A Little Leak Will Sink a Great Ship: Survey of Transparency for Large Language Models from Start to Finish

Anonymous ACL submission

Abstract

Large Language Models (LLMs) are trained on massive web-crawled corpora. An increasing issue is LLMs generating content based on leaked data, and the need to detect and suppress such generated results, including personal information, copyrighted text, and benchmark datasets. A fundamental cause of this issue is leaked data in the training dataset. However, existing research has not sufficiently clarified the relationship between leaked instances in the training data and the ease of output and detection of these leaked instances by LLMs. In this paper, we conduct an experimental survey to elucidate the relationship between the rate of leaked instances in the training dataset and the generation and detection of LLMs in relation to the leakage of personal information, copyrighted texts, and benchmark data. Our experiments reveal that LLMs generate leaked information in most cases despite there being little such data in the training set. Furthermore, the lower the rate of leaked instances, the more difficult it becomes to detect the leakage. When addressing the leakage problem in the training dataset, we must be careful as reducing leakage instances does not necessarily lead to only positive effects. Finally, we demonstrate that explicitly defining the leakage detection task using examples in LLMs can help mitigate the impact of the rate of leakage instances in the training data on detection.

1 Introduction

Large Language Models (LLMs) have achieved remarkable performance in various real-world applications (Brown et al., 2020; Wei et al., 2021; Ouyang et al., 2022). One of the success factors is the massive web-crawled corpora used for pre-training LLMs (Kaplan et al., 2020; Wei et al., 2022). The corpora for pre-training LLMs consist of webpages, books, scientific papers, and programming code (Almazrouei et al., 2023; Zhao et al., 2023). Developers of well-known LLMs such as ChatGPT[1] and Claude 3[2] infamously do not disclose the composition of the training data, to maintain a competitive edge.

The large-scale nature and privatization of such training data increases the risk of leaking inappropriate data such as personal information, copyrighted works, and LLM benchmarks (Ishihara, 2023). It has been revealed that it is possible to efficiently recover training data from LLMs under various settings, including those with and without alignment learning (Nasr et al., 2023). This facilitates the collection of personal information and copyrighted works by malicious actors through LLMs. In practice, it has been confirmed that personal information, such as names, phone numbers, and email addresses, has leaked from LLMs via membership inference attacks (Shokri et al., 2016), an attack method that guesses whether a particular instance is included in the training data (Carlini et al., 2020; Huang et al., 2022; Kim et al., 2023). The leak of benchmarks significantly enhances the reported performance of LLMs (Deng et al., 2023; Zhou et al., 2023), leading to over-confidence in the abilities of LLMs. Furthermore, it has become apparent that works such as news articles[3] and books[4] can be directly generated by LLMs, and that the training data includes pirated content (Eldan and Russinovich, 2023). As just described, the leakage of inappropriate content in the training data of LLMs can lead to a loss of trust in the coexistence of humans and AI.

Data leakage in LLMs originates from the leakage of instances in the pre-training data, leading to the output of leaked instances by the LLMs. Data

[1] https://chat.openai.com/

[2] https://claude.ai/chats

[3] https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html

[4] https://www.theatlantic.com/technology/archive/2023/08/books3-ai-meta-llama-pirated-books/675063/

leakage detection can be conducted to ensure that the LLM output does not contain any leaked instances. We establish the following three criteria concerning leakage issues:

- **Leakage Rate** refers to the proportion of leaked instances contained in the pre-training data of LLMs.
- **Generation Rate** refers to the proportion of the evaluation dataset where the LLM can reproduce leaked instances when instructed to do so.
- **Detection Rate** refers to the performance of LLMs in distinguishing between leaked and non-leaked instances in the evaluation dataset.

Despite the leakage rate being the origin of data leakage issues, it is not understood how it affects the generation rate and detection rate. In this paper, we conduct an experimental survey to elucidate the relationship between the leakage rate and both the generation rate and detection rate for personal information, copyrighted texts, and benchmark data. This leads to new insights into how we should address leaks in pre-training data, which are the root cause of leakage issues.

Regarding the leakage rate, while there have been reports on the investigation of personal information leakage in pre-training data (Subramani et al., 2023; Longpre et al., 2023), the leakage rates in copyrighted texts and benchmarks have not been disclosed. The work has been conducted using regular expressions, which cannot be easily applied to detecting copyrighted texts and benchmarks. We investigate the leakage rates in pre-training data not only for personal information but also for copyrighted texts and benchmarks using web searches. Regarding the detection rate, existing methods detect whether instances are leaked based on the likelihood or loss function thresholds of LLMs (Carlini et al., 2020; Shi et al., 2023; Fu et al., 2023).

In our experiments, based on sampling 5 million instances from the pre-training data of LLMs and investigating the leakage rates for personal information, copyrighted texts, and benchmarks, the rates are to be 75.1%, 19.0%, and 0.1%, respectively. Detection rates are increasingly high for, in order, personal information, copyrighted texts, and benchmarks, with higher leakage rates leading to better detection performance. This suggests that the higher the leakage rate, the more beneficial information LLMs can learn during pre-training to distinguish leaked instances. On the other hand, no significant difference is observed between the generation rates for personal information, copyrighted texts, and benchmarks. These results indicate that a small leakage rate in pre-training data does not significantly influence the tendency of LLMs to output leaked instances, but it can make detecting leaked instances more challenging. Therefore, simply reducing the leakage rate does not necessarily bring only positive effects. It is necessary to apply preprocessing to balance the leakage and detection rates.

Finally, we aim to mitigate the impact of the leakage rate on the detection rate. Existing methods do not explicitly define the task of classifying leaked and non-leaked instances for LLMs. Therefore, if the number of leaked instances in the training data is small, the information from these instances may not be sufficiently reflected in the output. We introduce a detection method that explicitly teaches the task definition by using a few-shot approach to present leaked and non-leaked instances. Our experimental results show that the few-shot-based detection method performs on average about 7 points higher than existing methods. On the other hand, the detection rate drops in the zero-shot case without providing examples, suggesting that providing examples to LLMs is particularly important.

2 Leakage Rate

The leakage rate is the proportion within the leakage instances we targeted in the pre-training dataset, including personal information, copyrighted texts, and benchmark datasets. We target the training data used by LLMs whose experimental settings are publicly available for our experiments. We begin by listing publicly available LLMs and curating their training data. Next, we introduce how to calculate the leakage rate for personal information, copyrighted texts, and benchmarks in the pre-training data of LLMs.

2.1 Pre-training Datasets

In this study, we target the pre-training data of the following six LLMs for which the details of the experimental setup are publicly available.

- **T5** (Raffel et al., 2019): T5 uses the Colossal Clean Crawled Corpus (C4) containing about 800 GB of text data collected from filtered and cleaned web pages as its pre-training data. Scientific texts, books, and news account for

LLMs	Size	C4	CommonCrawl	The Pile	GitHub	Wikipedia	Books	Papers	Conversations
T5	800	**100.0%**	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
LLaMA	4,700	15.0%	**67.0%**	0.0%	4.5%	4.5%	4.5%	2.5%	2.0%
Pythia	800	0.0%	0.0%	**100.0%**	0.0%	0.0%	0.0%	0.0%	0.0%
MPT	4,000	**63.4%**	8.5%	0.0%	14.5%	4.0%	3.0%	5.2%	1.4%
Falcon	3,600	0.0%	**84.0%**	0.0%	3.0%	1.0%	6.0%	1.0%	5.0%
OLMo	5,300	5.7%	**78.7%**	0.0%	12.6%	0.1%	0.1%	2.8%	0.0%

Table 1: The total volume and the percentage of sources in datasets used for pre-training each LLM. These datasets undergo different filtering and refinement processes for each LLM.

approximately 25% in C4. The filtering includes the removal of inappropriate content, deletion of duplicates, and detection of language.

- **LLaMA** (Touvron et al., 2023): LLaMA employs English CommonCrawl, C4, Github, Wikipedia, Books, ArXiv, and StackExchange as pre-training datasets.
- **Pythia** (Biderman et al., 2023): Pythia uses the Pile[5], which comprises 800GB of text data. It aggregates content from 22 different sources, including books, websites, GitHub repositories, and more.
- **MPT** (Team, 2023): MPT uses RedPajama dataset (Computer, 2023), which preprocesses the Common Crawl, Wikipedia, Books, ArXiv, and StackExchange to remove low-quality content and duplicate pages.
- **Falcon** (Almazrouei et al., 2023): Falcon utilizes the RefinedWeb dataset (Penedo et al., 2023), which employs heuristic rules to filter the Common Crawl dataset and remove duplicates.
- **OLMo** (Groeneveld et al., 2024): OLMo uses Dolma (Soldaini et al., 2024), which is a dataset of 3T tokens from a diverse mix of web content, academic publications, code, books, and encyclopedic materials.

We present the configuration of the LLMs and the pre-training data used in our experiments in Table 1. The most common sources included in all LLMs are web page sources such as C4, CommonCrawl, and the Pile. Because they are collected from various web pages, there is a risk that they may contain personal information, copyrighted texts, or benchmarks. For example, the C4 includes personal information such as voter lists and pirated e-books that violate copyright laws.[6] Data from books and papers particularly related to copyrighted texts are explicitly included in LLaMA, MPT, and Falcon at a rate of more than 5%. Using the entire pre-training datasets is not practical from a computational resource perspective. We sampled 5 million instances from the pre-training data used in each of the LLMs and investigated the leakage rates of personal information, copyrighted texts, and benchmarks.

2.2 Scopes of Leakage Instances in the Pre-training Datasets

We determine whether personal information is included in the text through regular expressions proposed in the existing research (Subramani et al., 2023). This regular expression targets 20 types[7] of personal information. Additionally, we determine whether a person's name is included in the text using named entity recognition from the `spaCy` library[8]. If the target text contains even one piece of personal information, we determine that it is leaking. We targeted books, news articles, and papers found on Google Books[9], Google News[10], and Google Scholar[11] as the subjects of the copyrighted texts. We use the `Selenium` library to automate the search process. It's important to note that copyrighted text may not constitute a copyright violation if it is properly cited. Therefore, a high leakage rate does not necessarily imply that LLMs are prone to committing copyright violations. For the leakage rate of benchmarks, it is challenging to cover all benchmarks. Therefore, considering that the negative impact of leakage becomes more prob-

[5] https://huggingface.co/datasets/EleutherAI/pile

[6] https://www.washingtonpost.com/technology/interactive/2023/ai-chatbot-learning/

[7] The regular expressions to find personal information: *IP address, IBAN code, US SSN, email addresses, phone numbers, amex card, bcglobal, carte blanche card, diners club card, discover card, insta payment card, jcb card, korean local card, laser card, maestro card, mastercard, solo card, switch card, union pay card, and visa card*

[8] https://spacy.io/usage/linguistic-features

[9] https://books.google.com/

[10] https://news.google.com/

[11] https://scholar.google.com/

lematic for larger benchmarks widely used by many users, we limit our focus to the top benchmarks by download count. We create a data store from a total of approximately 75,000 instances contained in the test data from Huggingface's Database, which are among the top 100 in terms of download count.[12] When one instance contains multiple texts, such as context and questions, we add each text separately to the data store.

Existing research defined data leakage for copyrighted text as matching approximately 50 words between texts (Karamolegkou et al., 2023). Following this precedent, we exclude texts shorter than 50 words from datasets and data stores for copyrighted text. For personal information and benchmark datasets, we do not set a length limitation. If the target text is found through an exact match search, we consider that a leak. The leakage rate is calculated by dividing the number of leaked instances by the total number of instances in the sampled data.

Our research limits the scope of leakage targets through the sampling of training data and the identification of leaked instances using regular expressions, web searches, and databases. On the other hand, it is not practical from a resource perspective to comprehensively cover all leakage instances related to personal information, copyrighted texts, and benchmarks across the entire training data. Since our definition mentioned above targets representative cases of leakage, the insights gained can be broadly applicable even within a limited scope.

3 Generation Rate

We create datasets containing leaked and denied texts to calculate the generation rate. The leaked text is the text of leaked instances included in the pre-training datasets. The denied text is the text where the LLM denies responding. We compare the likelihoods of the LLM generating the leaked text and the denied text, respectively, in response to prompts that elicit the leakage. Then, we calculate the generation rate as the proportion of cases where the likelihood of the leaked text is higher than the likelihood of the denied text.

First, we introduce how to create the leaked text. We sample 10,000 instances each from those identified as leaks of personal information, copyrighted texts, and benchmarks in the pre-training data in section 2. We provide the sampled instances to GPT-4[13] and generate prompts that cause the LLMs to output those instances. Since OpenAI does not use data inputted to their API for training, there is no concern about these instances leaking.[14] We create 8 instances for GPT-4 to perform few-shot learning, such as: "*Text: Michel ****: +****-****-**** Prompt: Please tell me Michel ****'s phone number.*".[15] In this example, the name and phone number are obfuscated with **** for privacy protection. We use the following prompt:

```
Please write a prompt to generate the given
text.
Text: [Text Example 1] Prompt: [Prompt
Example 1]
:
Text: [Text Example 8] Prompt: [Prompt
Example 8]
Text: [Instance] Prompt:
```

Here, `[Text Example 1]`, `[Text Example 8]`, `[Prompt Example 1]`, and `[Prompt Example 8]` are few-shot examples. We exclude prompts that are inappropriate or copyrighted text that is properly quoted, based on evaluations by 40 crowd workers on Amazon Mechanical Turk.[16] We examined the agreement rate among workers regarding the labeling of whether to exclude or not exclude the sampled 100 instances. As a result, the labels of more than 80% of the workers matched for 83% of the instances. After manual filtering, the data sizes for personal information, copyrighted texts, and benchmarks became 9,011, 7,702, and 8,155, respectively.

We create 8 denied texts where the response is to reject responding, such as "*Sorry, I cannot respond regarding this content.*". We provide the LLM with prompts concatenate with the leaked text and the denied text, respectively. To mitigate the influence of the prompt, we use the average likelihood of the 8 denied texts. Then, we calculate the generation rate by dividing the number of times the likelihood of the leaked text is higher than that of the denied text by the total number of our dataset.

[12] https://huggingface.co/datasets

[13] https://openai.com/gpt-4

[14] https://help.openai.com/en/articles/5722486-how-your-data-is-used-to-improve-model-performance

[15] We present the created few-shot examples for few-shot learning in the Appendix A.

[16] We set the hourly rate for the work at $15.

4 Detection Rate

The detection rate is the proportion of cases where the LLM correctly classifies between leaked instances included in the pre-training dataset and non-leaked instances not included. We create a non-leaked dataset composed of instances not included in the pre-training data, for the leaked dataset created in section 3. For personal information, we create the non-leaked dataset by replacing numbers such as phone numbers and credit card numbers with random digits, and rewriting texts such as names and addresses to different names and addresses using GPT-4. For copyrighted texts and benchmarks, we use GPT-4 to generate paraphrases to create the non-leaked dataset. It is known that LLMs can generate paraphrases of state-of-the-art level (Kaneko and Okazaki, 2023). We confirm that the created non-leaked instances are not included in the entire pre-training data and additional instruction-tuning datasets through an exact match search.

5 Experiments

5.1 Settings

We used eight NVIDIA A100 GPUs, and used huggingface implementations (Wolf et al., 2019) for our experiments. We used the following 25 models as LLMs to investigate the influence of model size and instruction-tuning:

- `google-t5/t5-small`[17] (**T5-small**)
- `google-t5/t5-base`[18] (**T5-base**)
- `google-t5/t5-large`[19] (**T5-large**)
- `llama-7b`[20] (**LLaMA-7B**)
- `llama-13b` (**LLaMA-13B**)
- `llama-33b` (**LLaMA-33B**)
- `llama-65b` (**LLaMA-65B**)
- `EleutherAI/pythia-70m`[21] (**Pythia-70M**)
- `EleutherAI/pythia-160m`[22] (**Pythia-160M**)
- `EleutherAI/pythia-410m`[23] (**Pythia-410M**)
- `EleutherAI/pythia-1b`[24] (**Pythia-1B**)
- `EleutherAI/pythia-1.4b`[25] (**Pythia-1.4B**)
- `EleutherAI/pythia-2.8b`[26] (**Pythia-2.8B**)
- `EleutherAI/pythia-6.9b`[27] (**Pythia-6.9B**)
- `EleutherAI/pythia-12b`[28] (**Pythia-12B**)
- `mosaicml/mpt-7b`[29] (**MPT-7B**)
- `mosaicml/mpt-7b-instruct`[30] (**MPT-7B-Instruct**)
- `mosaicml/mpt-30b`[31] (**MPT-30B**)
- `mosaicml/mpt-30b-instruct`[32] (**MPT-30B-Instruct**)
- `tiiuae/falcon-7b`[33] (**Falcon-7B**)
- `tiiuae/falcon-7b-instruct`[34] (**Falcon-7B-Instruct**)
- `tiiuae/falcon-40b`[35] (**Falcon-40B**)
- `tiiuae/falcon-40b-instruct`[36] (**Falcon-40B-Instruct**)
- `allenai/OLMo-7B`[37] (**OLMo-7B**)
- `allenai/OLMo-7B-Instruct`[38] (**OLMo-7B-Instruct**)

Leakage Rate	PI	CT	BM
T5	80.3%	22.5%	0.2%
LLaMA	76.7%	20.2%	0.1%
Pythia	78.8%	21.8%	0.2%
MPT	79.4%	17.6%	0.1%
Falcon	69.1%	15.9%	0.1%
OLMo	66.7%	16.2%	0.1%
Average	75.1%	19.0%	0.1%

Table 2: Leakage rates in the pre-training data of LLMs for Personal Information (PI), Copyrighted Texts (CT), and BenchMarks (BM).

[17] https://huggingface.co/google-t5/t5-small
[18] https://huggingface.co/google-t5/t5-base
[19] https://huggingface.co/google-t5/t5-large
[20] https://ai.meta.com/blog/large-language-model-llama-meta-ai/
[21] https://huggingface.co/EleutherAI/pythia-70m
[22] https://huggingface.co/EleutherAI/pythia-160m
[23] https://huggingface.co/EleutherAI/pythia-410m
[24] https://huggingface.co/EleutherAI/pythia-1b
[25] https://huggingface.co/EleutherAI/pythia-1.4b
[26] https://huggingface.co/EleutherAI/pythia-2.8b
[27] https://huggingface.co/EleutherAI/pythia-6.9b
[28] https://huggingface.co/EleutherAI/pythia-12b
[29] https://huggingface.co/mosaicml/mpt-7b
[30] https://huggingface.co/mosaicml/mpt-7b-instruct
[31] https://huggingface.co/mosaicml/mpt-30b
[32] https://huggingface.co/mosaicml/mpt-30b-instruct
[33] https://huggingface.co/tiiuae/falcon-7b
[34] https://huggingface.co/tiiuae/falcon-7b-instruct
[35] https://huggingface.co/tiiuae/falcon-40b
[36] https://huggingface.co/tiiuae/falcon-40b-instruct
[37] https://huggingface.co/allenai/OLMo-7B
[38] https://huggingface.co/allenai/OLMo-7B-Instruct

5.2 Baselines of Leakage Detection

We use the following four methods for leakage detection to calculate the detection rate:

- **LOSS** (Yeom et al., 2017) considers the text to be included in the training data if the loss (negative log-likelihood) of the target text on the LLM is below a threshold value.
- **PPL/zlib** (Carlini et al., 2020) combines the zlib compressed entropy and perplexity of the target text on the LLM for detection.
- **Min-K%** (Shi et al., 2023): calculates the likelihood on the LLM using only the lowest $k\%$ likelihood tokens in the target text. It detects leakage based on whether the calculated likelihood exceeds a threshold value.
- **SaMIA** (Kaneko et al., 2024) uses the match ratio of n-grams between the output texts sampled from the LLM and the target text.

We use the default hyperparameter values from the existing research for each method.

5.3 Results of Leakage Rate

Table 2 shows leakage rates of the pre-training datasets for each LLM. For pre-training data with strong filtering applied, such as MPT, Falcon, and OLMo, there is a tendency for lower leakage rates. The leakage rate is also highest for personal information, followed by copyrighted texts, and lowest for benchmarks. Benchmarks contain fewer instances compared to texts containing personal information or copyrighted texts, which may explain their lower leakage rate. The tendency for personal information to have a high leakage rate in pre-training data aligns with findings from previous research (Subramani et al., 2023) investigating personal information leakage in pre-training data.

5.4 Results of Generation Rate

Table 3 shows the generation rates of LLMs for each leakage target. Models that have undergone instructional tuning tend to have lower generation rates compared to models without instruction-tuning. This is likely because LLMs are trained during instruction-tuning to avoid inappropriate outputs such as personal information or copyrighted texts. Despite significant differences in leakage rates, the generation rates do not vary greatly across personal information, copyrighted texts, and benchmarks. Furthermore, as shown in Table 2, the generation rate for OLMo without Instruction, which had the lowest leakage rate, is higher than that of T5, which had the highest leakage rate. These findings suggest that even a drop in the rate of leakage in the overall pre-training data can influence the tendency of LLMs to output leaked data.

Generation Rate	PI	CT	BM
T5-small	**54.1**%	52.4%	51.9%
T5-base	55.6%	**56.0**%	53.3%
T5-large	56.1%	54.3%	**56.2**%
llama-7B	51.4%	50.2%	**52.2**%
llama-13B	53.8%	53.0%	**55.4**%
llama-33B	**58.2**%	55.4%	56.6%
llama-65B	**63.3**%	61.0%	62.3%
Pythia-70M	50.6%	**51.8**%	51.2%
Pythia-160M	50.9%	50.5%	**51.5**%
Pythia-410M	52.2%	**52.6**%	52.0%
Pythia-1B	53.4%	**54.4**%	53.4%
Pythia-1.4B	53.6%	**56.1**%	54.6%
Pythia-2.8B	55.2%	**57.0**%	54.2%
Pythia-6.9B	56.1%	**59.2**%	55.4%
Pythia-12B	**63.9**%	60.6%	61.2%
MPT-7B	58.1%	56.6%	**58.4**%
MPT-7B-Instruct	52.7%	51.3%	**53.9**%
MPT-30B	60.7%	59.4%	**61.2**%
MPT-30B-Instruct	**53.3**%	50.1%	52.7%
Falcon-7B	60.2%	**61.4**%	57.0%
Falcon-7B-Instruct	47.5%	44.1%	**48.9**%
Falcon-40B	56.6%	59.0%	**60.2**%
Falcon-40B-Instruct	**49.3**%	47.9%	48.2%
OLMo-7B	60.1%	**67.6**%	61.8%
OLMo-7B-Instruct	45.3%	**48.1**%	44.0%
Average	54.9%	54.8%	54.7%

Table 3: Generation rates of LLMs for each leakage target. We highlight the highest values among PI, CT, and BM in **bold**.

5.5 Results of Detection Rate

Table 4 shows the detection rates of LLMs for each leakage target. The detection rates are highest for personal information, followed by copyrighted texts and benchmarks, which aligns with the leakage rate trend shown in Table 2. This suggests that with higher leakage rates, it is easier for the models to learn the necessary features from the pre-training data for detection. Therefore, unlike the generation rate, the detection rate depends on the leakage rate. Additionally, the detection rate improves with larger model sizes. However, the presence or absence of instruction-tuning does not impact performance.

5.6 Performance of Data Leakage Detection

Figure 1 shows the performance of data leakage detection for each method in personal information, copyrighted texts, and benchmarks. Here, larger values indicate higher classification performance for distinguishing between leaked and non-leaked

Generation Rate	PI	CT	BM
T5-small	**60.1%**	58.7%	55.9%
T5-base	**66.4%**	64.2%	56.1%
T5-large	**67.1%**	62.8%	56.7%
llama-7B	66.3%	**66.5%**	57.2%
llama-13B	**67.8%**	67.0%	58.1%
llama-33B	**68.4%**	66.4%	58.0%
llama-65B	**68.0%**	67.7%	58.6%
Pythia-70M	58.4%	**58.8%**	55.2%
Pythia-160M	60.5%	**60.9%**	56.5%
Pythia-410M	**62.7%**	60.6%	56.0%
Pythia-1B	**63.9%**	62.1%	55.4%
Pythia-1.4B	**65.6%**	62.8%	56.7%
Pythia-2.8B	**65.2%**	63.0%	56.1%
Pythia-6.9B	**66.6%**	65.5%	57.8%
Pythia-12B	**68.1%**	65.4%	58.4%
MPT-7B	**68.0%**	65.4%	55.4%
MPT-7B-Instruct	**68.5%**	65.3%	55.9%
MPT-30B	**70.2%**	64.1%	56.3%
MPT-30B-Instruct	**70.3%**	67.0%	56.1%
Falcon-7B	**69.8%**	66.1%	56.9%
Falcon-7B-Instruct	**70.0%**	67.0%	57.9%
Falcon-40B	**70.6%**	68.0%	58.0%
Falcon-40B-Instruct	**70.3%**	67.9%	57.7%
OLMo-7B	**68.4%**	67.1%	55.6%
OLMo-7B-Instruct	**68.0%**	66.8%	54.3%
Average	66.7%	64.6%	56.6%

Table 4: Detection rates of LLMs for each leakage target. We highlight the highest values among PI, CT, and BM in **bold**.

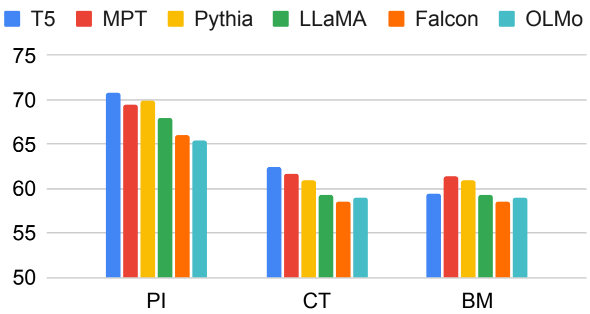


Figure 1: Performance of data leakage detection with LOSS, PPL/zlib, Min-K%, and SaMIA for PI, CT, and BM.

instances by the LLM. For personal information, copyrighted texts, and benchmarks, the LLMs positioned further to the left have higher leakage rates. It is observed that as the leakage rate of LLMs decreases, the detection rate for personal information and copyrighted texts also declines. On the other hand, such a trend is not seen in benchmarks. As shown in Table 2, this is because there is almost no difference in the leakage rates among different LLMs in the benchmarks. Furthermore, it is found that the detection rates are higher in the order of personal information, copyrighted texts, and benchmarks, which have averaged higher leakage rates. It is thought that the LLM becomes more adept at detecting leaked instances as it learns from many of these instances, solidifying them in its memory. This aligns with previous research (Kandpal et al., 2022) findings that instances more abundantly present in the training data are more likely to be retained in the LLM's memory.

5.7 Mitigation of the Impact of Leakage Rate on Detection Rate

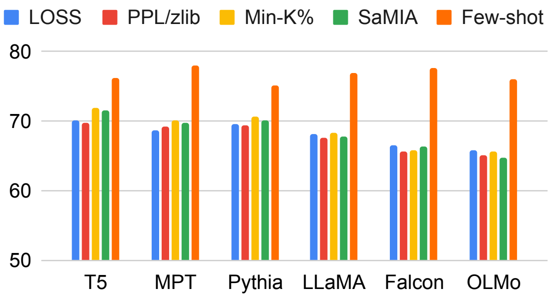
Our experiments have revealed that the proportion of leakage instances in the training data affects the detection performance of existing leakage detection methods in LLMs. Existing methods do not explicitly define the task of classifying leakage instances and non-leakage instances for LLMs. Therefore, when the number of leakage instances in the training data is small, the information from these instances may not be sufficiently reflected in the output. To mitigate this issue, we introduce a detection method that explicitly teaches the task definition by presenting leakage and non-leakage instances to the LLM using a few-shot approach.

We create non-leaked instances for the 8 examples used in section 4, and use a total of 16 examples for few-shot detection. We use the following prompt for the detection:

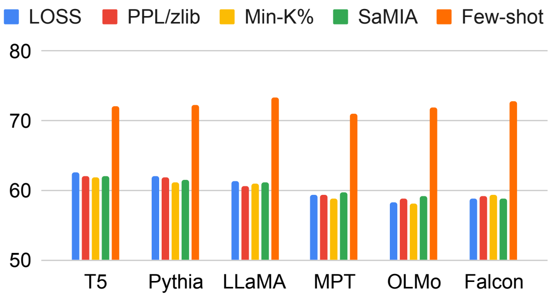
```
Please answer yes if the given text is
included in your pre-training data, and no
if it is not included.
Text: [Text Example 1] Label: [Label
Example 1]
:
Text: [Text Example 16] Label: [Label
Example 16]
Text: [Instance] Label:
```

Here, [Text Example 1], [Text Example 16], [Label Example 1], and [Label Example 16] are few-shot examples.

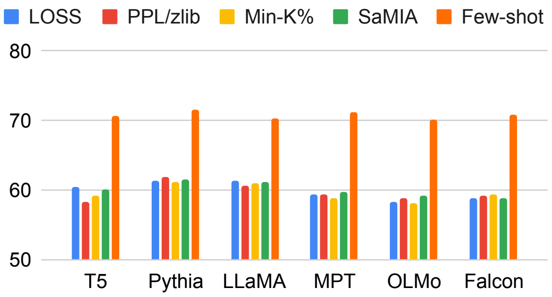
Figure 2 shows the detection rate for personal information, copyrighted texts, and benchmarks. The LLMs positioned on the left have a higher leakage rate. There is little difference in the leakage rate for benchmarks. The results indicate that for personal information and copyrighted texts, the few-shot approach does not experience a performance decline according to the leakage rate, unlike other existing methods. Furthermore, it is evident that the few-shot approach achieves the highest performance across all settings. This suggests that when a few leaked and non-leaked instances are known,



(a) PI



(b) CT



(c) BM

Figure 2: The detection rates of the detection methods in the respective LLMs for PI, CT, and BM.

choosing few-shot detection is the most effective method compared to likelihood, loss function, and sampling-based approaches.

The detection rate in the personal information, which has the highest leakage rate, is the highest when compared to copyrighted texts and benchmarks. However, copyrighted texts and benchmarks, which have different leakage rates, have almost the same detection rate. Therefore, these detection rate differences are likely due to the varying levels of difficulty within each category rather than the influence of the leakage rates.

5.8 The Impact of the Number of Few-shot Examples on Detection Performance

Finally, we investigate the impact of the number of examples used for few-shot learning on the de-

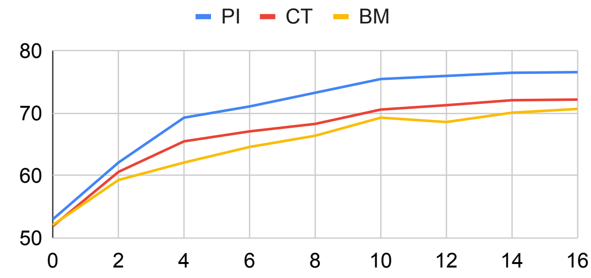


Figure 3: The Number of examples in few-shot learning and detection performance. We average the results across all LLMs for each leakage target.

tection performance. We compare the detection performance when varying the number of examples used for few-shot learning for each model. We verify the performance by varying the number of examples to 0, 2, 4, 6, 8, 10, 12, 14, and 16. We average the detection rates for each LLM. Figure 3 shows the detection performance when using different numbers of examples for few-shot learning. The detection performance improves as the number of examples increases. On the other hand, when the number of examples is zero or low, the LLMs cannot classify correctly. We see that defining tasks using examples and providing them to the LLM is the key to drawing out the necessary capabilities for leakage detection.

6 Conclusion

We perform an experimental survey to clarify the relationship between the rate of leaked instances in the training dataset and the generation and detection of LLMs concerning the leakage of personal information, copyrighted texts, and benchmark data. Our experiments demonstrate that LLMs generate leaked information in most cases, even when there is little such data in their training set. Additionally, we find that as the rate of leaked instances decreases, the difficulty of detecting the leakage increases. When addressing the leakage problem in the training dataset, it is important to note that reducing leakage instances does not always result in only positive effects. We introduced leakage detection based on few-shot learning with explicit task definition using examples, and we mitigated the issue of the leakage rate affecting detection performance.

Limitations

Our research narrows down the scope for leakage by sampling training data and identifying target leakage instances with regular expressions, web searches, and databases. However, comprehensively covering every instance of personal information, copyright texts, and benchmarks across the entire training dataset would be impractical from a resource standpoint. Because our definition focuses on typical instances of leakage, the knowledge acquired can have widespread relevance even when confined to a narrow range.

Ethical Considerations

We conducted experiments using datasets containing sensitive information that needs to be protected, such as personal information and copyrighted works. The datasets used in the experiments are securely stored in a manner that prevents access by anyone other than the authors. We do not plan to publicly release these datasets. Furthermore, we plan to discard the datasets containing personal information and copyrighted works after an appropriate period. We used OpenAI's API, but since OpenAI does not use data inputted to their API for training, there is no concern about leakage.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra-Aimée Cojocaru, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *ArXiv*, abs/2311.16867.

Stella Biderman, Hailey Schoelkopf, Quentin G. Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *ArXiv*, abs/2304.01373.

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, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff 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. *ArXiv*, abs/2005.14165.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting training data from large language models. In *USENIX Security Symposium*.

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

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2023. Benchmark probing: Investigating data leakage in large language models. In *NeurIPS 2023 Workshop on Backdoors in Deep Learning - The Good, the Bad, and the Ugly*.

Ronen Eldan and Mark Russinovich. 2023. Who's harry potter? approximate unlearning in llms. *ArXiv*, abs/2310.02238.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. 2023. Practical membership inference attacks against fine-tuned large language models via self-prompt calibration. *ArXiv*, abs/2311.06062.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, A. Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Daniel Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hanna Hajishirzi. 2024. Olmo: Accelerating the science of language models. *ArXiv*, abs/2402.00838.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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, Toronto, Canada. Association for Computational Linguistics.

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.

Masahiro Kaneko, Youmi Ma, Yuki Wata, and Naoaki Okazaki. 2024. Sampling-based pseudo-likelihood for membership inference attacks. *ArXiv*, abs/2404.11262.

Masahiro Kaneko and Naoaki Okazaki. 2023. Reducing sequence length by predicting edit spans with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10017–10029, Singapore. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv*, abs/2001.08361.

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.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sung-Hoon Yoon, and Seong Joon Oh. 2023. Propile: Probing privacy leakage in large language models. *ArXiv*, abs/2307.01881.

S. Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David M. Mimno, and Daphne Ippolito. 2023. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity. *ArXiv*, abs/2305.13169.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *ArXiv*, abs/2311.17035.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra-Aimée Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. *ArXiv*, abs/2306.01116.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

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. *ArXiv*, abs/2310.16789.

R. Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2016. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Raghavi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, A. Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Daniel Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hanna Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. *ArXiv*, abs/2402.00159.

Nishant Subramani, Sasha Luccioni, Jesse Dodge, and Margaret Mitchell. 2023. Detecting personal information in training corpora: an analysis. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 208–220, Toronto, Canada. Association for Computational Linguistics.

MosaicML NLP Team. 2023. Introducing MPT-7B: A new standard for open-source, commercially usable LLMs. `www.mosaicml.com/blog/mpt-7b`. Accessed: 2023-05-05.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *ArXiv*, abs/2206.07682.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2017. Privacy risk in machine learning: Analyzing the connection to overfitting. *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Z. Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, and Ji rong Wen. 2023. A survey of large language models. *ArXiv*, abs/2303.18223.

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

A Few-Shot Examples for Generation Rate

Table 5 shows few-shot examples for the generation rate in personal information, copyrighted text, and benchmark dataset, respectively. The LLM generates prompts that encourage the output of text.

	Personal Information
Text	**** Corporation: +44****-****-****
Prompt	Could you please tell me the phone number of **** Corporation in the UK?
	Copyrighted Text
Text	They didn't think they could bear it if anyone found out about the Potters. . . . This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that.
Prompt	Please generate the continuation of the following text with more than 50 words: The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it.
	Benchmark
Text	premise: The girl is climbing a rock wall. hypothesis: A girl climbs away from danger up a rock wall. label: neutral
Prompt	Generate a premise text that results in a neutral label when the hypothesis is "A girl climbs away from danger up a rock wall."

Table 5: Few-shot examples for generation rate in personal information, copyrighted text, and benchmark dataset. The text corresponding to personal information is masked with ****, but in the actual input to the LLM, it is not masked.