51.1
A4	Mem 5	49.2	48.7	50.6	52.7	50.5	50.0	49.9	49.1	50.8
	Mem 15	48.9	48.9	51.0	51.9	51.8	50.6	49.2	48.6	50.7
	Mem 25	49.6	48.8	51.1	51.6	51.1	50.3	49.5	48.2	51.3
A6	Entropy 5	51.0	49.1	49.5	52.0	49.7	48.4	49.1	49.3	50.9
	Entropy 15	50.8	49.1	49.7	52.0	49.3	49.6	49.0	49.0	50.9
	Entropy 25	50.8	49.2	49.8	51.9	49.1	50.0	49.0	48.9	51.0
Average AUC		50.3	49.2	50.2	51.8	49.9	49.4	49.3	49.0	50.9
$PPL_2$00	Seen	2.8±3.0	3.6±1.9	6.1±4.4	7.9±3.7	11.2±5.8	12.2±5.9	12.2±4.5	13.2±16.0	13.7±9.1
	Unseen	2.9±4.1	3.8±6.2	6.7±5.6	8.0±3.6	10.9±5.3	11.5±5.1	12.4±5.4	12.7±10.6	14.3±11.0

Table 12: Average contamination detection AUC for the **pythia-6.9b** model, under different domains within the Pile dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

Assumptions & Metric		Github	FreeLaw	Enron-Emails	ArXiv	OpenWeb-Text2	Open-Subtitles	Hacker-News	Youtube-Subtitles	Pile-CC
A1	PPL_50	49.9	48.7	49.5	51.0	50.4	48.5	48.3	50.7	49.8
	PPL_100	50.9	49.6	50.2	50.9	48.9	47.4	47.3	50.2	50.3
	PPL_200	50.9	49.5	51.0	51.0	48.5	48.4	50.5	49.4	51.8
	Min 5% token	51.9	50.2	49.2	51.4	49.5	49.2	47.8	48.7	50.9
	Min 15% token	51.7	49.4	49.8	51.3	49.0	50.2	49.4	48.7	51.0
	Min 25% token	51.4	49.1	49.9	51.4	48.6	50.8	49.3	48.8	51.0
A4	Mem 5	49.0	49.0	50.9	53.9	51.0	52.6	49.6	48.5	50.9
	Mem 15	48.5	49.5	51.0	52.5	51.1	49.0	49.1	48.3	50.0
	Mem 25	49.5	49.2	50.6	52.7	50.7	48.6	49.2	48.2	51.4
A6	Entropy 5	51.0	49.0	49.5	52.0	50.0	49.2	48.8	49.3	51.0
	Entropy 15	50.9	49.1	49.7	51.9	49.6	50.6	48.8	49.1	50.9
	Entropy 25	50.9	49.2	49.8	51.8	49.4	51.0	48.8	49.0	50.9
Average AUC		50.3	49.2	50.2	52.1	49.9	49.9	48.9	48.9	50.8
$PPL_2$00	Seen	2.6±2.8	3.4±1.7	5.4±4.0	7.5±3.5	10.4±5.4	11.0±5.7	11.2±4.0	12.3±15.0	12.9±8.4
	Unseen	2.7±3.7	3.6±5.6	5.9±5.0	7.5±3.4	10.1±5.0	10.6±5.0	11.5±5.0	11.8±10.0	13.4±10.1

Table 13: Average contamination detection AUC for the **pythia-12b** model, under different domains within the Pile dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

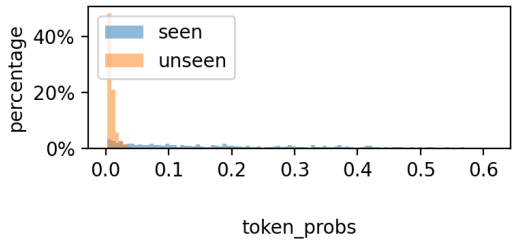

Figure 8: The density plot of **Min 25% Prob** from the Pythia-9.6b model, when the seen instances are from the **Github** domain, and the unseen instances are from the **Pile-CC** domain.

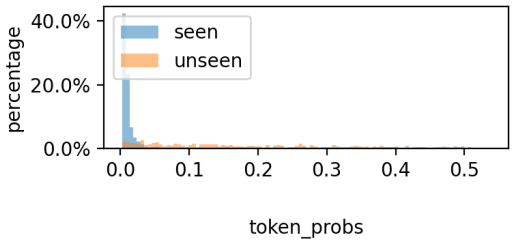

Figure 9: The density plot of **Min 25% Prob** from the Pythia-9.6b model, when the seen instances are from the **Pile-CC** domain, and the unseen instances are from the **Github** domain.

Assumptions & Metric		cpp	python	Github-Lean	julia	tex	Github-Isabelle	fortran	Github-Coq	r
A1	PPL_50	51.1	51.7	49.9	50.2	49.4	50.9	50.3	49.1	49.5
	PPL_100	51.4	51.6	52.0	51.3	50.4	51.7	50.6	49.1	51.4
	PPL_200	51.8	50.8	51.2	51.6	50.0	50.8	51.1	49.0	53.0
	Min 5% token	50.1	51.1	49.0	52.0	49.4	51.2	48.1	49.4	50.9
	Min 15% token	50.2	50.4	50.7	51.2	48.6	50.4	48.1	49.0	51.5
	Min 25% token	50.3	49.8	51.3	51.0	48.7	49.9	48.2	48.9	52.4
A4	Mem 5	51.3	48.2	51.8	49.0	48.2	55.4	50.8	50.5	51.6
	Mem 15	52.0	49.6	50.7	48.3	48.5	54.6	50.6	50.4	52.0
	Mem 25	51.6	50.5	50.1	49.1	49.4	54.3	50.4	50.8	50.9
A6	Entropy 5	50.2	49.4	51.7	51.3	48.8	49.0	48.8	48.9	54.1
	Entropy 15	50.2	49.4	51.7	51.3	48.8	48.9	48.8	49.0	53.5
	Entropy 25	50.3	49.5	51.8	51.2	48.8	49.0	48.8	49.0	53.5
Average AUC		50.9	49.8	50.9	50.4	49.0	51.6	49.6	49.7	52.2
PPL_{200}	Seen	4.3±2.6	6.7±3.9	6.9±3.4	8.3±5.0	8.3±5.1	8.7±5.4	9.5±6.8	10.4±8.3	10.5±7.0
	Unseen	4.6±3.0	6.8±4.1	6.9±3.0	8.6±5.6	8.7±6.3	9.2±6.5	9.9±7.6	9.9±7.2	10.5±5.6

Table 14: Average contamination detection AUC for the **OLMo-2-1124-7B** model, under different domains within the Algebraic Stack dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

Assumptions & Metric		RE-2012temp	STS-B	DC-MTSample	RE-2011coref	events-BioRed	events-NLMGene	events-2012temp	events-2006deid	events-BioASQ
A1	PPL_50	**62.5**	**83.3**	51.9	**60.5**	50.4	49.4	50.0	51.4	**67.6**
	PPL_100	**95.3**	**93.3**	60.0	**70.0**	50.0	48.8	50.4	59.3	52.6
	PPL_200	**99.4**	**96.8**	58.9	**70.5**	**89.4**	**87.4**	**76.2**	**79.0**	**66.1**
	Min 5% token	**92.9**	**93.4**	47.4	**72.6**	**61.4**	**76.1**	57.5	**78.0**	**74.9**
	Min 15% token	**93.3**	**93.4**	51.7	**70.1**	**88.3**	**85.3**	**71.2**	**82.3**	**69.4**
	Min 25% token	**93.4**	**92.1**	53.3	**68.5**	**94.1**	**80.5**	**74.8**	**76.5**	**64.7**
A4	Mem 5	41.4	48.0	47.9	46.6	49.8	52.0	51.0	52.5	49.2
	Mem 15	45.2	53.3	49.1	46.4	48.9	52.3	51.9	52.7	52.8
	Mem 25	48.9	55.9	50.3	48.4	46.6	52.1	52.6	53.1	52.5
A6	Entropy 5	**93.2**	**80.5**	55.4	**63.9**	**94.5**	**65.3**	**74.3**	**62.6**	43.1
	Entropy 15	**93.1**	**82.0**	54.8	**64.5**	**94.5**	**66.6**	**74.2**	**63.7**	46.0
	Entropy 25	**93.1**	**82.6**	54.1	**64.7**	**94.5**	**67.1**	**74.4**	**64.2**	47.6
Average AUC		**74.1**	**73.1**	52.0	59.0	**69.8**	**63.5**	**62.4**	**62.8**	55.1
PPL_{200}	Seen	1.4±0.1	1.5±0.1	1.7±0.2	2.1±0.8	2.8±0.3	3.1±0.4	3.2±0.4	3.3±0.6	8.1±2.3
	Unseen	3.0±1.6	2.0±0.3	1.8±0.2	3.6±2.4	3.8±0.9	4.3±1.0	4.1±1.1	4.6±1.6	9.4±2.7

Table 15: Average contamination detection AUC for the **BioMistral** model, under different domains within the Medical-NLU dataset. 'PPL_200' represents the average perplexity ± STD, from the first 200 tokens within every instance. The color green represents AUCs higher than 60.

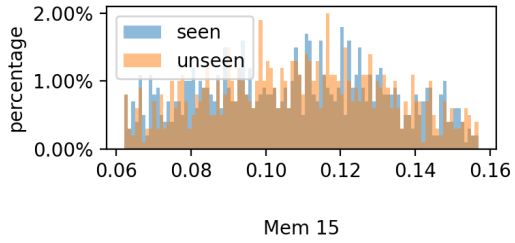

Figure 10: The density plot of **Mem 25** from the Pythia-6.9b model, when both seen and unseen instances are from the **Github** domain.

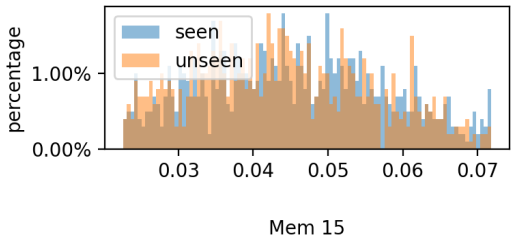

Figure 11: The density plot of **Mem 25** from the Pythia-6.9b model, when both seen and unseen instances are from the **Pile-CC** domain.

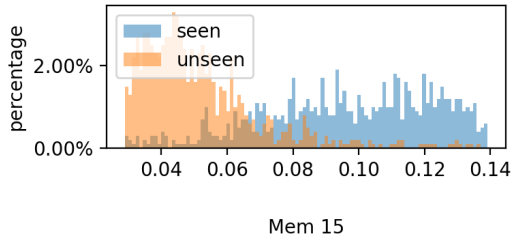

Figure 12: The density plot of **Mem 25** from the Pythia-9.6b model, when the seen instances are from the **Github** domain, and the unseen instances are from the **Pile-CC** domain.

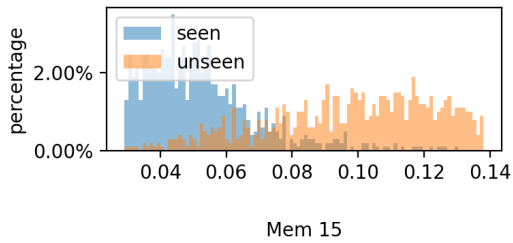

Figure 13: The density plot of **Mem 25** from the Pythia-9.6b model, when the seen instances are from the **Pile-CC** domain, and the unseen instances are from the **Github** domain.

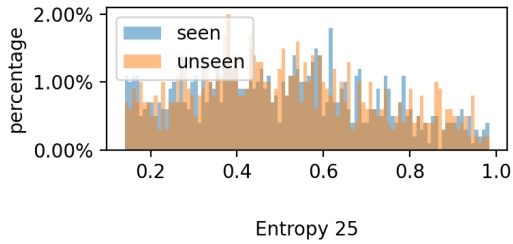

Figure 14: The density plot of **Entropy 25** from the Pythia-6.9b model, when both seen and unseen instances are from the **Github** domain.

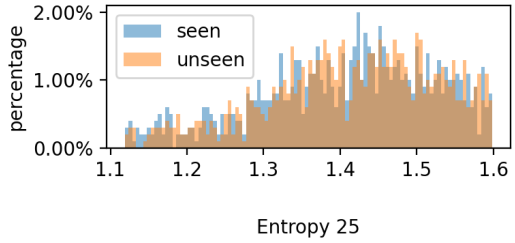

Figure 15: The density plot of **Entropy 25** from the Pythia-6.9b model, when both seen and unseen instances are from the **Pile-CC** domain.

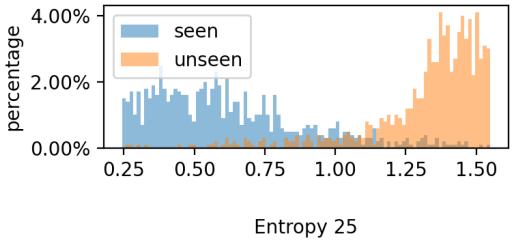

Figure 16: The density plot of **Entropy 25** from the Pythia-9.6b model, when the seen instances are from the **Github** domain, and the unseen instances are from the **Pile-CC** domain.

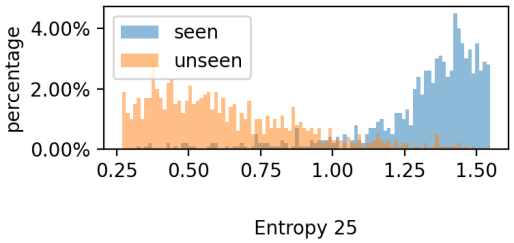

Figure 17: The density plot of **Entropy 25** from the Pythia-9.6b model, when the seen instances are from the **Pile-CC** domain, and the unseen instances are from the **Github** domain.

A.5.3 Cross-Domain Detection with Different Metrics

In this section, we present AUC results with other metrics, when seen and unseen instances are from different domains, for the Pythia-6.9b model. The metrics include Min 25% token (Figure 18), Mem 25 (Figure 19), and Entropy 25 (Figure 20). All metrics exhibit higher AUC values in the top-right corner and lower values in the bottom-left, while diagonal points approach random guessing.

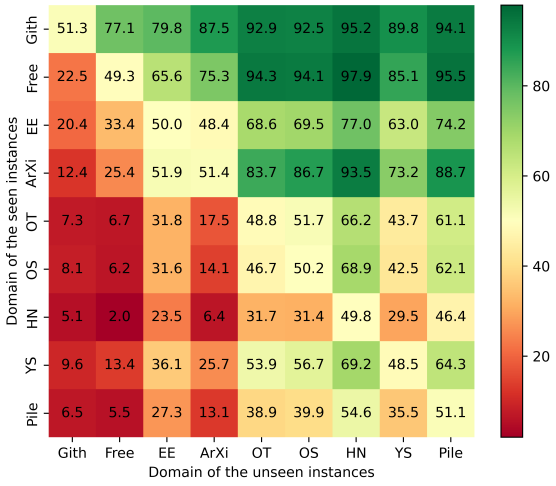

Figure 18: Average contamination detection AUC for the Pythia-6.9b model with the metric, **Min 25% token**, when the seen and unseen instances are from different domains. The abbreviations represent the domains in Table 12.

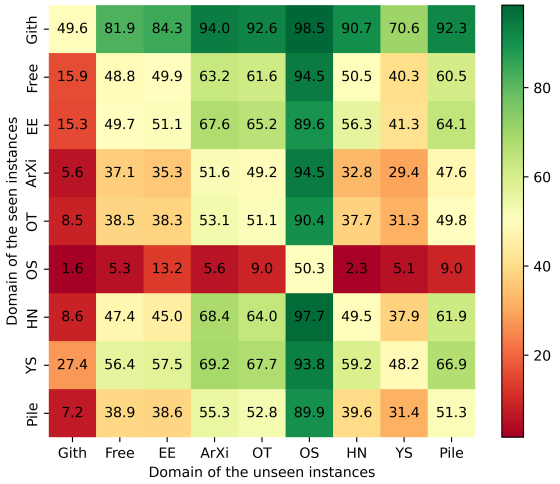

Figure 19: Average contamination detection AUC for the Pythia-6.9b model with the metric, **Mem 25**, when the seen and unseen instances are from different domains. The abbreviations represent the domains in Table 12.

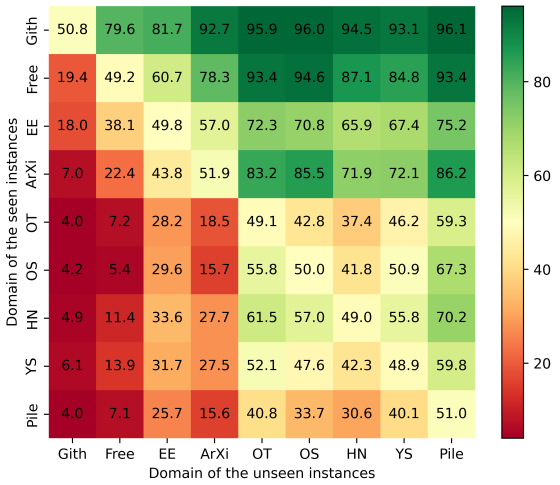

Figure 20: Average contamination detection AUC for the Pythia-6.9b model with the metric, **Entropy 25**, when the seen and unseen instances are from different domains. The abbreviations represent the domains in Table 12.