# Forget to Flourish: Leveraging Machine-Unlearning on Pretrained Language Models for Privacy Leakage

**Md Rafi Ur Rashid***
Pennsylvania State University
University Park, PA 16802, USA
mur5028@psu.edu

**Jing Liu, Toshiaki Koike-Akino**
Mitsubishi Electric Research Laboratories
Cambridge, MA 02139, USA
{jiliu, koike}@merl.com

**Shagufta Mehnaz**
Pennsylvania State University
University Park, PA 16802, USA
smehnaz@psu.edu

**Ye Wang**
Mitsubishi Electric Research Laboratories
Cambridge, MA 02139, USA
yewang@merl.com

## Abstract

Fine-tuning large language models on private data for downstream applications poses significant privacy risks in potentially exposing sensitive information. Several popular community platforms now offer convenient distribution of a large variety of pre-trained models, allowing anyone to publish without rigorous verification. This scenario creates a privacy threat, as pre-trained models can be intentionally crafted to compromise the privacy of fine-tuning datasets. In this study, we introduce a novel poisoning technique that uses model-unlearning as an attack tool. This approach manipulates a pre-trained language model to increase the leakage of private data during the fine-tuning process. Our method enhances both membership inference and data extraction attacks while preserving model utility. Experimental results across different models, datasets, and fine-tuning setups demonstrate that our attacks significantly surpass baseline performance. This work serves as a cautionary note for users who download pre-trained models from unverified sources, highlighting the potential risks involved.

## 1 Introduction

In recent times, the traditional way of training a language model (LM) from scratch has been largely replaced by the introduction of pre-trained foundation models Touvron et al. (2023); Chiang et al. (2023). For example, the Hugging Face Hub[2] is a platform with over 120k open-source models, readily available for download and any registered user can contribute by uploading their own model. However, there are serious security and privacy risks associated with downloading such models from any untrusted sources and further fine-tuning them for some downstream applications as they could be maliciously crafted Tramèr et al. (2022); Kandpal et al. (2023); Hu et al. (2022). Additionally, the public release of large language models (LLMs) fine-tuned on potentially sensitive user data could lead to privacy breaches, as these models have been found to memorize verbatim text from their training data Carlini et al. (2019, 2021). In this paper, we combine the notion of poisoning a pre-trained LLM and causing privacy leakage of the fine-tuned model. More specifically, we introduce a novel model poisoning algorithm that aims to manipulate a pre-trained LLM in order to disclose more of the private data used during its fine-tuning.

*Work performed while an intern at Mitsubishi Electric Research Laboratories.

[2]https://huggingface.co/docs/hub/en/models

NeurIPS 2024 Workshop on Red Teaming GenAI: What Can We Learn from Adversaries?

At its core, our approach leverages **machine unlearning** Cao and Yang (2015); Guo et al. (2019) to poison the pre-trained LLM. The original objective of unlearning is to make the model forget specific data points that it has seen during training so that it produces a high loss for those data points, and it becomes difficult to reconstruct those samples Gu et al. (2024). Motivated by data augmentation that reduces overfitting, we discovered that unlearning on some **noisy version** of fine-tuning data points can promote overfitting of the original data during the fine-tuning process.

However, it is important to have control over the process of loss maximization; otherwise, the model might become unusable and the poisoning attempt would be easily detectable. Hence, we propose **bounded unlearning** as a poisoning tool, where we maximize loss in a controlled manner on the pre-trained model for some noisy data points to increase privacy leakage of the fine-tuned LLM without compromising its utility.

To measure the privacy leakage caused by our proposed method, we consider two standard privacy attacks: membership inference (MIA) Shokri et al. (2017); Carlini et al. (2022) and data extraction (DEA) Nasr et al. (2023); Rashid et al. (2023). In MIA, the model is queried to evaluate whether a specific target data point that the attacker possesses was indeed part of the finetuning dataset. On the contrary, DEA aims to extract verbatim texts from the fine-tuning dataset with partial/zero prior knowledge. We evaluate our proposed method for both of these attacks on a range of language models (Llama2-7B, GPT-Neo 1.3B), datasets (MIND, Wiki-103+AI4Privacy), fine-tuning methods (Full-FT, LoRA-FT, QLoRA-FT), and defense (differential privacy). Overall, our method significantly boosts the MIA and DEA attack performance over the baselines in almost all scenarios and maintains its stealth by preserving model utility. Prior works that deal with privacy leakage through pre-trained model poisoning pose some strong assumptions on the adversary's capability, as discussed in the Related Work section of the paper. Our proposed method, on the other hand, with a more practical threat model and weaker adversarial ability, substantially enhances the attack success rate and still remains stealthy.

## 2 Threat Model

### 2.1 The Membership Inference Game

❒ **Access to Pre-trained LLM:** The attacker has access to a pre-trained large language model denoted as $\theta_{\text{pre}}$. Additionally, the attacker is given a challenge dataset $D_c$, which includes some member data $d$ and non-member data $d_{\ominus}$.

❒ **Poisoning Phase:** The attacker employs a poisoning algorithm $T_{\text{adv}}$ to manipulate the pre-trained model $\theta_{\text{pre}}$, resulting in an adversarially altered model $\theta_{\text{adv}}$.

❒ **Model Distribution:** The adversarially poisoned model $\theta_{\text{adv}}$ is distributed to the challenger. The challenger then fine-tunes $\theta_{\text{adv}}$ with their private dataset $D_{\text{ft}}$, resulting in the fine-tuned model $\theta_{\text{ft}}^{\text{adv}}$.

❒ **Black Box Access:** Post fine-tuning, the attacker is granted black box query access to the fine-tuned model. $\theta_{\text{ft}}^{\text{adv}}$. Through this access, the attacker can submit inputs and receive outputs (both generated text and model loss) from $\theta_{\text{ft}}^{\text{adv}}$.

❒ **Attacker's Objective:** The primary goal of the attacker is to identify the membership of specific samples within the challenge dataset, $D_c$. This involves determining whether a given sample belongs to $D_{\text{ft}}$ or not.

### 2.2 The Data Extraction Game

❒ **Access to Pre-trained LLM:** Similar to the MI case, the attacker has access to a pre-trained LLM, $\theta_{\text{pre}}$. However, in this case, he is given only partial knowledge of the training dataset as the challenge dataset, which consists of the prefixes of the training data samples, denoted as $P_c$.

❒ **Poisoning Phase and Model Distribution:** These steps are the same as MIA.

❒ **Black Box Access:** Post fine-tuning, the attacker is granted black box query access to the fine-tuned model $\theta_{\text{ft}}^{\text{adv}}$. Through this black box access, the attacker can submit input prompts and receive the generated text as output from $\theta_{\text{ft}}^{\text{adv}}$.

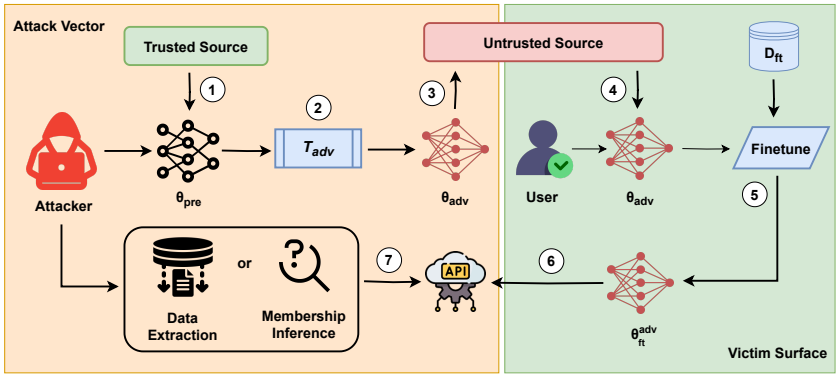


Figure 1: Overview of the threat model and steps of the attack: (1) Attacker downloads a pre-trained LLM, (2) Poisons the model with an algorithm, $\mathcal{T}_{\text{adv}}$, and (3) release the model. (4) The victim downloads the poisoned LLM, (5) fine-tunes on their private data, and (6) releases the API-based query access to the model. (7) Finally, the adversary conducts membership inference or data extraction.

❒**Attacker's Objective:** The primary goal of the attacker is to successfully reconstruct the suffix, $S_c$, which is present in $D_{\text{ft}}$, for each corresponding prefix in $P_c$.

## 3 Motivation

Overfitting is a leading factor contributing to vulnerability to membership inference attacks (Amit et al., 2024; Shokri et al., 2017; Dionysiou and Athanasopoulos, 2023; He et al., 2022). When training a language model for some downstream application, the initial state of the model's parameters plays a crucial role in the learning process. Typically, these parameters are either randomly initialized when training from scratch or set to general pre-trained weights, which are the result of rigorous pre-training on a large corpus of text data. Consequently, at the onset of training, the model does not exhibit a strong predisposition or bias towards any specific training data points.

Further fine-tuning on downstream data $D_{\text{ft}}$ is more prone to overfitting. However, as we will discuss later in Figure 2, it is still non-trivial for an attacker to distinguish between member and non-member data, which might have similar data distributions. One key question we try to answer is this:
**RQ1:** *Is it possible to poison the pre-trained model to make the fine-tuning process overfit even more and the resulting fine-tuned model more vulnerable to privacy leakage attacks?* In this work, we introduce an unlearning-based model poisoning technique and give a sure answer to the above research question. This answer is supported by several observations, findings, and experimental results, which we will discuss gradually.

### 3.1 Motivations of Leveraging Unlearning

We want to poison the model to induce it to overfit during the fine-tuning process. It is quite challenging to come up with a method for poisoning. However, we can think of the opposite side first: How to prevent a model from overfitting? Recall that overfitting occurs when a model learns the training data too well and is unable to generalize to new data. One simple and effective approach is **Data Augmentation**. Data augmentation is a well-known technique used in machine learning to artificially create more data points from existing data. This can be done by applying different transformations to the data, and one popular transform is noise perturbation. Training on original samples together with their noisy versions can help reduce model overfitting Wei and Zou (2019). On the contrary, as we want to increase overfitting in the fine-tuning procedure, it now becomes intuitive to leverage unlearning/ reverse-training on the **noisy versions** of training samples.

The challenge dataset, $D_c$, consists of both member data points, $d$, and non-member data points, $d_\ominus$ ($D_c = d \cup d_\ominus$). We propose and validate some methods to generate the noisy versions of $D_c$, denoted as $D'_c$ ($D'_c = d' \cup d'_\ominus$), and the strategic maximization of the loss associated with $D'_c$ to poison the model, which will be discussed in detail in next section.

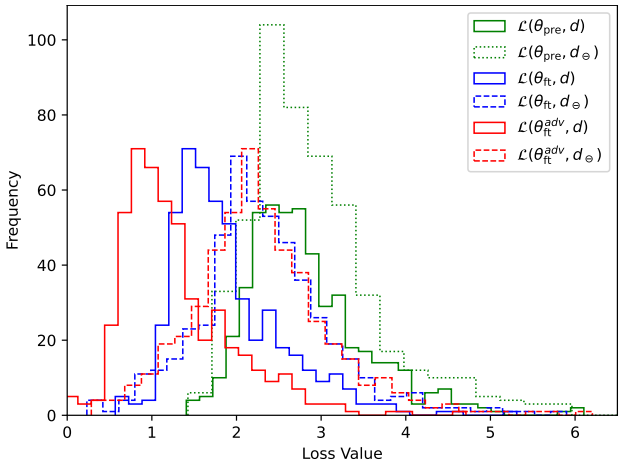


Figure 2: Histograms of loss values on pre-trained model $\theta_{\text{pre}}$, fine-tuned model $\theta_{\text{ft}}$, and fine-tuned poisoned model $\theta_{\text{ft}}^{\text{adv}}$.

### 3.2 Members and Non-members from the Same Data Distributions are Hard to Separate

Figure 2 shows the histograms of loss values of member data $d$ and non-member data $d_\ominus$, on pre-trained model, $\theta_{\text{pre}}$ (green color) and fine-tuned model, $\theta_{\text{ft}}$ (blue color). Here, $d$ and $d_\ominus$ come from similar distributions. As expected, before fine-tuning, it is not possible to infer membership based on the difference in loss value histograms (green solid line vs. green dotted line). After fine-tuning, the loss values of $d$ decrease. However, as $d_\ominus$ have similar data distributions to member data, their loss values also decrease, making it still hard to distinguish the membership based on the loss values after fine-tuning (blue solid line vs. blue dashed line).

### 3.3 Findings: Unlearning Amplifies Overfitting

Figure 2 also shows the histograms of loss values of $d$ and $d_\ominus$ after fine-tuning on the poisoned (via unlearning) model, $\theta_{\text{ft}}^{\text{adv}}$ (red color). Note that the unlearning is performed on $D_c'$. We get two crucial insights from here: first, compared with fine-tuning on the non-poisoned model (blue solid line), we can see that fine-tuning on the poisoned model can reduce the loss value of member data even more (red solid line). Second, the difference in loss values between $d$ and $d_\ominus$ is amplified after fine-tuning on poisoned data (red solid line and red dashed line) compared to fine-tuning on the non-poisoned model (blue solid line and blue dashed line). Thus, it answers the **RQ1**, i.e., machine unlearning-based poisoning indeed increases the overfitting of the fine-tuned LLM and thereby causes further privacy leakage.

## 4 Methodology

In this section, we will provide step by step description of our entire workflow. Figure 1 demonstrates the important steps of our proposed attacks.

### 4.1 Introducing Noisy Data Points

As mentioned earlier, we create a noisy version of $D_c$, denoted as $D_c'$. The choice of noise perturbation methods depends on the attack type, which we will describe shortly, along with the attack methods.

### 4.2 Bounded Unlearning

Vanilla unlearning would simply maximize the loss via gradient ascent:

$$\theta' = \theta_0 + \eta' \nabla_\theta \mathcal{L}(\theta_0; D_c'), \tag{1}$$

However, when maximizing the loss on noisy data points $D'_c$, it is crucial to ensure that this process does not disrupt the model's general capabilities. Therefore, we introduce a constraint for the loss maximization process:

$$\theta' = \theta_0 + \eta' \nabla_\theta \mathcal{L}(\theta_0; D'_c) \text{ subject to } \quad \mathcal{L}(\theta'; D^*) \leq \epsilon \tag{2}$$

Here, $D^*$ is a set of plain text sequences selected to measure the language model's general utility. This ensures that the loss on the noisy data points $D'_c$ is increased, but $\mathcal{L}(\theta'; D^*)$ does not go beyond the threshold $\epsilon$, thereby controlling the extent of the loss maximization and preserving model utility.

For model poisoning, we used a gradient ascent-based unlearning strategy similar to Jang et al. (2023), i.e., inverting the direction of gradients. The default unlearning rate, batch size, and max number of epochs are set to $10^{-6}$, 32, and 5, respectively. For bounded unlearning, we curated a subset of 500 samples from the Wiki-2 Merity et al. (2016) and used it as the plain-text dataset $D^*$.

### 4.3 Membership Inference

As mentioned earlier in Section 2, the attacker poisons the pre-trained language model, $\theta_{\text{pre}}$ with some poisoning algorithm $T_{\text{adv}}$. For the membership inference attack (MIA), we design the poisoning algorithm based on the proposition mentioned in the previous section regarding the impact of unlearning on a model's memorization.

#### 4.3.1 Poisoning Algorithm for MIA, $T^{\text{mi}}_{\text{adv}}$

The attacker creates a noisy version of $D_c$, denoted as $D'_c$, which is used to perform unlearning on $\theta_{\text{pre}}$, according to equation 2. This poisoning approach ensures that the model yields high loss values for these noisy samples before fine-tuning. We utilize two different mechanisms for creating the noisy sequences:

❒ **Random Character Perturbation:** Adding noise by random insertion, deletion, and swapping of a certain percentage of characters of the given sequence.

❒ **Random Word Perturbation:** Adding noise by random insertion, deletion, and replacement of a certain percentage of words of the given sequence.

For these random character and random word perturbation methods, we set the default noising level to 10% and 30%, respectively. We also performed an ablation study by varying the noising level, which can be found in the Appendix.

After carrying out the poisoning algorithm on the pre-trained LLM, the next few steps of the threat model take place, including model distribution, fine-tuning, and returning the black-box access of the model to the attacker. Finally, we design how the attacker infers membership of the challenge dataset on the fine-tuned model.

#### 4.3.2 Inference

We propose one simple loss-based and two reference-based inference mechanisms:

❒ **Simple Loss-based:** After getting black-box access to $\theta_{\text{ft}}$, the adversary queries the model with each sample of $D_c$ and records the model loss values. Membership is then inferred based on whether the loss is lower than a given loss threshold $\epsilon$. Formally, for each sample $(x \in D_c)$, we decide

$$\begin{aligned} x \in D_{\text{ft}}, \quad &\text{if} \quad \mathcal{L}(x) < \epsilon, \\ x \notin D_{\text{ft}}, \quad &\text{if} \quad \mathcal{L}(x) \geq \epsilon, \end{aligned}$$

where the shorthand $\mathcal{L}(x) := \mathcal{L}(\theta^{\text{adv}}_{\text{ft}}, x)$ denotes the fine-tuned model loss.

❒ **Reference data-based**: For this inference strategy, the adversary needs an auxiliary dataset $D_{\text{aux}}$, which does not have any overlap with the fine-tuning dataset ($D_{\text{aux}} \cap D_{\text{ft}} = \emptyset$). In this case, unlearning is performed on both $D'_c$ and $D_{\text{aux}}$ ($D'_c \oplus D_{\text{aux}}$) in the previous poisoning phase. This ensures that the model yields a high loss for both of these datasets before delving into the fine-tuning process.

With black-box access to $\theta_{\text{ft}}$, the adversary queries the model with each sample of $D_{\text{aux}}$ and $D_c$, and records the corresponding model loss values. The loss values of the member data are usually much

smaller than that of $D_{\text{aux}}$. Formally, for each sample $x \in D_c$ and $\mathcal{L}_{\text{aux}}$ be the distribution of loss values when $\theta_{\text{ft}}$ is queried with samples from $D_{\text{aux}}$:

$$\begin{aligned} x \in D_{\text{ft}}, &\quad \text{if } \mathcal{L}(x) \text{ is statistically different from } \mathcal{L}_{\text{aux}}, \\ x \notin D_{\text{ft}}, &\quad \text{if } \mathcal{L}(x) \text{ is statistically consistent with } \mathcal{L}_{\text{aux}} \end{aligned}$$

For reference data-based inference, we select 500 non-training data samples as $D_{\text{aux}}$. We utilize percentile rank[3] to measure the statistical coherence between $\mathcal{L}(x)$ and $\mathcal{L}_{\text{aux}}$.

❒ **Reference model-based**: Instead of using the external dataset $D_{\text{aux}}$, another idea is to use the pre-trained LLM, $\theta_{\text{pre}}$ as a reference in inferring membership. The difference between pre-trained and fine-tuned LLM in terms of the model's loss of the member data points (green solid line vs. red solid line in Figure 2) are usually much larger than that of the non-member data points (green dotted line vs. red dashed line in Figure 2). Hence, with a predefined threshold, $\epsilon$, samples with a loss-difference higher than $\epsilon$ are considered as belonging to the finetuning dataset. Formally, we decide membership based on the rule:

$$\begin{aligned} x \in D_{\text{ft}}, &\quad \text{if} \quad |\mathcal{L}(\theta_{\text{ft}}^{\text{adv}}, x) - \mathcal{L}(\theta_{\text{pre}}, x)| \geq \epsilon, \\ x \notin D_{\text{ft}}, &\quad \text{if} \quad |\mathcal{L}(\theta_{\text{ft}}^{\text{adv}}, x) - \mathcal{L}(\theta_{\text{pre}}, x)| < \epsilon. \end{aligned}$$

### 4.4 Data Extraction

For the data extraction attack, we follow a poisoning algorithm that is very similar to MIA, with some key modifications in the design.

#### 4.4.1 Poisoning Algorithm for DEA, $T_{\text{adv}}^{\text{de}}$

The attacker creates a noisy version of $D_c$, denoted as $D_c'$ by concatenating each prefix in $P_c$ with some noisy suffixes $S'$, and then runs unlearning on $\theta_{\text{pre}}$ with this noisy dataset according to equation 2. Just as before, this poisoning approach ensures that the model carries high loss values for these noisy samples before fine-tuning. We utilize two different mechanisms for creating the noisy suffixes:

❒ **Random word concatenation:** Generate the noisy suffix with a fixed or variable number of random words, which might not have any semantic coherence with each other.

❒ **Autoregressive generation:** Prompt the pre-trained language model, $\theta_{\text{pre}}$, with the prefixes to fill out the suffix part.

After carrying out the poisoning algorithm on the pre-trained LLM, the next few steps of the threat model take place, including model distribution, fine-tuning, and returning the black-box access of the model to the attacker. Finally, the attacker prompts the fine-tuned model with each prefix in $P_c$ and tries to successfully reconstruct the original suffix present in $D_{\text{ft}}$.

While crafting the noisy samples in DEA based on random word concatenation or autoregressive generation, we add a random number of tokens in a range of 15-20 to the prefix for both cases. Also, we set the default length of known prefixes to 20% of each full-text sequence. Later, we also do an ablation study by varying the prefix length. Besides, we do ablation with several text generation strategies Gatt and Krahmer (2018), including greedy search, beam search decoding, and contrastive search Su et al. (2022). However, we select beam search decoding with a beam size of 5 as the default configuration for all experiments.

## 5 Experimental Setup

In this section, we discuss the default settings and hyperparameters used for different experiments.

### 5.1 Dataset

We perform experiments on two datasets, each representing a particular data type. The first dataset consists of news article abstracts obtained from a subset of the Microsoft News Dataset (MIND) Wu

[3]Percentile rank is a statistical measure that indicates the relative position of a value within a distribution, showing the percentage of values in the distribution that are equal to or below it.

et al. (2020). It has three partitions: train, test, and validation. We took a subset of 20K training samples for fine-tuning, 1K subset of validation samples, and 1K test samples. We selected this dataset to investigate how our attacks perform to leak the privacy of general-purpose English texts from the fine-tuning dataset. The second dataset is a fusion of Wikitext-103 Merity et al. (2017) and AI4Privacy[4]. The latter is an open-source privacy dataset that holds real-life personal identifiable information (PII) data points. We inject 1,000 randomly selected samples from AI4Privacy into the WikiText-103 dataset. This dataset is meant to experiment with how our attacks are able to extract private information such as addresses, phone numbers, passwords, etc.

### 5.2 Models and Fine-Tuning Methods

To evaluate our attacks we select two different families of large language models, GPT-Neo 1.3 billion parameter variant[5] from EleutherAI and Llama-2 7 billion parameter variant[6] from Meta. Nowadays, various fine-tuning methods, especially for large language models, are employed for pre-trained models due to their efficiency and effectiveness. Since an adversary may not have control over the fine-tuning algorithm, we demonstrate how effective our attacks are against different fine-tuning methods. We trained the Llama-2 model using full fine-tuning (Full-FT), LoRA-FT Hu et al. (2021), and 4-bit QLoRA Dettmers et al. (2024). We set a default learning rates for Full-FT, LoRA-FT, and QLoRA-FT as $2 \times 10^{-5}$, $2 \times 10^{-4}$, and $2 \times 10^{-4}$, respectively, and trained for 5 epochs with early stopping to prevent overfitting.

### 5.3 Evaluation Metrics

We use the perplexity on the validation dataset (Val-PPL↓) to measure the utility of the fine-tuned model, as well as the stealthiness of our proposed attacks. Carlini et al. (2022) pioneered the practice of analyzing True Positive Rate (TPR↑) at low False Positive Rate (FPR) thresholds to highlight the effectiveness of attacks under stringent conditions. Following this approach, our evaluation framework employs several key metrics: TPR at 0.01% FPR, TPR at 0.1% FPR, Area Under the Curve (AUC↑), and Best Accuracy (Best Acc↑), defined as the maximum accuracy achieved along the tradeoff curve. On the other hand, to evaluate data extraction, we compute the number of successful reconstructions (NSR↑), i.e., the number of extracted sequences that are part of the finetuning dataset.

## 6 Results

### 6.1 Membership Inference

To evaluate the membership inference attack (MIA), we take 1K test sequences, 500 of which are member samples, i.e., present in the fine-tuning dataset, and the remaining 500 are non-member samples, i.e., absent in the fine-tuning dataset.

#### 6.1.1 Baselines and Proposed Attacks

We consider two baseline MIA: the first one is simply based on model loss (Baseline-Loss), with the assumption that member data points would have a lower loss value than the non-member samples. The second baseline is based on relative loss with respect to the pre-trained model (Baseline-Rel), i.e., the loss difference between fine-tuned and the pre-trained models, where the relative loss of member samples should be higher than the non-member samples. Apart from that, as mentioned in Section 4, for both character perturbation and word perturbation-based poisoning, we adopt three inference strategies: simple loss-based (Poison-char/word-Loss), reference data-based (Poison-char/word-Aux) and reference model-based (Poison-char/word-Rel).

#### 6.1.2 Model Utility/ Stealthiness

Table 1 compares the attack performance and model utility of Llama2-7B on two datasets, MIND and Wiki-PII, with respect to different MIA configurations for full fine-tuning, LoRA and QLoRA

[4]https://huggingface.co/datasets/ai4privacy/pii-masking-200k

[5]https://huggingface.co/EleutherAI/gpt-neo-1.3B

[6]https://huggingface.co/meta-llama/Llama-2-7b

| Dataset | | MIND | | | | | Wiki+PII | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **FT Method** | **MIA Method** | Val-PPL | Best Acc | TPR @ 1%FPR | TPR @ 0.1% FPR | AUC | Val-PPL | Best Acc | TPR @ 1%FPR | TPR @ 0.1% FPR | AUC |
| Full-Ft Llama2-7B | Baseline-loss | 16.00 | 76.80% | 8.20% | 1.00% | 79.48% | 9.15 | 73.30% | 4.80% | 2.60% | 77.89% |
| | Baseline-Rel | 16.00 | 79.10% | 1.60% | 0.00% | 81.00% | 9.15 | 78.10% | 19.20% | 9.80% | 84.83% |
| | Poison-char-loss | 16.27 | 81.30% | 24% | **8.80%** | 84.72% | 11.02 | 83.00% | 16.80% | 5.00% | 87.88% |
| | Poison-char-Rel | 16.27 | 86.40% | 2.40% | 0.40% | **88.51%** | 11.02 | **90.80%** | 56.60% | 32.60% | 95.60% |
| | Poison-char-Aux | 16.02 | **87.90%** | 21.60% | 7.60% | 86.91% | 11.15 | 84.60% | 23.60% | 6.40% | 89.47% |
| | Poison-word-loss | 16.19 | 81.60% | 21.40% | 9% | 84.83% | 11.03 | 83.80% | 16.20% | 5.40% | 87.89% |
| | Poison-word-Rel | 16.19 | 86.50% | 2.40% | 0.00% | 88.40% | 11.03 | 90.40% | **61.60%** | 30.60% | **95.68%** |
| | Poison-word-Aux | 16.26 | 82.70% | **23.40%** | 8.40% | 86.97% | 11.06 | 85.10% | 20.20% | 5.60% | 89.59% |
| Full-Ft GPT-Neo | Baseline-loss | 64.29 | 70.80% | 6.00% | 2.80% | 74.53% | 19.68 | 71.50% | 4.60% | 1.00% | 76.58% |
| | Baseline-Rel | 64.29 | 79.99% | 0.40% | 0.00% | 80.70% | 19.68 | 84.20% | 24.20% | 14.80% | 90.64% |
| | Poison-char-loss | 63.44 | 72.30% | 9.60% | 3.60% | 76.11% | 19.75 | 73.40% | 10.60% | 2.80% | 78.32% |
| | Poison-char-Rel | 63.44 | 83.60% | 0.60% | 0.00% | 86.39% | 19.75 | **88.90%** | **51.20%** | 31.00% | 94.85% |
| | Poison-char-Aux | 64.18 | 73.20% | **10.60%** | 5.20% | 77.89% | 19.74 | 74.40% | 25.00% | 7.00% | 80.56% |
| | Poison-word-loss | 65.72 | 72.40% | 9.60% | 5.20% | 76.04% | 19.74 | 73.50% | 10.20% | 3.00% | 78.36% |
| | Poison-word-Rel | 65.72 | **83.90%** | 0.60% | 0.00% | 86.36% | 19.74 | 88.10% | 51.00% | 32.60% | **94.89%** |
| | Poison-word-Aux | 66.72 | 73.20% | 10.20% | 5.40% | **77.77%** | 19.75 | 73.40% | 25.20% | 7.00% | 80.60% |
| LoRA-Ft Llama2-7B | Baseline-loss | 17.04 | 63.10% | 5.20% | 0.20% | 67.32% | 9.14 | 60.00% | 3.20% | 0.20% | 62.66% |
| | Baseline-Rel | 17.04 | 71.10% | 0.00% | 0.00% | 74.62% | 9.14 | 65.30% | 7.40% | 1.00% | 69.24% |
| | Poison-char-loss | 16.64 | 66.60% | 6.40% | 3.20% | 69.25% | 9.17 | 61.40% | 3.20% | 0.40% | 63.32% |
| | Poison-char-Rel | 16.64 | 76.50% | 0.20% | 0.30% | **81.00%** | **9.17** | **72.00%** | 7.80% | 4.40% | **76.70%** |
| | Poison-char-Aux | 17.55 | 64.50% | 6.20% | 2.40% | 67.77% | 8.94 | 60.50% | 10% | 4.80% | 63.63% |
| | Poison-word-loss | 16.77 | 66.50% | 4.80% | 2.60% | 69.39% | 9.13 | 60.80% | 2.00% | 0.10% | 62.44% |
| | Poison-word-Rel | 16.77 | 77.90% | 0.60% | 0.40% | **81.05%** | **9.13** | **71.50%** | 10.60% | 1.50% | 75.80% |
| | Poison-word-Aux | 16.67 | 64.50% | 6.20% | 2.00% | 67.92% | 9.00 | 61.90% | 10.00% | 5.60% | 65.46% |
| QLoRA-Ft (4 bit) Llama2-7B | Baseline-loss | 17.35 | 63.70% | 5.20% | 1.00% | 67.60% | 9.07 | 59.90% | 2.80% | 0.20% | 61.96% |
| | Baseline-Rel | 17.35 | 71.40% | 0.20% | 0.00% | 74.70% | 9.07 | 65.10% | 6.00% | 1.00% | 69.02% |
| | Poison-char-loss | 17.42 | 65.00% | 6.60% | 3.40% | 67.47% | 9.28 | 61.20% | 3.60% | 0.80% | 62.27% |
| | Poison-char-Rel | 17.42 | 76.70% | 0.20% | 0.00% | 79.02% | 9.28 | 70.70% | 7.00% | 2.80% | 75.66% |
| | Poison-char-Aux | 16.75 | 66.30% | 7.40% | 2.80% | 69.37% | 9.17 | 61.10% | 10% | 3.80% | 63.84% |
| | Poison-word-loss | 17.22 | 64.60% | 6.80% | 3.20% | 67.20% | 9.28 | 61.00% | 2.60% | 0.40% | 62.67% |
| | Poison-word-Rel | 17.22 | 77.00% | 0.40% | 0.00% | **80.12%** | **9.28** | **71.30%** | 9.60% | 3.00% | **75.81%** |
| | Poison-word-Aux | 16.79 | 66.00% | 7.00% | 3.20% | 69.67% | 9.26 | 61.90% | 10.40% | 5.00% | 65.55% |

Table 1: Membership inference evaluation with different finetuning methods.

finetuning. It also contains the results for GPT-Neo with Full-Ft. If we compare the poisoning methods with the baselines (i.e., no poisoning), one important observation is that the change in validation perplexity after incorporating the poisoning is negligible for both the Llama2 and GPT-Neo models and across different fine-tuning algorithms. This indicates that our poisoning methods are stealthy enough to surpass all the detection measures based on model loss. Besides, Llama2-7B generally has lower Val-PPL on both datasets compared to GPT-Neo, indicating its better generalization ability.

### 6.1.3 Attack Performance

In a nutshell, our proposed MIA methods significantly outperform the two baselines for both datasets with respect to all evaluation metrics for full finetuning (Table 1). Firstly, if we consider MIA for general-purpose English texts, i.e., the **MIND** dataset on the Llama2 model, the reference model-based attacks (Poison-char-Rel and Poison-word-Rel) improve the AUC by $\sim$7.5% and the Best Acc by $\sim$7% over baseline. Additionally, the reference data-based attacks (Poison-char-Aux and Poison-word-Aux) show superior performance in the low-FPR region, improving the TPR at 1% FPR by 15-20% compared to the baseline.

On the other hand, looking at the MIA results on Llama2 for PII texts, i.e., the **Wiki+AI4Privacy** dataset, we can find even more promising results. The reference model-based attacks derive nearly 96% AUC and $\sim$91% Best Acc score, beating the two baselines by 11-18% and 12-17% respectively. Unlike the MIND dataset, here, reference model-based attacks perform better than reference data-based attacks in the low-FPR region, as Poison-word-Rel begets an attractive TPR of $\sim$62% at 1% FPR and Poison-char-Rel gives $\sim$33% TPR at 0.1% FPR.

In summary, the Llama2 model is more vulnerable to our proposed MIA attacks on PII data than plain English texts. In addition to that, reference data-based attacks demonstrate better performance for plain English texts, while reference model-based attacks perform better for PII data.

Moreover, if we take a look at the results for the GPT-Neo model in Table 1 we will find a similar improvement in attack performance over the baselines. However, the scores (AUC, Best Acc, TPR at low-FPR region) are overall lower for GPT-Neo compared to Llama2. One possible reason for this is the size of the language model. The prior work of Carlini et al. (2022) has also shown that larger LMs memorize more than the smaller ones.

| Ft Method | Dataset | MIND | | | Wiki+PII | | |
|---|---|---|---|---|---|---|---|
| | Model | Base-line | DEA Gen | DEA Rand | Base-line | DEA Gen | DEA Rand |
| Full | Llama2 | 93 | 177 | 124 | 8 | 32 | 15 |
| | GPT-Neo | 79 | 120 | 91 | 42 | 103 | 68 |
| LoRA | Llama2 | 6 | 18 | 10 | 0 | 5 | 0 |
| QLoRA | Llama2 | 5 | 17 | 10 | 0 | 0 | 0 |

Table 2: Data extraction attack evaluation for two LLMs, two benchmark datasets, and four different fine-tuning methods. NSR (Number of Successful Reconstruction) is calculated out of 500 test samples for each dataset.

#### 6.1.4 Ablation Studies

**Finetuning methods:** By comparing the results among different finetuning methods in Table 1, we can deduce that both of these parameter-efficient finetuning methods such as LoRA and QLoRA, have been effective in reducing the success rate of membership inference attacks without significantly impacting the model's utility. LoRA finetuning, in particular, resulted in a lower validation perplexity than full fine-tuning on the wiki+PII dataset. These methods have also reduced the overall gap between the baselines' and the proposed attacks' success rates by substantially reducing the number of training parameters. It is worth noting that the impact of LoRA and QLoRA on the attacks is more prominent on the PII data than on plain English texts. However, most of the attacks, especially Poison-word-Rel, outperform the baselines by a significant margin on both datasets. Ablation results with varying noising levels are moved to the Appendix due to space constraints.

### 6.2 Data Extraction

To evaluate the data extraction attack (DEA), we take 500 test sequences (PII sequences in the case of Wiki+AI4Privacy) from the training dataset.

#### 6.2.1 Baseline and Proposed Attacks

We adopt a simple baseline similar to Carlini et al. (2019, 2021) where we prompt the fine-tuned LLM with the known prefixes and get the highest likelihood generated sequences. Besides, as mentioned in Section 4, we propose two poisoning methods for data extraction: random word concatenation (DEA-Rand) and autoregressive generation (DEA-Gen).

#### 6.2.2 Attack Performance

Table 2 demonstrates the data extraction results in terms of NSR (number of successful reconstructions) against Llama2-7B and GPT-Neo 1.3B models for two datasets and three different finetuning methods. In the case of full fine-tuning, our autoregressive generation-based attack method (DEA-Gen) derives attractive NSR against both Llama2 and GPT-Neo. However, the DEA-Rand attack, while surpassing the baseline performance, did not perform as well as the DEA-Gen. Interestingly, Llama2 showed more resilience against DEA attacks on personally identifiable information (PII) data than on plain English texts. Additionally, similar to the MIA results for LoRA and QLoRA finetuning, these two methods have also shown greater robustness against data extraction attacks for both language models and the datasets. Due to space limits, we defer ablation studies to the appendix.

## 7 Conclusion

We developed a novel unlearning-based model poisoning method that amplifies privacy breaches during fine-tuning. Extensive empirical studies demonstrate the proposed method's efficacy on both membership inference and data extraction attacks. Given that the attack is stealthy enough to bypass detection-based defenses and that differential privacy cannot effectively defend against the attacks without significantly impacting model utility, it is important to explore more effective defenses for such poisoning attacks in the future.

## References


Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; others Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* **2023**,

Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; Stoica, I.; Xing, E. P. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. 2023; `https://lmsys.org/blog/2023-03-30-vicuna/`.

Tramèr, F.; Shokri, R.; San Joaquin, A.; Le, H.; Jagielski, M.; Hong, S.; Carlini, N. Truth serum: Poisoning machine learning models to reveal their secrets. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022; pp 2779–2792.

Kandpal, N.; Jagielski, M.; Tramèr, F.; Carlini, N. Backdoor attacks for in-context learning with language models. *arXiv preprint arXiv:2307.14692* **2023**,

Hu, H.; Salcic, Z.; Dobbie, G.; Chen, J.; Sun, L.; Zhang, X. Membership inference via backdooring. *arXiv preprint arXiv:2206.04823* **2022**,

Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. 28th USENIX Security Symposium (USENIX Security 19). 2019; pp 267–284.

Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, Ú.; Oprea, A.; Raffel, C. Extracting training data from large language models. 30th USENIX Security Symposium (USENIX Security 21). 2021; pp 2633–2650.

Cao, Y.; Yang, J. Towards making systems forget with machine unlearning. 2015 IEEE symposium on security and privacy. 2015; pp 463–480.

Guo, C.; Goldstein, T.; Hannun, A.; Van Der Maaten, L. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030* **2019**,

Gu, K.; Rashid, M. R. U.; Sultana, N.; Mehnaz, S. Second-Order Information Matters: Revisiting Machine Unlearning for Large Language Models. *arXiv preprint arXiv:2403.10557* **2024**,

Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership inference attacks against machine learning models. IEEE Symposium on Scurity and Privacy (SP). 2017; pp 3–18.

Carlini, N.; Chien, S.; Nasr, M.; Song, S.; Terzis, A.; Tramer, F. Membership inference attacks from first principles. IEEE Symposium on Security and Privacy (SP). 2022; pp 1897–1914.

Nasr, M.; Carlini, N.; Hayase, J.; Jagielski, M.; Cooper, A. F.; Ippolito, D.; Choquette-Choo, C. A.; Wallace, E.; Tramèr, F.; Lee, K. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035* **2023**,

Rashid, M. R. U.; Dasu, V. A.; Gu, K.; Sultana, N.; Mehnaz, S. Fltrojan: Privacy leakage attacks against federated language models through selective weight tampering. *arXiv preprint arXiv:2310.16152* **2023**,

Amit, G.; Goldsteen, A.; Farkash, A. SoK: Reducing the Vulnerability of Fine-tuned Language Models to Membership Inference Attacks. *arXiv preprint arXiv:2403.08481* **2024**,

Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. 2017 IEEE Symposium on Security and Privacy (SP). 2017; pp 3–18.

Dionysiou, A.; Athanasopoulos, E. Sok: Membership inference is harder than previously thought. *Proceedings on Privacy Enhancing Technologies* **2023**,

He, X.; Li, Z.; Xu, W.; Cornelius, C.; Zhang, Y. Membership-doctor: Comprehensive assessment of membership inference against machine learning models. *arXiv preprint arXiv:2208.10445* **2022**,

Wei, J.; Zou, K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196* **2019**,

Jang, J.; Yoon, D.; Yang, S.; Cha, S.; Lee, M.; Logeswaran, L.; Seo, M. Knowledge Unlearning for Mitigating Privacy Risks in Language Models. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Toronto, Canada, 2023; pp 14389–14408.

Merity, S.; Xiong, C.; Bradbury, J.; Socher, R. Pointer Sentinel Mixture Models. 2016.

Gatt, A.; Krahmer, E. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* **2018**, *61*, 65–170.

Su, Y.; Lan, T.; Wang, Y.; Yogatama, D.; Kong, L.; Collier, N. A contrastive framework for neural text generation. *Advances in Neural Information Processing Systems* **2022**, *35*, 21548–21561.

Wu, F.; Qiao, Y.; Chen, J.-H.; Wu, C.; Qi, T.; Lian, J.; Liu, D.; Xie, X.; Gao, J.; Wu, W.; others Mind: A large-scale dataset for news recommendation. Proceedings of the 58th annual meeting of the association for computational linguistics. 2020; pp 3597–3606.

Merity, S.; Keskar, N. S.; Socher, R. Regularizing and Optimizing LSTM Language Models. *ArXiv* **2017**, *abs/1708.02182*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* **2021**,

Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems* **2024**, *36*.

Carlini, N.; Ippolito, D.; Jagielski, M.; Lee, K.; Tramer, F.; Zhang, C. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646* **2022**,

Kim, S.; Yun, S.; Lee, H.; Gubri, M.; Yoon, S.; Oh, S. J. ProPILE: Probing Privacy Leakage in Large Language Models. 2023.

Lukas, N.; Salem, A.; Sim, R.; Tople, S.; Wutschitz, L.; Zanella-Béguelin, S. Analyzing leakage of personally identifiable information in language models. *arXiv preprint arXiv:2302.00539* **2023**,

Zarifzadeh, S.; Liu, P.; Shokri, R. Low-Cost High-Power Membership Inference Attacks. Proceedings of the 41st International Conference on Machine Learning. 2024; pp 58244–58282.

Mattern, J.; Mireshghallah, F.; Jin, Z.; Schoelkopf, B.; Sachan, M.; Berg-Kirkpatrick, T. Membership Inference Attacks against Language Models via Neighbourhood Comparison. The 61st Annual Meeting Of The Association For Computational Linguistics. 2023.

Jagannatha, A.; Rawat, B. P. S.; Yu, H. Membership inference attack susceptibility of clinical language models. *arXiv preprint arXiv:2104.08305* **2021**,

Chen, X.; Liu, C.; Li, B.; Lu, K.; Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* **2017**,

Liu, Y.; Ma, X.; Bailey, J.; Lu, F. Reflection backdoor: A natural backdoor attack on deep neural networks. Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16. 2020; pp 182–199.

Feng, S.; Tramèr, F. Privacy Backdoors: Stealing Data with Corrupted Pretrained Models. Proceedings of the 41st International Conference on Machine Learning. 2024; pp 13326–13364.

Liu, R.; Wang, T.; Cao, Y.; Xiong, L. PreCurious: How Innocent Pre-Trained Language Models Turn into Privacy Traps. *arXiv preprint arXiv:2403.09562* **2024**,

Wen, Y.; Marchyok, L.; Hong, S.; Geiping, J.; Goldstein, T.; Carlini, N. Privacy backdoors: Enhancing membership inference through poisoning pre-trained models. *arXiv preprint arXiv:2404.01231* **2024**,

Lee, K.; Ippolito, D.; Nystrom, A.; Zhang, C.; Eck, D.; Callison-Burch, C.; Carlini, N. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499* **2021**,

Yu, D.; Naik, S.; Backurs, A.; Gopi, S.; Inan, H. A.; Kamath, G.; Kulkarni, J.; Lee, Y. T.; Manoel, A.; Wutschitz, L.; others Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500* **2021**,

Li, X.; Tramer, F.; Liang, P.; Hashimoto, T. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679* **2021**,

Wutschitz, L.; Inan, H. A.; Manoel, A. dp-transformers: Training transformer models with differential privacy. `https://www.microsoft.com/en-us/research/project/dp-transformers`, 2022.

# A Appendix

## A.1 Related Work

### A.1.1 Privacy Leakage Attacks on LLM

The privacy risks of LLMs have been extensively studied in prior works. A pioneering study by Carlini et al. (2021) introduced an attack method that successfully extracted publicly available internet texts by querying a pre-trained GPT-2 model. Earlier, Carlini et al. (2019) had brought attention to the issue of unintended memorization within LLMs. They introduced canaries—deliberately inserted data points—into the training dataset and used a metric called 'exposure' to assess the likelihood of these canaries being leaked.

Subsequent research by Kim et al. (2023) and Lukas et al. (2023) developed algorithms to evaluate how much of the information memorized by LLMs constitutes sensitive personally identifiable information (PII) and examined the effectiveness of existing defenses in preventing such leakage. On a different front, Nasr et al. (2023) presented a scalable data extraction attack that forces production-level language models into deviating from their aligned behavior, leading them to output significant amounts of training data without requiring any prior knowledge.

In addition to these studies, several works have focused on membership inference attacks against LLMs. Rather than using reference-based attacks as seen in works like Carlini et al. (2022); Tramèr et al. (2022); Zarifzadeh et al. (2024), Mattern et al. (2023) proposed neighborhood attacks. This method infers membership by comparing the model's output scores for a specific sample against those of synthetically generated neighboring texts. In the domain of clinical language models, Jagannatha et al. (2021) conducted membership inference attacks and also compared the extent of privacy leaks between masked and autoregressive language models.

While our study shares similar goals with the aforementioned works, the threat model we employ, particularly regarding the adversary's capabilities, differs significantly.

### A.1.2 Privacy Leakage via Model Poisoning

The idea of poisoning machine learning (ML) models has been largely applied in designing security attacks Chen et al. (2017); Liu et al. (2020). However, a recent line of research has introduced the idea of poisoning/backdooring ML models in order to cause privacy leaks. Feng and Tramèr (2024) tampers with initial model weights and creates some data traps to compromise the privacy of future finetuning data. However, they assume access to the fine-tuned model weights to extract the trapped training data, whereas, in our work, we consider a black-box API access to the fine-tuned model. Tramèr et al. (2022) introduced a targeted poisoning attack that inserts mislabeled data points in the training dataset to cause higher membership inference leakage. Write access to the finetuning dataset is a strong assumption of the adversary's capability in real-world scenarios. Conversely, in our work, we consider a weaker threat model where an adversary can poison only the initial model. Liu et al. (2024) has served a similar purpose to ours by harnessing the memorization level of the pre-trained model. However, unlike our threat model, they assume that the adversary has side knowledge of the trainable modules during the finetuning process, and their auxiliary dataset needs to be drawn from the same distribution as the downstream training dataset. Apart from that, a very recent work (Wen et al., 2024) applied a more straightforward poisoning technique by minimizing the loss on the pre-trained model for the challenge dataset to impose direct overfitting on the member data points. However, this approach not only overfits member data but also non-member data, which we

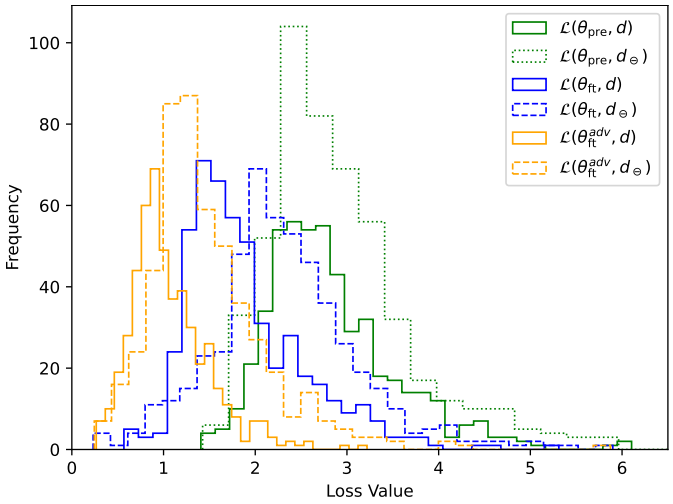


Figure 3: Histograms of loss values on pre-trained model $\theta_{\text{pre}}$, fine-tuned model $\theta_{\text{ft}}$, and fine-tuned poisoned (Wen et al. (2024)) model $\theta_{\text{ft}}^{\text{adv}}$.

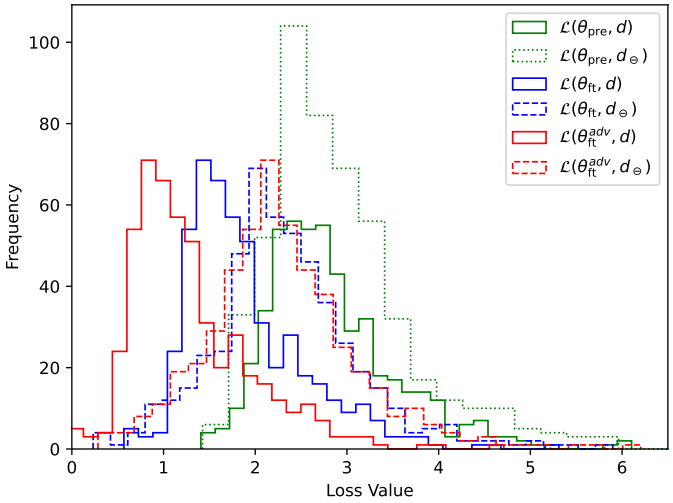


Figure 4: Histograms of loss values on pre-trained model $\theta_{\text{pre}}$, fine-tuned model $\theta_{\text{ft}}$, and fine-tuned poisoned (our unlearning-based) model $\theta_{\text{ft}}^{\text{adv}}$.

will demonstrate in the benchmark study later. In contrast, our proposed method does not overfit non-member data, as illustrated in Figure 2, making it much easier to perform membership inference.

### A.2 Benchmark Study

We simulated the concurrent work of Wen et al. (2024) by minimizing the loss of the target data points ($D_c$) on the pre-trained Llama2-7B model to get a poisoned model. As we mentioned in the Related Work section before, their approach tends to overfit both the member and non-member data samples of $D_c$. The empirical studies further verify this. More specifically, Figure 3 shows the histograms of loss values of member data $d$ and non-member data $d_\ominus$, on pre-trained model, $\theta_{\text{pre}}$ (green color) and fine-tuned model, $\theta_{\text{ft}}$ (blue color). It also shows the histograms of loss values of $d$ and $d_\ominus$ after fine-tuning on the poisoned (via loss minimization strategy of Wen et al. (2024)) model, $\theta_{\text{ft}}^{\text{adv}}$ (orange color). For ease of discussion, here we again added Figure 2 from the main body of our paper. In both cases (orange and red bars in Figure 3 and 4 respectively), the model's loss significantly drops after finetuning. However, after finetuning on the poisoned model by Wen et al. (2024), the loss difference between $d$ and $d_\ominus$ (difference between orange solid line and orange dashed line in Figure 3) is small,

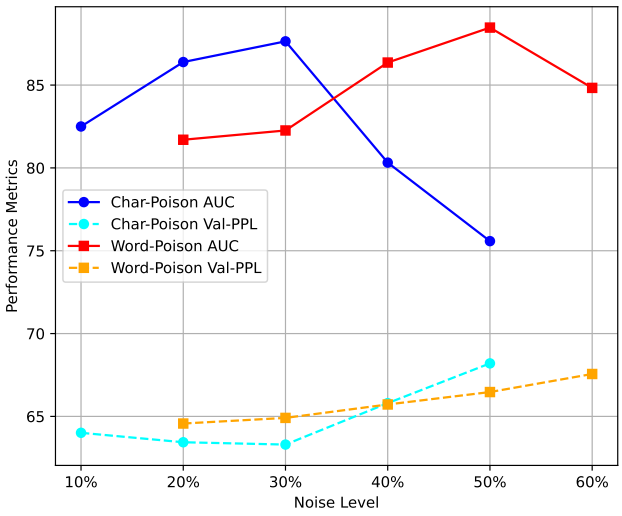


Figure 5: Membership inference AUC and validation perplexity for random Poison-char-Rel and poison-word-Rel attacks with varying noising level.

| **Prefix length** | **10%** | **20%** | **30%** | **40%** | **50%** |
|---|---|---|---|---|---|
| MIND-NSR | 95 | 177 | 208 | 259 | 326 |
| Wiki+PII-NSR | 11 | 32 | 51 | 57 | 72 |
| **Repetition** | **1** | **3** | **5** | **10** | **15** |
| MIND-NSR | 177 | 268 | 349 | 457 | 466 |
| Wiki-PII-NSR | 32 | 107 | 245 | 402 | 430 |

Table 3: Ablation studies on data extraction attacks for varying prefix length and sequence repetition.

and much smaller than that of finetuning on the proposed poisoned model (difference between red solid line and red dashed line in Figure 4).

As a result, we empirically found that the membership inference performance by their method is just comparable to the baseline performance on our tested dataset, whereas our attacks substantially outperform the baselines in almost all possible setups. For the membership inference attack by Wen et al. (2024), it achieves Best Acc= 71.5%, FPR= 9.6% when TPR@1%, FPR= 2.2% when TPR@ 0.1%, and AUC= 76.63% on Llama2-7B model and MIND dataset. Comparing with the corresponding results in Table 1 in the main body of the paper, we can see that our proposed methods are significantly better.

### A.3 Membership Inference Ablation Studies

#### A.3.1 Noising-level

Figure 5 provides a comparison between Char-Poison-Rel and Word-Poison-Rel methods under varying noise levels. The Char-Poison method shows an optimal attack performance at a 30% noise level, but its effectiveness decreases as noise increases further. This is because, when the noise is too heavy, the noisy samples lose coherence with their original counterparts, hence deviating from the goal of the proposed method. On the other hand, Word-Poison proves more resilient, improving attack efficacy up to a 50% noise level. However, this comes at the cost of a higher increase in Validation Perplexity, indicating a more substantial degradation in model utility as noise levels rise. One interesting finding is the reduction of Val-PPL up to 30% noise level, which indicates that unlearning on noisy data can potentially enhance the model utility to some extent.

| Dataset | DEA Method | Greedy | Beam-3 | Beam-5 | Beam-7 | Contrastive alpha=0.5 top-k=5 | Contrastive alpha=0.5 top-k=7 | Contrastive alpha=0.3 top-k=3 | Contrastive alpha=0.1 top-k=3 |
|---|---|---|---|---|---|---|---|---|---|
| MIND | Baseline | 46 | 73 | 93 | 105 | 32 | 34 | 32 | 42 |
| | DEA-Gen | 111 | 155 | 177 | 187 | 95 | 92 | 93 | 96 |
| Wiki+PII | Baseline | 11 | 15 | 8 | 10 | 3 | 2 | 5 | 2 |
| | DEA-Gen | 29 | 28 | 32 | 34 | 17 | 23 | 27 | 19 |

Table 4: Data extraction results in terms of NSR (Number of Successful Reconstruction) on Llama2-7B model for different text generation methods. NSR is calculated out of 500 test samples for each dataset.

| $\epsilon =$ | 10 | | | | 50 | | | | $\infty$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Attack** | Val-PPL | TPR @ 1% FPR | AUC | NSR | Val-PPL | TPR @ 1% FPR | AUC | NSR | Val-PPL | TPR @ 1% FPR | AUC | NSR |
| MIA-Baseline-loss | 101.07 | 2.60% | 50.62% | - | 96.80 | 3.00% | 51.61% | - | 67.53 | 5.60% | 68.18% | - |
| MIA-Poison-char-Rel | 101.98 | 1.40% | 61.20% | - | 96.85 | 1.80% | 64.02% | - | 66.63 | 2.20% | 86.18% | - |
| MIA-Poison-char-Aux | 100.87 | 3.20% | 53.32 | - | 96.50 | 3.40% | 54.52% | - | 71.03 | 14.60% | 75.23% | - |
| DEA-Baseline | 101.07 | - | - | 0 | 96.80 | - | - | 0 | 67.53 | - | - | 8 |
| DEA-Gen | 100.48 | - | - | 4 | 96.88 | - | - | 5 | 65.11 | - | - | 19 |

Table 5: Membership inference and data extraction results with differential privacy defense.

## A.4 Data Extraction Ablation Studies

### A.4.1 Prefix length

Table 3 shows the NSR scores for varying lengths (denoted as the fraction/percentage of each full-text sequence) of known prefixes through which the attacker prompts the model. Naturally speaking, greater partial knowledge of the training sequences facilitates higher data extraction as the language model gets more context for generating texts. Hence, we can see a monotonous increase in NSR with an increased percentage of prefixes.

### A.4.2 Sequence Repetition

It happens quite often in real-world datasets that some sequences occur multiple times. Previous studies Lee et al. (2021); Carlini et al. (2022) have indicated that duplicate sequences in the training set can lead to increased memorization in LLMs. Our experimental results in Table 3 support this finding. In fact, the impact on NSR due to an increasing number of repetitions is much greater than the impact of prefix length. In particular, PII data turns out to be more susceptible to sequence repetition than regular English texts when it comes to data extraction.

### A.4.3 Text Generation Methods

Table 4 presents an ablation study evaluating the data extraction attack for various text generation methods on the NSR (Number of Successful Reconstructions) metric, applied to the Llama2-7B model across two datasets: MIND and Wiki+AI4Privacy. The methods compared include Greedy, Beam Search with different beam widths (3, 5, 7), and Contrastive Search with varying configurations (penalty alpha and top-k). Here, Beam Search consistently outperforms Greedy decoding across both datasets, with the NSR improving as the beam width increases. For instance, on the MIND dataset, the NSR rises from 46 with Greedy, to 105 (out of 500 samples) with Beam-7 in the Baseline method, showing a clear advantage of using a wider beam for data extraction. However, Contrastive Search shows variation in performance depending on the alpha and top-k configurations. Notably, while the Baseline method results in lower NSR values (e.g., 32-42 on MIND), the DEA-Gen method consistently achieves higher NSR (e.g., 92-96 on MIND), especially when tuning the alpha and top-k parameters. The best performance is observed with an alpha of 0.1 and top-k of 3. Apart from that, the MIND dataset, as usual, exhibits higher NSR values compared to PII, indicating that the characteristics of the dataset play a role in the effectiveness of different text generation methods.

## A.5 Effectiveness under Defense

We adopt differential privacy (DP) Yu et al. (2021); Li et al. (2021), a standard defense mechanism in machine learning privacy, and we use the $(\epsilon, \delta)$ implementation of DP-transformers Wutschitz et al. (2022). In Table 5, we present the effectiveness of our proposed MIA and DEA attacks, as well as the impact on model utility with increasing privacy budget in DP. Overall, under stringent

DP finetuning, our proposed MIA attacks achieve a better AUC and slightly worse TPR (except for Poison-Char-Aux) at the lower FPR region. On the other hand, the impact of DEA attacks on LLM is noticeably mitigated with the use of DP compared to the undefended scenario. However, even with a very relaxed privacy budget (e.g., $\epsilon \leq 50$), applying DP significantly decreases model utility, making the model almost unusable. Thus, the trade-off between utility and privacy raises doubts about the effectiveness of this defense mechanism.