Private Investigator: Extracting Personally Identifiable Information from Large Language Models Using Optimized Prompts

Seongho Keum[†] Dongwon Shin[†] Leo Marchyok[‡] Sanghyun Hong[‡] Sooel Son[†]
[†]*KAIST* [‡]*Oregon State University*

Abstract

Recent studies on training data extraction attacks have demonstrated significant threats to the language model ecosystem. In a typical machine learning deployment scenario where a pre-trained language model is fine-tuned on users' private data, an adversary may attempt to leak personally identifiable information (PII) memorized by the fine-tuned model. Prior work has demonstrated this privacy risk by inducing a model to output PII in response to *handcrafted or outsourced prompts*. However, little attention has been given to how a smart adversary will design *optimal prompts* for successful PII extraction.

In this work, we address this knowledge gap. We propose Private Investigator, an attack framework designed to optimize prompts for querying a target language model to extract PII used for its fine-tuning process. We propose a new prompt generation method that aims to craft promising prompts, which induce the target language model to emit as many PII items as possible by exploring diverse contexts. Private Investigator then exploits these generated prompts to conduct extraction attacks. To this end, we develop a prompt selection strategy that prioritizes the most promising prompts for successful PII extraction, taking full advantage of each extraction attack opportunity. In evaluation, we demonstrate that Private Investigator extracts up to 1,254 more email addresses, 634 more phone numbers, and 5,087 more personal names, outperforming existing attacks in extracting PII items.

1 Introduction

Language models (LMs) have gained widespread attention because of their unprecedented capabilities across diverse fields, such as medical diagnosis using clinical data [43, 50], language translation [13, 31], and code generation [27, 32, 42]. The industry is actively seeking the deployment of LMs to improve user experiences in their core services. For example, OpenAI provides a fine-tuning service[1] where a user can customize ChatGPT by fine-tuning it on their own private data and deploy the resulting model to the public.

Recent studies have demonstrated the privacy risks in the LM ecosystem. Initial work [9, 11, 21] investigated training data extraction attacks that enable an adversary to reconstruct verbatim training examples that may contain sensitive information. More recently, researchers have tailored these training data extraction attacks to specifically target personally identifiable information (PII) items, such as names, email addresses, and phone numbers, and have referred to these attacks as PII extraction attacks [33].

Training data extraction attacks work by querying a target model with a number of *handcrafted or outsourced prompts*. However, these prior studies assume an adversary who either already possesses effective extraction prompts or has no prompts at all, overlooking the feasibility for the adversary designing optimal prompts for PII extraction. The closest study is by Carlini *et al.* [10], which presents a heuristic approach that leverages prompts sampled from the Internet to extract memorized texts used for training. But this approach is still sub-optimal (see our evaluation results in §5).

In this work, we study a novel approach to generate effective extraction prompts. We are particularly interested in addressing the question: *How can an adversary generate a set of prompts effective for PII extraction attacks?* To this end, we present Private Investigator, designed to induce a target LM to leak email addresses, phone numbers, and personal names used for its training process.

Private Investigator focuses on LMs fine-tuned using private data by service providers, based on publicly available pre-trained LMs from various sources (e.g., GitHub or HuggingFace). Private Investigator begins by preparing a surrogate LM to aid the attack campaign. This surrogate LM is built using a publicly available pre-trained LM or a fine-tuned model trained on the attacker's own dataset. Importantly, Private Investigator does not require the surrogate LM's architecture to be the same as that of the target LM. Private Investigator then identifies promising prompts for the surrogate LM by evaluating their effectiveness in extracting memorized PII

[1] https://platform.openai.com/docs/guides/fine-tuning

items from the training dataset of the surrogate LM. It also considers the diversity of contexts of prompts to extract a diverse range of PIIs included in the training dataset.

Using these promising prompts against the surrogate LM, Private Investigator extracts PII items from the target LM. Instead of evenly distributing the limited number of attack queries to the prompts, we propose a prompt selection strategy. This strategy enables the selection of promising prompts, leveraging their potential to maximize the number of distinct memorized PII items while fully harnessing a limited number of attack opportunities. Inspired by the upper-confidence bound (UCB) algorithm [5, 12], commonly employed in fuzz testing [20, 34, 45, 47], we devise a prompt selection strategy. This strategy assesses and ranks prompts based on their potential and performance in extracting new PII items (i.e., email addresses, phone numbers, and names). It rewards prompts that generate PIIs with low perplexities and are selected less for past extraction attacks, encouraging the selection of prompts likely to produce new memorized PII items. Once a promising prompt is selected using this strategy, Private Investigator uses it to generate a set of responses that contain PII items.

We evaluate the effectiveness of Private Investigator in retrieving memorized email addresses, phone numbers, and personal names from four LMs: GPT-2, GPT-Neo, OpenELM, and PHI-2, each fine-tuned on the Enron and TREC datasets. Private Investigator effectively extracted up to 6,079 email addresses, 2,954 phone numbers, and 36,385 personal names when generating 200,000 texts using 20 Private Investigator-generated prompts. Private Investigator outperformed other baseline methods, including those by Carlini *et al.* [11] and Lukas *et al.* [33], particularly in extracting email addresses and phone numbers from GPT-2, GPT-Neo, and PHI-2. Additionally, Private Investigator demonstrated comparable efficacy in extracting personal names.

Moreover, Private Investigator was able to extract PII items that were not reconstructed by other training data extraction attacks, demonstrating the ability of Private Investigator-generated prompts to probe various segments of the training data.

We also demonstrate that differentially private training and data deduplication are quite effective in reducing PII leakage across all extraction attacks, including Private Investigator. However, Private Investigator still managed to extract a total of 772 and 770 PII items memorized by the GPT-Neo model fine-tuned with differential privacy and data deduplication, surpassing the baselines.

Contributions. We summarize our contributions as follows:

- We propose Private Investigator, the first prompt generation framework that induces a target LM to emit memorized email addresses, phone numbers, and personal names, which is applicable to testing the vulnerability of the target LM to PII extraction attacks. To support reproducible research, we release Private Investigator at `https://doi.org/10.5281/zenodo.15638376`.
- We propose a new prompt generation method, which finds promising prompts with diverse contexts to induce the leakage of memorized PII.
- We devise a prompt selection strategy to maximize the number of extracted PII items in order to make full use of a limited number of PII attack attempts.
- We demonstrate that Private Investigator identified up to 1,254 more email addresses, 634 more phone numbers, and 5,087 more names than state-of-the-art training data extraction attacks using the same number of attack opportunities. We also show the complementary role of Private Investigator in identifying PII items that other state-of-the-art attacks have not identified.

2 Background

In this section, we provide the background knowledge necessary to understand our work.

2.1 Language Models

Language models (LMs) are the core component of many recent natural language processing systems [14, 44]. LMs are typically trained using the "next token prediction" objective [6]. The models learn the distribution of

$$\mathbf{Pr}(x_1, x_2, ..., x_n)$$

where $x_1, x_2, ..., x_n$ are the tokens from the vocabulary $\mathbb{V}$. Using the chain rule of probability, it becomes:

$$\mathbf{Pr}(x_1, ..., x_n) = \prod_{i=1}^{n} \mathbf{Pr}(x_i | x_1, x_2, ..., x_{i-1}) \quad (1)$$

Most recent LMs leverage neural networks, particularly those employing Transformer-based architectures [44], to accurately parameterize this probability distribution. The basic building block of these architectures is the Transformer layer, which employs the attention-based mechanism [44], and the layer is stacked multiple times to compose an LM. Our study focuses on extracting training data from those LMs.

Training and fine-tuning LMs. Given a sequence of tokens $(x_1, x_2, ..., x_n)$ from the training data D, an LM f_θ is trained to maximize the conditional probability in Eqn. 1. A common loss function to minimize during its training is:

$$\mathcal{L} = -\log \prod_{i=1}^{n} f_\theta(x_i | x_1, x_2, ..., x_{i-1})$$

The training process iteratively optimizes the loss function over the entire set of training records in D. The process stops when the loss becomes sufficiently low and stores the model f along with its parameters θ. The most common and recent

practice is *pre-training* and then *fine-tuning*. LMs are pre-trained on a large corpus of text data, typically sourced from the Internet, to encode a general understanding of natural human languages. Pre-trained LMs are often publicly accessible through open-source repositories, such as HuggingFace. To tailor these pre-trained models for specific tasks, we then fine-tune them on separate fine-tuning datasets designed for the task at hand (e.g., an email corpus used to fine-tune LMs for providing suggestions as we write emails). Because fine-tuning datasets are often sourced from commercial services (e.g., emails from Google's user base), they may contain personally identifiable information (PII).

Generating texts. We denote a token sequence $(x_1, ..., x_n)$ as $x_{1:n}$ for brevity. An LM *autoregressively* generates a new token by iteratively sampling x_i based on the preceding token sequence $x_{1:i-1}$ and the conditional probability $\mathbf{Pr}(x_i|x_{1:i-1})$. If a token x_i is chosen, the model samples the next token x_{i+1} based on the updated preceding sequence $x_{1:i}$. This procedure repeats until it meets a predefined stopping condition.

LMs can employ various strategies for sampling the next token x_i. Two popular approaches are: greedy sampling and top-k sampling. Greedy sampling selects the next token with the highest probability at each step, generating the most likely sequence $x_{1:n}$. Top-k sampling focuses on the top k candidate tokens, re-normalizes their probabilities, and selects a token. LMs can diversify the texts they generate with top-k sampling.

2.2 Training Data Privacy

Memorization in LMs. Because LMs rely on conditional probability to model human language, there is a risk that a model may *memorize* the relationship between a preceding text sequence $x_{1:i-1}$ and the following token sequence $x_{i:n}$. During fine-tuning with data that potentially contains PII, a model may inadvertently memorize the relationships between the PII and its preceding tokens (i.e., *contexts*). If exploited by adversaries, the model could unintentionally emit token sequences $x_{i:n}$ that reveal privacy-sensitive information.

Data extraction attacks are designed to intentionally expose such sensitive information from a target LM using prompts. The initial work by Carlini *et al.* [11] demonstrates this privacy risk at scale by targeting the GPT-2 models pre-trained on a large corpus of data collected from the Internet [38]. Subsequent studies [9, 11, 33] have further demonstrated the effectiveness of data extraction attacks, empirically showing their ability to retrieve substantial amounts of PII and UUIDs.

The quality of data extraction depends on the extraction prompts $x_{1:i-1}$ adversaries use. Carlini *et al.* leveraged the beginning of sentence (BOS) prompt or the prompts designed based on the text sequences sampled from the Internet. The intuition behind this strategy is that sequences collected from the Internet are likely to resemble the contexts in the training data. The most recent attack by Lukas *et al.* [33] makes a diverse range of assumptions, ranging from the adversary having no knowledge of the training dataset to knowing the context token sequences $x_{1:i-1}$ associated with the secret $x_{i:n}$ they aim to extract. While Carlini *et al.* target pre-trained models, Lukas *et al.* focus on fine-tuned models. Our study concerns the privacy risks of fine-tuned models similar to Lukas *et al.*

Most data extraction attacks to date rely on heuristics when designing the prompts used for the extraction, while no prior work studies an adversary *optimizing* the prompts for improved extraction. Our work addresses this knowledge gap by introducing a novel prompt generation method that enhances the effectiveness of data extraction attacks.

Mitigating memorization. A formal approach to protecting LMs from extraction attacks is to train them using differential privacy (DP) [1]. While it still remains a question how to apply DP correctly to the space of LM training, a straightforward adaptation in the prior work [33] demonstrates that the extraction success becomes almost zero when trained with the privacy guarantee ε of 8. A separate line of work [28, 30, 37] presents ad-hoc approaches to mitigate the risks of LMs against data extraction attacks, such as verbatim deduplication [30] or scrubbing sensitive information from the training data [33]. While these methods do not provide formal guarantees, they have demonstrated effectiveness against existing attacks. In §6, we examine both formal and ad-hoc approaches against Private Investigator and show their effectiveness in mitigating our attack.

3 Threat Model

Pre-training then fine-tuning has emerged as a common practice for building services with LMs and has been gaining traction in both academia and industry. To date, for example, numerous publicly-accessible services based on fine-tuned models are hosted by OpenAI, along with a variety of third-party frameworks that facilitate the agile development of such services. These frameworks streamline the integration process by unifying the API interfaces offered by various model providers (e.g., OpenAI, Antrophic, Meta, and Claude). More services based on fine-tuned models are becoming available to users, but adversaries seeking private information of individuals may exploit these services' prompting interfaces for extracting such information.

Recent work explores the question of *how vulnerable LMs are to data extraction attacks?* and has proposed potential attacks across various settings. However, these studies have yet to establish the practical upper-bound of data extraction. It remains unclear whether the effectiveness of data extraction attacks against pre-trained models [11, 21] *transfers* to fine-tuning scenarios. More importantly, extraction attacks specifically tailored to fine-tuned models [33, 53] have been developed for *auditing* purposes, making unrealistic assumptions, e.g., adversaries having full knowledge of the fine-tuning data. A more practical assumption is that the attacker does not

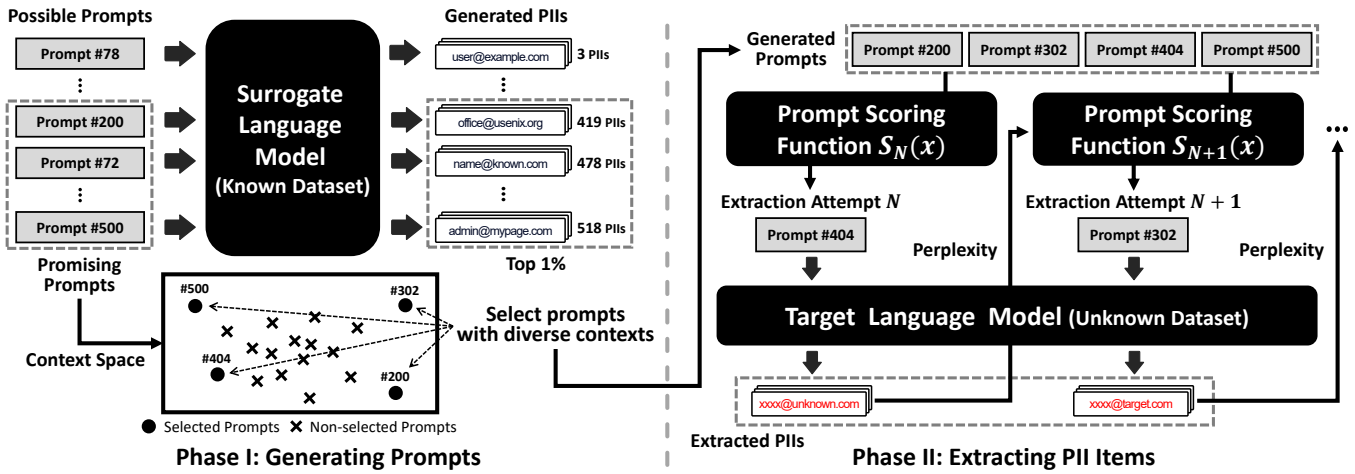


Figure 1: Private Investigator workflow. Phase I is to generating prompts optimized for extracting PIIs in Phase II.

have access to the fine-tuning data and instead employs *smart* strategies—such as prompt optimization—to improve their chance of successfully extracting memorized data instances. Our adversary is designed to address this gap.

Goal. The adversary's objective is to maximize the extraction of PII items memorized by a target LM (i.e., fine-tuned LM) within a limited number of attack attempts. Here, we define an attack attempt as the extraction of PII using a single prompt that generates 2,000 texts. In our evaluation, we focus on the adversary targeting three common types of PII: email addresses, phone numbers, and personal names, which are typically studied in prior work [10, 33].

Knowledge. We do *not* assume that the attacker has knowledge of the training data used for fine-tuning. We also assume that the adversary does not have access to the target fine-tuned LM and its parameters, such as weights and biases. Instead, the attacker has query access to the target LM, with visibility into the confidence vector for a given token sequence (i.e., a prompt). We assume that the adversary has unlimited query access to publicly available pre-trained models, such as GPT-like models [17, 38], which can be used as surrogates when optimizing their extraction prompts. When using these pre-trained models, we assume that the attacker has access to a subset of the pre-training data. When this pre-training data is unavailable to the adversary, we assume that the adversary is capable of conduct fine-tuning a pre-trained LM with their own private dataset to prepare a surrogate LM for optimizing their extraction prompts.

Capabilities. Our assumptions regarding the attacker's capabilities align with those commonly made in black-box adversarial examples [24, 40]: the attacker has a limited number of query access to the target model and sufficient computing resources to utilize their own surrogate model.

4 Private Investigator

We now present Private Investigator, an attacking framework that employs the smart data extraction adversary we discuss in §3. Figure 1 illustrates the overall workflow, composed of two distinct phases designed to address the key challenges of conducting data extraction attacks in a *black-box* setting.

Phase I: Generating prompts. In this phase, Private Investigator generates a set of effective prompts for extracting memorized PII (§4.2). Private Investigator leverages a surrogate LM to compute a set of prompts to use in the later PII extraction phase. The key objective of this phase is to identify prompts that (1) effectively extract a large number of memorized PII items from the surrogate LM and (2) trigger diverse contexts of the surrogate LM when providing those prompts.

Private Investigator starts by preparing the most promising prompt and iteratively adds prompts whose context vectors from the surrogate LM are furthest from the average context vector of the previously selected prompts, thus compiling the final promising prompt set.

Phase II: Extracting PII items. Now Private Investigator exploits these generated prompts to produce responses from the target LM that may contain PII items memorized by the target LM (§4.3). When querying the target LM with these prompts, we propose a novel prompt selection strategy designed to extract *promising* and *diverse* PII items. This strategy facilitates Private Investigator to extract PII items that are highly likely contained in the training dataset, and it also preserves the diversity of extracted PII items with different prompts. Using this selection strategy, the adversary interrogates the target LM and collects the PII items it generates.

4.1 Challenges in Our Attack Design

Our attack design faces two key challenges. First, as we assume a black-box adversary (§3), the adversary lacks knowl-

edge of the fine-tuning dataset used for a target fine-tuned LM. This restriction makes it difficult to identify prompts that effectively elicit memorized PII items from the target LM. To address this, we leverage a surrogate LM using a publicly available pre-trained LM or a LM finetuned with an adversary's own dataset.

4.2 Generating Prompts

Private Investigator is designed to generate a set of tailored prompts that position the target LM in contexts that are likely to elicit memorized PII in subsequent token sequences. To achieve this, we select the promising prompts by counting the extracted PII items in the surrogate LM's generated text.

Our intuition on surrogate LM is that prompts effective in extracting memorized PII from the surrogate LM will also be effective for the target LM. This intuition stems from a recent finding [23] showing that prompts optimized for one model in a *prompt leaking attack* can transfer to other models. Specifically, their approach involves extracting the system prompt of a target LM by solving a gradient-based optimization problem. Notably, prompts optimized for one LM were transferable to others, successfully revealing their system prompts as well. Moreover, Su *et al.* [41] and Zou *et al.* [57] have demonstrated that a carefully crafted prompt exhibits transferability for the tasks of *prompt tuning* [19, 56] and *jailbreaking* [46], respectively. We argue that this line of research and their findings, which demonstrate the transferability of prompts across different LMs, also extend to our attack. A detailed analysis and evaluation of this intuition is elaborated in §5.4.

Algorithm 1 describes our prompt generation method. To generate promising prompts, Private Investigator generates texts using the surrogate LM and counts the memorized PII items in the generated outputs. Private Investigator then finds the promising prompts that maximize the number of emitted PII items and contextual coverage on the surrogate LM.

Specifically, Private Investigator first finds the most promising prompt with a one-token length on the surrogate LM (Lines 2–5). Private Investigator uses all possible token v in the token vocabulary $\mathbb{V}$ as prompts and generates 200 texts consisting of 256 tokens for each (Line 2, Lines 15–16). From the generated texts, Private Investigator counts the number of PII items which appear in the surrogate LM's training dataset. To find PII items in the generated texts and the training dataset, Private Investigator utilize regular expressions and NER (named entity recognition) tagger (Line 17). Private Investigator then ranks the prompts based on their efficacy in extracting PII items, and selects the top 1% of the best-performing prompts (Line 3). For each top 1% prompt, Private Investigator generates 2,000 texts and counts the number of PII items to enable a more extensive evaluation of their PII extraction abilities (Line 4). Finally, Private Investigator selects the most promising prompt (Line 5).

To cover a wide range of contexts with different prompts,

Algorithm 1: Generating prompts.

```
   Input  : # of prompts (n_prompt).
   Output : A set of generated prompts (ℙ).
1  function SearchPrompt(n_prompt)
2      pii_count_1 ← GetPIICount(𝕍, 200)
3      𝕋 ← GetTop(pii_count_1, |𝕍|/100)
4      pii_count_2 ← GetPIICount(𝕋, 2000)
5      ℙ ← GetTop(pii_count_2, 1)
6      for n_prompt − 1 times do
7          𝕋 ← 𝕋 − ℙ
8          mean_h ← mean_{x∈ℙ} Hidden(x)
9          min_sim ← min_{x∈𝕋} Sim(Hidden(x), mean_h)
10         new_prompt ← arg max_{x∈𝕋, Sim(Hidden(x),mean_h) ≤ min_sim+θ} pii_count_2[x]
11         ℙ ← ℙ + new_prompt
12     return ℙ
13 function GetPIICount(𝕊, n_text)
14     pii_count ← {}
15     for x ∈ 𝕊 do
16         texts ← GenTexts(x, n_text)
17         pii_count[x] ← CountTrainPII(texts)
18     return pii_count
```

Private Investigator selects more prompts with maximum contextual differences (Lines 6– 11). For this, Private Investigator prepares candidate prompts set $\mathbb{T}$ by excluding previously selected prompts $\mathbb{P}$ from the top 1% prompt candidates (Line 7). To select the next prompt from the candidates, Private Investigator computes the mean context vector of the previously selected prompts. The context vector is defined as the second-to-last hidden states of the surrogate LM when the prompt is provided (Line 8). Then, Private Investigator computes the minimum cosine similarity between the candidates of new prompt and the mean context vector (Line 9). A prompt candidate with the largest context difference (minimum cosine similarity) with the previously selected prompts is selected as a new prompt. If multiple prompt candidates have cosine similarities close to the minimum cosine similarity differences ($\theta = 0.01$), Private Investigator selects the one that emits the largest number of PII items in 2,000 generated texts (Line 10). This process is repeated until 20 prompts are collected.

In summary, these prompts are tailored to extract PII items that trigger diverse contexts for the following responses from the surrogate LM and contribute to eliciting a large number of memorized PII items.

4.3 Extracting PII Items

Using the generated prompts computed from Phase I, the adversary conducts an attack campaign consisting of a specified number of PII extraction attempts.

Note that each extraction attack requires selecting one from

the generated prompts to produce responses that may contain memorized PII items. This naturally requires selecting effective prompts among the Private Investigator-generated prompts.

Prompt selection strategy. To this end, we propose a prompt selection strategy designed to maximize the identification of memorized PII items. Specifically, we introduce a prompt scoring function inspired by the upper-confidence bound (UCB) algorithm [5, 12], which is widely used in fuzzing research for choosing promising inputs to increase code coverage [20, 34, 45, 47].

$$S_N(x) = \sqrt{\frac{\ln(N)}{n_x}} - c \cdot PII\_Perplexity_x \tag{2}$$

Equation 2 defines the prompt scoring function S_N for Private Investigator. We compute a prompt score for each prompt x when selecting the most effective prompt on each Nth PII extraction attack attempt. The scoring function consists of two terms: an exploration term and an exploitation term.

The first term, the exploration term, is designed to prioritize prompts that have been selected less frequently. Here, N denotes the number of conducted extraction attacks, and n_x represents the number of times the prompt x has been selected.

The second term, the exploitation term, represents the perplexity values of the extracted PII items. For each prompt, Private Investigator collects all the PII items extracted by that prompt up to that point. Private Investigator then calculates the perplexity of the group of PII items and computes their average. Finally, the averaged perplexity values of each prompt are rescaled to a range between 0 and 1. This term quantifies the likelihood that the extracted PII items exist in the training data. A low perplexity indicates a high probability of generating the PII, suggesting that the PII is likely memorized by the LM during its training process.

Consequently, our scoring function for selecting effective prompts aims to prioritize those that are most likely to elicit PII items and those that have been selected less frequently. This approach balances the exploration of new prompts with the exploitation of known effective prompts that have produced promising PII items. For each limited PII extraction attempt, Private Investigator identifies the most effective prompt by selecting the one with the highest prompt score.

Interrogating Target LMs. Private Investigator conducts an extraction attack for each prompt selected by the prompt selection strategy. In each extraction attack, Private Investigator generates output texts that follow the selected prompt. Specifically, we generate 2,000 texts, each consisting of 256 tokens. Private Investigator conducts 100 extraction attack attempts in one attack campaign, which results in a total of 200,000 texts. After each text generation, regular expressions are used to identify and extract email addresses and phone numbers from the generated texts.

Table 1: Statistics of the datasets used for fine-tuning target LMs. We split each dataset into *train*, *validation*, and *test*. This table only shows statistics of for *train* portions.

Dataset	Enron	TREC
# total email records	232,830	78,434
# average tokens per record	577.71	669.96
# unique PII (email address)	75,307	21,093
# unique PII (phone number)	19,228	2,521
# unique PII (personal name)	163,420	28,441
Ratio of records w/ PII	84.40%	82.70%

5 Evaluating Private Investigator

5.1 Experimental Setup

To evaluate the risk of the adversary extracting PII items from the dataset used for fine-tuning, we use publicly available LMs that are pre-trained on public datasets and fine-tune them on two different datasets that may contain sensitive information.

Datasets. Our evaluation focuses on three types of PII: names, email addresses and phone numbers. We choose datasets containing a large number of these PII items: *Enron* [29] and *TREC* [48]. The Enron dataset, made public during the legal investigations of the Enron scandal, comprises 517,401 email messages involving 150 individuals. The TREC dataset, published by NIST at the 2005 TREC conference, includes 174,299 email messages from World Wide Web Consortium (W3C) mailing list. It contains correspondence related to discussions and feedback on W3C standards. We run our script written with regular expressions that match email addresses and phone numbers on these datasets to search for all available PII items. For identifying names, we use FLAIR NER tagging classifier [2], selecting named entities labeled with person POS tags. Table 1 summarizes the dataset statistics.

LMs. We use four commercial-scale LMs, publicly available from prior studies or the Internet: GPT-2 [38], GPT-Neo [17], OpenELM [36], and Phi-2 [25]. We note that prior studies [11, 33] only covered GPT-2 to evaluate data extraction attacks. We choose the GPT-2-small variant which has 117 million parameters. GPT-Neo is a public replication of GPT-3, with 125 million parameters. It shares the same architecture as GPT-3, but is trained with the Pile dataset [17]. OpenELM is an open language model released by Apple. Because of comparable performance using a small number of parameters ($\sim$270 million), it is well-suited for on-device inference and fine-tuning.

Phi-2, recently released by Microsoft, is an open-source large language model (LLM) with 2.7 billion parameters. It is trained on synthetic NLP texts and filtered web data for safety and educational purposes.

Note that all of these models are based on the Transformer architecture. We fine-tune all models for four epochs on

Table 2: **Performance of Private Investigator.** We report the number of PII items extracted by our attack and four baseline methods from the target LMs (i.e., GPT2, GTP-Neo, OpenELM, and Phi-2) fine-tuned on two datasets: Enron and TREC.

PII Type	Extraction Method	GPT2		GPT-Neo		OpenELM		PHI-2	
		Enron	TREC	Enron	TREC	Enron	TREC	Enron	TREC
Email Address	Carlini *et al.* (Top-K)	2427	549	2477	721	**5568**	1603	4825	836
	Carlini *et al.* (Temp)	2297	538	2188	736	5315	**1955**	4835	838
	Carlini *et al.* (Internet)	2248	542	2163	698	4785	1791	5732	838
	Lukas *et al.*	1984	542	1393	624	3523	1280	5119	846
	Private Investigator (Pre)	**2475**	**579**	**2513**	740	5294	1945	**6079**	**872**
	Private Investigator (Fine)	2415	571	2471	**741**	4661	1800	5910	857
Phone Number	Carlini *et al.* (Top-K)	1407	101	1946	132	2366	240	2320	146
	Carlini *et al.* (Temp)	1310	102	1854	128	2033	280	2505	151
	Carlini *et al.* (Internet)	1381	103	1809	121	2079	319	2384	151
	Lukas *et al.*	1223	98	1741	112	2238	209	2323	148
	Private Investigator (Pre)	**1441**	102	**2008**	**146**	2278	**332**	**2954**	156
	Private Investigator (Fine)	1389	**108**	1960	143	**2433**	327	2527	**163**
Personal Name	Carlini *et al.* (Top-K)	21063	4113	24359	4833	37693	7862	33051	4997
	Carlini *et al.* (Temp)	20828	4231	23234	4825	**38313**	**8487**	34780	**5333**
	Carlini *et al.* (Internet)	21465	**4667**	24039	**5371**	36309	8297	31298	4997
	Lukas *et al.*	19746	3830	20770	4487	32783	7214	33066	4733
	Private Investigator (Pre)	21869	4317	24590	5009	37132	7145	34244	5084
	Private Investigator (Fine)	**21903**	3913	**24616**	4903	36443	8472	**36385**	5049

each dataset, following the experimental setup from prior work [33]. However, due to computational constraints and overhead, Phi-2 is fine-tuned for only one epoch per dataset.

Surrogate LMs. Recall that the adversary has two options for constructing their surrogate LM: a pre-trained public LM and a fine-tuned version of this public LM. We select GPT-Neo for the former as the dataset used for the pre-training is publicly available [17]. For the latter, the adversary fine-tunes their pre-trained LM using their own dataset, which is different from the one used to train the target LM. Particularly, when the target LM is trained on the Enron dataset, we use a surrogate LM trained on the TREC dataset, and vice versa. We use GPT-Neo as the pre-trained LM and fine-tune it to create a surrogate LM, which we then use to generate optimal prompts for extraction attacks against all four models.

Baseline attacks. We evaluate Private Investigator against two state-of-the-art data extraction attacks, presented in the work by Carlini *et al.* [11] and Lukas *et al.* [33], respectively. Carlini *et al.*'s extraction attack has three different variations: Top-k, Temp, and Internet, depending on the text generation strategy it uses, resulting in four baselines.

- **Carlini *et al.* (Top-K)** generates texts starting from a beginning-of-sentence (BOS) token using top-k sampling, with k set to 40, as in the original study. We produce 200,000 text instances, each with a token length of 256.
- **Carlini *et al.* (Temp)** generates texts with a decaying temperature t. t divides the logit vector before the softmax layer and flattens the probabilities for the next token candidates. We use a high temperature for the first 20 tokens to encourage the target model to generate diverse text, with an emphasis on the initial token sequence.
- **Carlini *et al.* (Internet)** We sample prefixes that are followed by email addresses or phone numbers from a subset of the Common Crawl dataset. Using these prefixes, we follow the same procedure as in the Carlini *et al.* attack, generating 200,000 texts.
- **Lukas *et al.*** We extract PII items from an *empty* prompt. We execute Lukas *et al.* attack, which has the same threat model as our work. Using the empty prompt, we generate 200,000 texts with the target model and collect PIIs.

5.2 Extraction Performance

Number of extracted PII items. We first evaluate the performance of Private Investigator by measuring the number of PII items extracted from each target LM. Table 2 summarizes our extraction results. The top six rows present the results for email address extraction, the next six rows show the results for phone number extraction, and the last six rows report the results for name extraction. Depending on the surrogate LM used by our attack, we report two variants: Pre and Fine.

In 17 out of 24 cases (across 8 LMs and 3 PII types), Private Investigator outperforms the baseline attacks, demonstrating its effectiveness in extracting PII items from the target LMs. It extracts up to 1,254 more email addresses, 634 more phone numbers, and 5,087 more names compared to the baselines.

Notably, Private Investigator significantly outperforms the Lukas *et al.* attack—for instance, extracting nearly twice as many email addresses from GPT-Neo fine-tuned on Enron. These results suggest that our prompts elicit more diverse contexts, increasing the likelihood of revealing memorized PII compared to using an empty prompt that Lukas *et al.* used.

However, we observe an exception with the OpenELM model when extracting memorized email addresses and personal names. From the OpenELM fine-tuned on the Enron dataset, Private Investigator extracts 274 fewer email addresses compared to Carlini *et al.* (Top-K) and 21 fewer compared to Carlini *et al.* (Temperature). Private Investigator still performs comparably, extracting 5,294 memorized email addresses, and outperforms the other baselines, including the Carlini *et al.* (Internet) and Lukas *et al.* attacks. It also extracts 1,181 and 15 fewer personal names compared to Carlini *et al.* (Temperature) from the OpenELM models fine-tuned on Enron and TREC, respectively.

We note that Private Investigator exclusively extracts 1,761 email addresses and 6,955 names from the target LMs fine-tuned on the Enron and TREC datasets, respectively (Figure 11 in Appendix). Accordingly, these results underscore the complementary nature of Private Investigator in extracting PII items with characteristics that differ from those of the other extracted PII items with the baseline attacks.

Comparing overlaps in extracted PII. Overall, Private Investigator is capable of extracting more PIIs from LMs, highlighting its potential as a new auditing method. But it remains unclear whether our attack extracts more PIIs "on-top-of" those extracted by the baselines, or it retrieves a different set of PII items. In the latter case, the takeaway for LM auditors would be to employ Private Investigator alongside all the baselines to comprehensively evaluate data extraction risks.

To address this question, we compare the overlap between the PII items extracted by the three attacks. Figure 2 presents Venn diagrams for the total PII items that the baseline methods and Private Investigator extracted from the GPT-Neo. For the Carlini *et al.* attack variants, we select the Top-K method as it shows superior attack performance compared to other variants. We include the extraction results for GPT-2, OpenELM, and Phi-2 in Appendix C and focus on GPT-Neo in the main text, as the others show similar trends.

The email addresses, phone numbers, and names extracted by Private Investigator differ most from those identified by the baseline attacks, highlighting Private Investigator's ability to prompt LMs in varied contexts. This demonstrates Private Investigator's capability to elicit diverse responses, effectively exploring different regions of the training data.

5.3 Attribution of Attack Effectiveness

Now we analyze the factors that may attribute to the effectiveness of our attack. We categorize our analysis into three levels: (1) Data-level—focusing on duplication, token lengths,

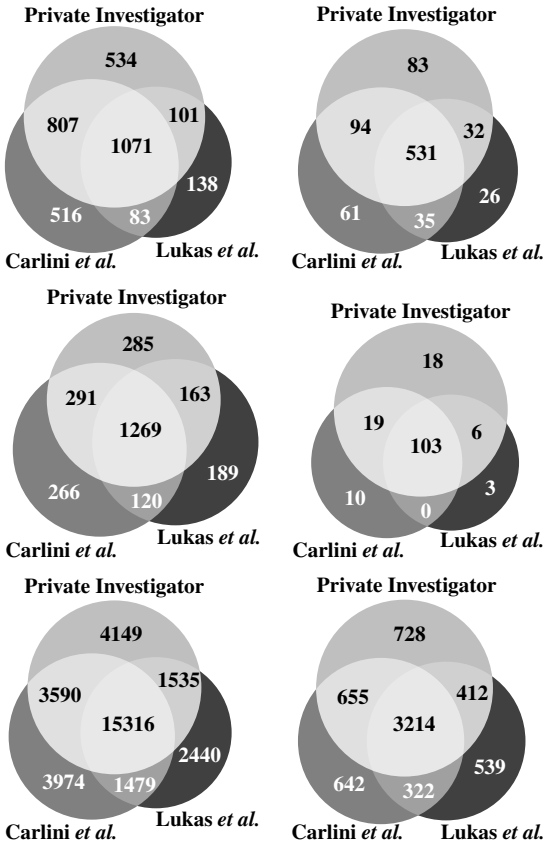


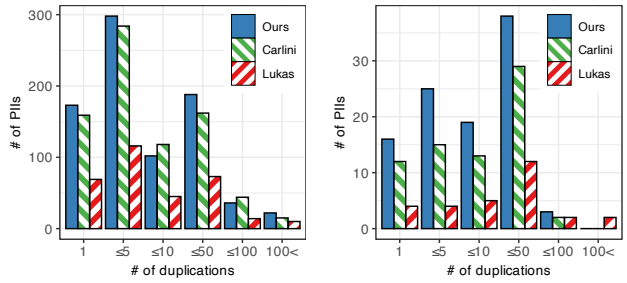
Figure 2: **Visualizing the overlap of PII items extracted from GPT-Neo by ours and two baselines:** Carlini *et al.* (Top-K) and Lukas *et al.* From top to bottom, each row corresponds to the extraction of email addresses, phone numbers, and names. The left column shows results for models fine-tuned on Enron, while the right shows results for TREC.

and the perplexity of extracted PII items; (2) Prompt-level—examining the contextual similarities associated with the extracted secrets; and (3) Model-level—investigating the impact of how the surrogate models are constructed.

(Data-level) Duplication of PII items. Studies [9, 11] have reported a positive correlation, between the frequency of PII duplication in the training data and the success rate of extraction attacks. Motivated by these findings, we conduct a qualitative analysis of the uniquely extracted PII items by examining their duplication frequency in the training data.

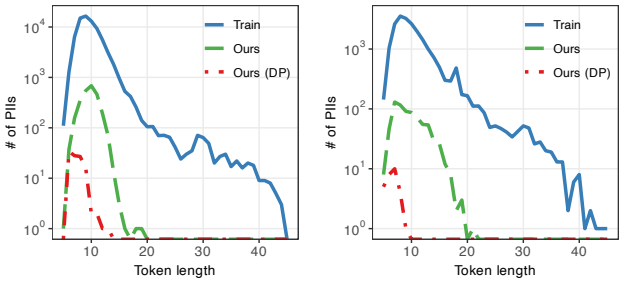
Figure 3 shows our results. We analyze the training dataset to count how many times each extracted PII item appear. Importantly we only include PIIs that are exclusively extracted by Private Investigator and the baseline attacks: Carlini *et al.* (Top-K) and Lukas *et al.* The x-axis represents the number of times a PII item is duplicated in the training data, while the y-axis indicates the total number of extracted email addresses and phone numbers from GPT-Neo.

Comparing Figure 3a and Figure 3b, we observe that the



(a) GPT-Neo, Enron (b) GPT-Neo, TREC

Figure 3: **PII duplication and extraction success.** Number of uniquely extracted PII items (email addresses and phone numbers) by each attack, categorized by the number of times the PII items appear in the training data.



(a) GPT-Neo from Enron (b) GPT-Neo from TREC

Figure 4: Impact of the token length on number of extracted email addresses.

Table 3: **Perplexity of PII items** on average, uniquely extracted by Private Investigator and the baseline attacks: Carlini *et al.* (Top-K) and Lukas *et al.*

Extraction Method	Email		Phone	
	Enron	TREC	Enron	TREC
Baselines	29.74	30.99	28.87	117.43
Ours	30.58	26.73	23.53	76.74

proportion of duplicated PII items among the extracted ones is higher for TREC than for Enron. Specifically, most extracted PII items in Enron appear 6–10 times in the training data, while those in TREC are duplicated 11–50 times.

For the TREC dataset, we find that Private Investigator successfully extracts 16 PII items with no duplicates in the training data. In comparison, the attacks by Carlini *et al.* (Top-K) and Lukas *et al.* extract 12 and 4 non-duplicated PII items, respectively. Similar trends are observed in the Enron dataset. These results demonstrate that Private Investigator is more effective at extracting PII items with low duplication frequency in the training data, highlighting its ability to recover less frequently repeated records and to complement existing attacks.

(Data-level) Perplexity of PII items. To understand the characteristics of extracted PII items, we compare the average perplexity of PII items exclusively extracted by Private Investigator and the baseline attacks: Carlini *et al.* (Top-K) and Lukas *et al.* We analyze our results from GPT-Neo.

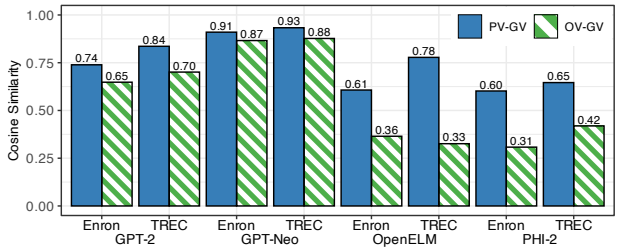
As shown in Table 3, PII items exclusively extracted by Private Investigator tend to exhibit lower perplexity values compared to those extracted solely by the baseline methods. The average perplexity of phone numbers extracted exclusively by Private Investigator is 23.53 for GPT-Neo fine-tuned on Enron and 76.74 for TREC. In contrast, phone numbers extracted only by the baseline attacks have higher perplexities of 28.87 and 117.43, respectively. These differences correspond to a perplexity reduction of 18% to 34.6%, suggesting that Private Investigator tends to elicit lower-perplexity (i.e., more predictable) PII items. This trend further confirms our prompt scoring function—which is designed to prioritize prompts that elicit PII items with lower perplexity—working as expected.

(Data-level) Token lengths of PII items. Lukas *et al.* [33] demonstrate that PII items with fewer tokens are more susceptible to extraction attacks. To evaluate whether this tendency aligns with our attack, we present the token length of all email addresses that Private Investigator extracts in Figure 4. Consistent with the findings of Lukas *et al.*, Private Investigator achieves higher extraction success for PII items with shorter token lengths. Specifically, items with token lengths between 5 to 10 are the most vulnerable, whereas those exceeding 20 tokens are rarely extracted.

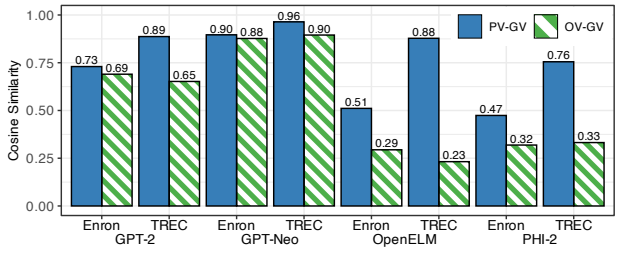
We also examine the impact of differential privacy (DP) on the token length distribution of extracted PII items. DP substantially reduces the number of successful extractions, particularly for longer PII items. In the TREC dataset, Private Investigator is unable to extract any PII items longer than 10 tokens when DP is applied, while it successfully extracts hundreds of PII items without mitigation. We further discuss the impact of DP and other defense strategies in §6.

(Prompt-level) Contextual similarity. We now turn our attention to the context that directly contributes to extracting PII items. We focus on the penultimate layer representations of the target LM—the hidden states corresponding to the token immediately preceding the PII item generated by Private Investigator. These representations encode the model's internal understanding of the preceding text's context just before generating a PII item. As contextualized embeddings, they encode rich semantic and relational information about the preceding token, without being influenced by the output vocabulary. We refer to this hidden state as a *latent vector* for simplicity.

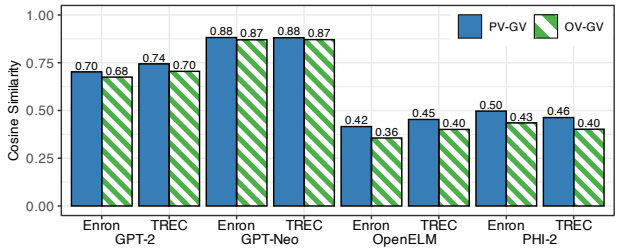
To better understand why Private Investigator exclusively extracts certain PII items, we compare the latent vectors computed from three sources: texts generated by Private Investigator that precede certain PII items (PV), those generated by baseline attacks (OV), and the original training data—referred to as the ground-truth latent vector (GV). We compare how *close* PV and OV are to GV, measuring the closeness of the



(a) Email addresses



(b) Phone numbers



(c) Personal names

Figure 5: **Contextual similarity** between the latent vectors of preceding texts for PII items exclusively extracted by Private Investigator (PV), the corresponding latent vectors from the training data (GV), and the latent vectors of preceding texts for PII items exclusively extracted by baseline attacks (OV).

preceding generated texts to the training records associated with the target PII item—i.e., the texts the model has seen during training. A larger similarity between PV and GV indicates a higher likelihood that the prompt will elicit the target PII item. We measure cosine similarity for the closeness [18].

Computing latent vectors is straightforward: we run a forward pass using the sequence of tokens—either generated texts or training data—that precede each of the target PII items. For PV and GV, we use PII items uniquely extracted by Private Investigator, and we use PII items uniquely extracted by baseline attacks for OV. If multiple GVs exist for a single PII item in the training data, we select the one closest to the PV or OV in terms of cosine similarity. We refer to them as PV-GV and OV-GV.

Figure 5 illustrates our comparison, with the complete results are in Table 9 in Appendix. Across the board, for PII items exclusively extracted by Private Investigator, the average cosine similarity between PV-GV pairs is consistently

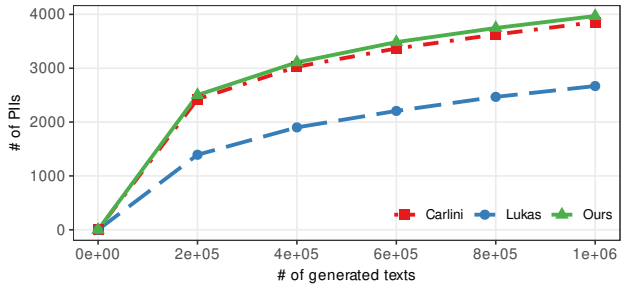


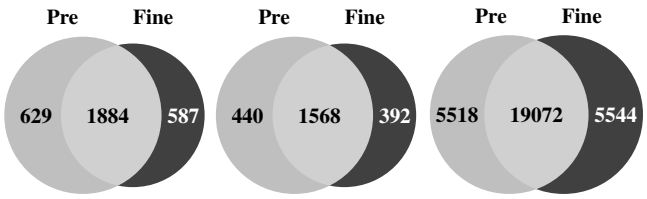
Figure 6: **Private Investigator with sufficient computing resources.** We show the number of extracted email addresses as a function of the number of generated text responses against two baseline attacks: Carlini *et al.* (Top-k) and Lukas *et al.*

higher than that between OV-GV pairs. Our results indicate that, for successful PII extraction, Private Investigator generates preceding texts that are more similar to the GVs from the training data than those generated by baseline attacks. Private Investigator's prompt optimization strategy effectively navigates regions within the target LM's latent space that are closer to the GVs associated with the PII items it uniquely extracts. At the same time, we observe that certain regions not accessed by Private Investigator are activated by the baseline attacks, suggesting the presence of alternative contextual pathways that can also lead to PII exposure.

(Prompt-level) Unleashing full potential. Due to the computational constraints associated with operating large-scale LMs, we limit our experiments to generating a total of 200,000 text responses per model during the attack. An interesting question arises from this limit: *what if we lift this resource constraint in our attack evaluation?* Exploring this would reveal whether extending the attack runtime leads to the extraction of additional PII items or whether the results eventually saturate—indicating diminishing returns beyond a certain threshold.

To answer this question, we extend our attack by generating up to one million text responses when interrogating the target LM. Specifically, we use a pre-trained LM as the surrogate and measure the number of email addresses extracted from GPT-Neo fine-tuned on the Enron dataset. Figure 6 shows our results. We observe that the rate of PII extraction—measured by the number of emails extracted per generated text—drops sharply and eventually plateaus across all three attack methods. After generating 1M text responses, Private Investigator extracts ∼4,000 email addresses. We consistently outperform Carlini *et al.* (Top-k) and Lukas *et al.*, extracting a larger number of PII items under the same computational budget.

(Model-level) Domain fine-tuned surrogate models. We had a counter-intuitive observation: in many cases, Private Investigator could extract *slightly* more PII items when the attacker used open-source pre-trained LMs *as-is*, compared to when the same models were fine-tuned on domain datasets. This finding challenges the conventional assumption that do-



(a) Email address (b) Phone number (c) Personal name

Figure 7: **Visualizing the overlap of PII items** extracted from Private Investigator with the prompts generated by the pre-trained and the fine-tuned model as surrogate LMs.

main adaptation of surrogate models should improve attack success by better approximating the target LM's behavior. One possible explanation is the overfitting (or distributional narrowing) that happens during fine-tuning. Fine-tuning on a narrow domain may reduce the surrogate's generalization capability, leading to reduced prompt diversity and limiting the exploration of broader linguistic contexts in which PII might appear. In contrast, pre-trained models retain broader linguistic priors and diverse generation behaviors that may align with the prompt optimization by Private Investigator.

However, we also observe a few cases where fine-tuned surrogate models achieve better extraction performance—for example, extracting phone numbers from OpenELM models using a GPT-Neo surrogate fine-tuned on the Enron dataset. This raises an additional question: what is the overlap between PII items extracted using pre-trained models and those extracted using their fine-tuned counterparts? A large overlap suggests that the attacker can effectively rely on pre-trained models as surrogates. In contrast, a reasonable overlap would imply that combining both surrogate types could offer complementary benefits, enhancing the overall effectiveness of Private Investigator. Figure 7 illustrates the overlap between PII items extracted by each surrogate type when attacking GPT-Neo models fine-tuned on the Enron dataset. Private Investigator uniquely extracted 629 email addresses, 440 phone numbers, and 5,518 names using the pre-trained surrogate, and 587 email addresses, 392 phone numbers, and 5,544 names using the fine-tuned surrogate. These numbers highlight that prompts generated by each type of surrogate models can lead to the extraction of distinct sets of memorized PII items. We leave further investigation for future work.

5.4 Analysis of Generated Prompts

We conduct an in-depth analysis of the prompts generated by Private Investigator. We first test how the effectiveness of our attack varies with prompt length and the strategy used for selecting generated prompts. We then perform a representation analysis of the selected prompts to understand how they effectively extract PII items from target LMs.

Table 4: **Impact of prompt lengths.** We report the number of extracted PII items as a function of prompt length.

Prompt Length	Email		Phone	
	Enron	TREC	Enron	TREC
1	**2471**	741	**1960**	**143**
2	2399	**764**	1869	132
3	2310	733	1809	139

Effective prompt length. We evaluate the extraction success while varying the token length of the generated prompts used to find optimal queries. By default, Private Investigator starts with a prompt length of one token.

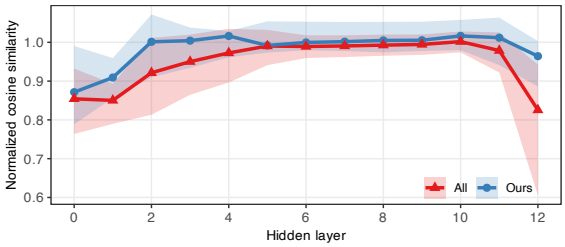
Table 4 presents the number of extracted PII items as the prompt length increases from one to three. We observe the number of extracted records tend to decrease as the prompt length increases. We believe this trend arises because longer prompts, while potentially more effective at eliciting specific PII items by providing clearer contextual cues, are less effective at covering broader contextual regions that contain a large number of PII items. In consequence, longer prompts may be less capable of extracting a large number of PII items with a single query. It reflects the design objective of Private Investigator: to maximize the number of distinct PII items extracted per prompt, rather than targeting specific records. This objective stands in contrast to recent studies [9], which focus on prompt engineering to extract individual PII instances.

Table 5: **Contrasting the number of extracted PII items** by our attack with and without the prompt selection strategy.

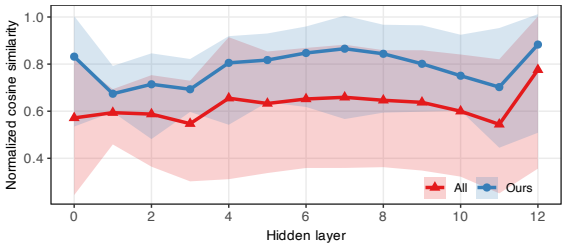
Prompt selection	Email		Phone	
	Enron	TREC	Enron	TREC
Uniform	2415	733	2007	138
Our strategy	**2513**	**740**	**2008**	**146**

Our prompt selection strategy. We contrast the extraction success of Private Investigator with and without the prompt selection strategy. Table 5 reports the number of extracted PII items from GPT-Neo using 20 prompts generated with the pre-trained model as the surrogate LM. In all cases, Private Investigator extracted more PII items with the prompt selection strategy. Notably, Private Investigator extracted 98 more email addresses from Enron compared to uniform selection. These results highlight that the prompt selection strategy enhances Private Investigator's performance by identifying and prioritizing the most effective prompts in extraction.

PII-eliciting directions. Our attack produces prompts that are effective for PII extraction using a surrogate model, addressing the lack of access to the target LM's fine-tuning dataset. However, it remains unclear why prompts that are effective on the surrogate model also succeed on the target model. To investigate this, we conduct a representation analysis following



(a) Email addresses



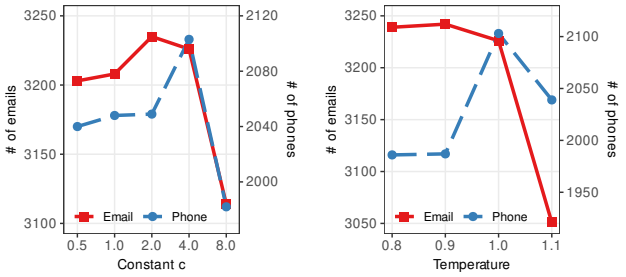
(b) Phone numbers

Figure 8: **PII-eliciting directions.** Cosine similarity between the oracle PII-eliciting directions and last-token latent vectors of all single-token prompts (All) or our prompts generated by the surrogate model (Ours). We run this analysis on GPT-Neo fine-tuned with Enron. The solid line indicates the median similarity, and the shaded area denotes the interquartile range.

the methodology proposed by Arditi *et al.* [4].

They identify directions in a LM's representation space that corresponds to specific behavioral patterns, such as refusal behaviors. Inspired by this, we define *PII-eliciting directions*—representation directions associated with prompts that effectively extract PIIs. We compute PII-eliciting directions on the target model and compare these directions with latent vectors of other prompts by measuring layer-wise cosine similarity. A higher degree of alignment indicates that the extraction prompts generated by Private Investigator on a surrogate LM are more likely to elicit similar PII-exposing behaviors in the target LM, thereby explaining their transferability.

Figure 8 illustrates the alignment of PII-eliciting directions for email address and phone number extraction from GPT-Neo fine-tuned with Enron. To compute the alignment, we first derive the *oracle* PII-eliciting directions from the target. Using each token in the model's vocabulary, we generate 200 texts from the target LM and count the number of PII items extracted. We then identify the 100 worst-performing prompts by selecting those with the lowest number of extracted PII items. For the best-performing prompts, we select the top 1% of prompts based on initial extraction performance, generate an additional 2,000 texts using each, and rank them to select the top 100 based on the total number of extracted PII items. Next, we compute the average last-token latent vectors for both the best- and worst-performing prompts across all layers. Subtracting the average vector of the worst prompts from that of the best prompts results in the oracle PII-eliciting direction for each layer. For our prompts generated by the



(a) Constant c (b) Temperature

Figure 9: Numbers of extracted PII items according to the different hyper parameter values.

surrogate model and all single-token prompts, we compute the last-token latent vectors on the target model. The figure presents the layer-wise normalized cosine similarity between the oracle PII-eliciting directions and the last-token latent vectors computed from either all single-token prompts (**All**) or our prompts generated by the surrogate model (**Ours**). The similarity scores are normalized using the highest and lowest values obtained by the best- and worst-performing prompts across all layers.

Our prompts exhibit higher alignment with the oracle PII-eliciting directions compared to all other single-token prompts. Prompts optimized to induce the surrogate to emit memorized PII items are also more likely to trigger the target to generate memorized PII, demonstrating both the effectiveness and transferability.

5.5 Impact of Attack Configurations

Now we evaluate whether an optimal configuration for Private Investigator exists that could further increase the number of extracted PII items. The PII extraction in Phase II (§4.3) employs a constant c of prompt score and temperature value during the generation of texts. We vary these factors individually while maintaining prompts unchanged.

Constant c. Figure 9a shows the number of memorized PII items extracted from the two fine-tuned target LMs, GPT-Neo trained on the Enron and TREC datasets, while varying the constant c. The constant c is used to balance the two terms in our prompt scoring function 4.3. As illustrated in the figure, we observe that Private Investigator achieved the best performance when c was set to 4. On the other hand, when c is set to a higher or lower value, focusing primarily on the exploration term or the exploitation term respectively, the attack performance deteriorates. This result indicates that refining the selection of prompts enhances the likelihood of extracting memorized PII items. We used a fixed value of c set to four for all experiments in §5.2.

Temperature. Figure 9b shows the number of memorized PII items extracted by Private Investigator while varying the temperature during text generation in Phase II. The temperature

Table 6: Numbers of extracted PII items using Private Investigator from the GPT-Neo fine-tuned with deduplicated dataset.

Extraction method	Email		Phone		Name	
	Enron	TREC	Enron	TREC	Enron	TREC
Carlini (Top-K)	45	563	73	12	4648	4466.
Carlini (Temp)	40	559	89	16	4606	4512
Carlini (Internet)	38	584	69	21	4544	**5852**
Lukas	20	496	40	24	3981	4168
Ours (Pre)	**50**	609	**94**	17	**5058**	4480
Ours (Fine)	41	**659**	84	**25**	4870	5682

Table 7: Numbers of extracted PII items using Private Investigator from the GPT-Neo fine-tuned with differential privacy.

Extraction method	Email		Phone		Name	
	Enron	TREC	Enron	TREC	Enron	TREC
Carlini (Top-K)	101	26	541	0	7349	1645
Carlini (Temp)	85	22	446	0	7504	**2236**
Carlini (Internet)	58	22	263	0	6870	1588
Lukas	45	11	263	0	5473	1415
Ours (Pre)	**111**	**28**	**633**	0	7943	1558
Ours (Fine)	92	26	627	0	**7954**	1945

value flattens the logit vector of LM's output. A higher temperature value enables the LM to choose tokens with lower probability, thereby diversifying the output. This probability adjustment affects the performance of Private Investigator.

We conducted our attack on two GPT-Neo models trained with the Enron and TREC dataset. We plotted the total number of memorized PIIs that Private Investigator extracted from the two target LMs for two PII types. As shown in the figure, Private Investigator extracted a large number of phone numbers with a temperature of 1, and its performance in extracting email addresses remains consistent within the temperature range of 0.8 to 1.0.

Private Investigator showed high performance in extracting email addresses at a low-temperature setting, while it extracted a relatively smaller number of phone numbers. This performance difference originates from the varying lengths of the target PII items. The average number of tokens for email addresses and phone numbers in the Enron dataset are 9.9 and 6.3, respectively. Sampling tokens with low probabilities increases the likelihood of incorrectly reconstructing longer token sequences. Consequently, a low-temperature value enables Private Investigator to extract a high number of email addresses. To extract a large number of both email addresses and phone numbers, we used 1.0 as the temperature value for all experiments in §5.2.

In contrast, Private Investigator and the other baselines extracted the highest number of personal names at a temperature of 1.4. Therefore, we use 1.4 as the temperature setting for PII extraction attacks targeting personal names.

6 Potential Countermeasures

We test the effectiveness of two representative countermeasures against Private Investigator: training data deduplication and differential privacy (DP).

Training data deduplication. LMs are prone to memorizing (sensitive) records that appear multiple times in the training data. Deduplication reduces the risk of such memorization, thereby lowering the likelihood of data extraction. Prior work [30] has shown that it can also improve model's utility.

The datasets we use for fine-tuning contain a large number of duplicated sequences, primarily due to emails sent to multiple recipients and replies that include the original messages. Following prior work [30], we apply a deduplication threshold of 50 tokens. As a result, we remove ∼799M duplicate sequences from the Enron dataset and ∼112M from the TREC dataset. To assess the impact of deduplication, we fine-tune GPT-Neo for four epochs on the deduplicated datasets.

Table 6 summarizes the attack results against models trained with deduplicated data. Compared to our main results on non-deduplicated models (Table 2), deduplication reduces the risk of extraction across all attack methods. Deduplication is more effective in protecting target LMs fine-tuned on Enron than those fine-tuned on TREC. This is likely due to the substantially larger number of duplicate sequences removed from Enron (∼799M) compared to TREC (112 sequences). However, we still find that Private Investigator generally outperforms the baselines across all datasets and PII types.

Unlike previous research suggesting that deduplication may enhance model performance [30], we observe a notable increase in perplexity after deduplication—from 5.42 to 18.40 on the Enron dataset and from 6.91 to 12.23 on the TREC dataset. We attribute this performance loss to the nature of the dataset, which are sourced from email communications and thus contain a high volume of duplicated sequences. The resulting aggressive deduplication likely removes much useful data, impairing the model's learning.

Differential privacy. DP-SGD is a de-facto standard in training private models [1]. It allows a defender to specify a formal privacy budget ε, which bounds the amount of private information a model can learn during training.

We test the effectiveness of DP-SGD against Private Investigator. To evaluate, we adopt DP-SGD during the fine-tuning process of the target LMs on both the Enron and TREC datasets. We use the `dp-transformers` library [49] to implement the Transformer model training with DP. Following the prior work [33], we set $\varepsilon = 8$. The failure probability δ is set to $1/N$, where N denotes the size of the fine-tuning dataset.

Table 7 summarizes our results. Across all extraction attacks, the number of extracted PII items is significantly reduced. For example, none of the attack methods are able to

extract a single phone number from the GPT-Neo model fine-tuned on the TREC dataset. These results indicate that DP effectively mitigates the risk of Private Investigator. However, applying DP comes at the cost of model utility: perplexity increases from 5.42 to 28.63 on the Enron dataset and from 6.91 to 21.51 on the TREC dataset, confirming a notable degradation in model performance. Despite this, Private Investigator outperforms all baseline attacks across two datasets and three PII types—except for name extraction on the TREC dataset—demonstrating its effectiveness even under DP.

7 Related Work

Privacy attacks. Machine learning models optimize their weights according to the training objectives over training datasets. During training, models can unintentionally memorize specific features of their training dataset [10], which makes the models susceptible to various privacy attacks: membership inference, model inversion, and attribute inference.

Membership inference attacks [7, 8, 40] classify whether a given data example is included in the model's training dataset. It leverages the intuition that machine learning models behave differently on the training dataset. Model inversion attacks [3, 15, 51, 55] aim to reconstruct the representatives of training data corresponding to specific target labels. Attackers can recover training examples by optimizing the input to maximize the confidence score for the target label. Attribute inference [16, 26, 35, 52] attacks infer unknown sensitive attribute values of a given training record.

Extracting training data from LMs. Training data extraction attacks [10] aim to reconstruct training examples. However, training data extraction attacks try to retrieve exact examples (i.e., *verbatim strings*) in the training dataset. This poses a critical threat to users because attackers can extract private information such as email addresses and phone numbers.

Carlini *et al.* [10] demonstrated that LMs are vulnerable to *unintended memorization*, which is persistent and hard to avoid. It enables attackers to easily extract secret sequences. Furthermore, they proposed a new metric called *exposure* that quantifies how vulnerable the model is to such *memorization*. Zanella-Béguelin *et al.* [54] proposed training data extraction attacks in the context of updating language models. They conducted extraction attacks by comparing the differences between two LM snapshots: before and after an update. However, these attacks are limited to targeting small LMs or intentionally making target LMs overfitted.

Carlini *et al.* [11] demonstrated that a real-world large language model (i.e., GPT-2) is vulnerable to training data extraction attacks. They proposed a simple yet performant attack using only black-box query access. Although LLMs do not suffer from overfitting [39], they successfully extract hundreds of private sequences including PII, UUIDs, and URLs. Carlini *et al.* [9] evaluated which LMs have a tendency to memorize training data. They showed that memorization becomes stronger when the model has a larger capacity, a higher number of repetitions in training data, and more prompt tokens. Lukas *et al.* [33] focused on analyzing PII leakage from LMs. Specifically, they introduce extraction, inference, and reconstruction attacks with only API access to LMs.

Defense against training data extraction attacks. Training language models with differential privacy (DP) provides a prevalent and effective defense mechanism against extraction attacks [10, 11, 33, 54]. DP provides strong safeguards regarding the privacy of individual examples in training datasets. The most common way of implementing DP is to adopt the differentially private stochastic gradient descent (DP-SGD) algorithm to the defender's model. Specifically, DP-SGD clips the gradient to a maximum norm and adds Gaussian noise [1]. This technique can successfully mitigate extraction attacks, preventing models from completely memorizing sensitive sequences in the training dataset [10].

Deduplicating training data removes duplicate texts within the training dataset [28, 30]. Lee *et al.* [30] demonstrated that deduplicating training data can enhance the language model's perplexity while reducing privacy risk. PII scrubbing is a data curation technique that replaces PII items with mask tokens [33]. This prohibits LMs from memorizing certain types of PII. Moreover, traditional techniques to mitigate overfitting, such as weight decay, dropout, and quantization [22], can be applied to defend against data extraction attacks, as these regularization techniques prevent overtraining.

8 Conclusion

We study *extractable memorization*, with a particular focus on the emerging deployment scenario of fine-tuned LMs. Unlike prior work that relies on prompts collected from Internet sources resembling the training data distribution or manually-designed ones, our work concerns adversaries who *optimizes* the extraction prompts. Our attacker performs this prompt optimization in a black-box manner. Private Investigator is our framework that enables this study, and we evaluate an approach commonly used in black-box adversarial attacks—leveraging surrogate LMs. For practicality, we select surrogate LMs from publicly available open-source pre-trained LMs. In our evaluation, Private Investigator successfully extracted up to additional 1,254 email addresses, 634 phone numbers, and 5,087 names from the training data across four LMs, outperforming the baseline attacks. We also show the complementary role of Private Investigator in uncovering unique PII items not extracted by previous methods, thereby contributing to a more comprehensive exploration of diverse PIIs within the target training data. Moreover, popular countermeasures against data extraction effectively reduced the number of extracted PII items. But we could still extract $\sim$770 PII items from private models. Our results imply data extraction as a practical threat to language models and highlight the urgent need for future work to develop protections.

Acknowledgments

We are grateful to the reviewers for their constructive feedback. This work was supported by (1) the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2020-II200153), (2) the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-NR076965), and (3) the Samsung Strategic Alliance for Research and Technology (START) Program 2025.

Ethics Considerations

Our research adhered to the ethical guidelines of USENIX Security, ensuring compliance with principles such as beneficence, respect for persons, justice, and adherence to legal and public interests. The primary goal of this research is to highlight the potential privacy risks in LMs. By identifying vulnerabilities, we aim to contribute to the development of safer LM ecosystems. All datasets we used for the experiments, Enron and TREC, are publicly available. Also, we ensure that no details about PIIs in the datasets appear in the paper. We acknowledge the potential misuse of our findings. While the new PII extraction attack could be misused, we believe that the benefits of raising awareness and auditing such vulnerabilities are more important than the risks. Also, we provide recommendations for mitigation strategies against PII extraction attacks to secure LMs.

Open Science

We made all of our artifacts publicly available for reproducible research. The full implementation of Private Investigator, including environment setup, attack execution scripts, and key analysis scripts, is available on Zenodo (https://doi.org/10.5281/zenodo.15653720) and GitHub (https://github.com/WSP-LAB/PrivateInvestigator). All models and fine-tuned weights are publicly accessible through Hugging Face (https://huggingface.co/models). The Enron dataset is available at https://www.cs.cmu.edu/~enron/, and the TREC dataset can be accessed at http://curtis.ml.cmu.edu/w/courses/index.php/W3C_Email_Corpus.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 308–318, 2016.

[2] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 54–59, 2019.

[3] Shengwei An, Guanhong Tao, Qiuling Xu, Yingqi Liu, Guangyu Shen, Yuan Yao, Jingwei Xu, and Xiangyu Zhang. Mirror: Model inversion for deep learning network with high fidelity. In *Proceedings of the Network and Distributed System Security Symposium*, pages 230–244, 2022.

[4] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Proceedings of the Advances in Neural Information Processing Systems*, 2024.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3(1):1137–1155, 2003.

[7] Hui Bo, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. Practical blind membership inference attack via differential comparisons. In *Proceedings of the Network and Distributed System Security Symposium*, 2021.

[8] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 1897–1914, 2022.

[9] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *Proceedings of the International Conference on Learning Representations*, 2023.

[10] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the USENIX Security Symposium*, pages 2633–2650, 2019.

[11] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *Proceedings of the USENIX Security Symposium*, pages 2633–2650, 2021.

[12] Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer. Upper-confidence-bound algorithms for active learning in multi-armed bandits. In *International Conference on Algorithmic Learning Theory*, pages 189–203, 2011.

[13] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 7059–7069, 2019.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

[15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[16] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in Pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *Proceedings of the USENIX Security Symposium*, pages 17–32, 2014.

[17] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2020.

[18] Henry Gilbert, Michael Sandborn, Douglas C. Schmidt, Jesse Spencer-Smith, and Jules White. Semantic compression with large language models. In *IEEE International Conference on Social Networks Analysis, Management and Security*, 2023.

[19] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level adversarial reprogramming. In *Proceedings of the 59th Annual Meeting on Association for Computational Linguistics*, 2021.

[20] Yubo He and Tuefei Zhu. RLTG: Multi-targets directed greybox fuzzing. *PLoS ONE*, 18(4), 2023.

[21] Jie Huang, Hanyin Shao, and Kevin Chang. Are large pre-trained language models leaking your personal information? In *Proceedings of the ICML Workshop on Knowledge Retrieval and Language Models*, 2022.

[22] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. In *Journal of Machine Learning Research*, pages 1–30, 2018.

[23] Bo Hui, Haolin Yuan, Neil Gong, Philippe Burlina, and Yinzhi Cao. PLEAK: Prompt leaking attacks against large language model applications. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 3600–3614, 2024.

[24] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proceedings of the International Conference on Machine Learning*, pages 2137–2146, 2018.

[25] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, Suriya Gunasekar, Mojan Javaheripi, Piero Kauffmann, Yin Tat Lee, Yuanzhi Li, Anh Nguyen, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Michael Santacroce, Harkirat Singh Behl, Adam Taumann Kalai, Xin Wang, Rachel Ward, Philipp Witte, Cyril Zhang, and Yi Zhang. Phi-2: The surprising power of small language models. https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/, 2023.

[26] Bargav Jayaraman and David Evans. Are attribute inference attacks just imputation? In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1569–1582, 2022.

[27] Ellen Jiang, Edwin Toh, Alejandra Molina, Kristen Olson, Claire Kayacik, Aaron Donsbach, Carrie J Cai, and Michael Terry. Discovering the syntax and strategies of natural language programming with generative language models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2022.

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

[29] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *Proceedings of the Conference on Email and Anti-Spam*, 2004.

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

[31] Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. Few-shot learning with multilingual generative language models. In *Proceedings of the Empirical Methods in Natural Language Processing*, pages 9019–9052, 2022.

[32] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 21558–21572, 2023.

[33] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 346–363, 2023.

[34] Chenyang Lyu, Hong Liang, Shouling Ji, Xuhong Zhang, Binbin Zhao, Meng Han, Yun Li, Zhe Wang, Wenhai Wang, and Raheem Beyah. SLIME: Program-sensitive energy allocation for fuzzing. In *Proceedings of the ACM International Symposium on Software Testing and Analysis*, pages 365–377, 2022.

[35] Shagufta Mehnaz, Sayanton V. Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models. In *Proceedings of the USENIX Security Symposium*, pages 4579–4596, 2022.

[36] Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. Openelm: An efficient language model family with open training and inference framewrok. *CoRR*, abs/2404.14619, 2024.

[37] Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. The text anonymization benchmark (TAB): A dedicated corpus and evaluation framework for text anonymization. *Journal of Computational Linguistics*, 48(4):1053–1101, 2022.

[38] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[39] Alec Radford, Jeffrey Wu, Dario Amodei, Daniella Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. Better language models and their implications. https://openai.com/index/better-language-models/, 2019.

[40] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 3–18, 2017.

[41] Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. On transferability of prompt tuning for natural language processing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3949–3969, 2022.

[42] Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. Intellicode compose: code generation using transformer. In *Proceedings of the International Symposium on Foundations of Software Engineering*, pages 1433–1443, 2020.

[43] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, 2017.

[45] Jinghan Wang, Chengyu Song, and Heng Yin. Reinforcement learning-based hierarchical seed scheduling for greybox fuzzing. In *Proceedings of the Network and Distributed System Security Symposium*, 2021.

[46] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In *Proceedings of the Advances in Neural Information Processing Systems*, 2024.

[47] Maverick Woo, Sang Kil Cha, Samantha Gottlieb, and David Brumley. Scheduling black-box mutational fuzzing. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 511–522, 2013.

[48] Yejun Wu, Douglas W. Oard, and Ian Soboroff. An exploratory study of the w3c mailing list test collection for retrieval of emails with pro/con arguments. In *Proceedings of the Conference on Email and Anti-Spam*, pages 238–258, 2006.

[49] Lukas Wutschitz, Huseyin A. Inan, and Andre Manoel. dp-transformers: Training transformer models with differential privacy. https://www.microsoft.com/en-us/research/project/dp-transformers, August 2022.

[50] Bin Yan and Mingtao Pei. Clinical-bert: Vision-language pre-training for radiograph diagnosis and reports generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2982–2990, 2022.

[51] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 225–240, 2019.

[52] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 268–282, 2018.

[53] Weichen Yu, Tianyu Pang, Qian Liu, Chao Du, Bingyi Kang, Yan Huang, Min Lin, and Shuicheng Yan. Bag of tricks for training data extraction from language models. In *Proceedings of the International Conference on Machine Learning*, pages 40306–40320, 2023.

[54] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 363–375, 2020.

[55] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 253–261, 2020.

[56] Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2921–2929, 2021.

[57] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *CoRR*, abs/2307.15043, 2023.

A Prompt Examples

Table 8 shows an example of prompts generated by Private Investigator across different prompt lengths.

Table 8: Prompts generated by Private Investigator using GPT-Neo fine-tuned with Enron data to extract email addresses.

Prompt length	Set of prompts
1	'—–', ' sniff', ' believer', 'checking', ' Sora', ' lobbyist', ' nonsense', ' villain', ' hostile', ' terror', 'worthiness', ' psyche', ' congest', ' NYPD', ' Canaan', ' bordering', ' emperor', 'closed', 'oso', ' courthouse'
2	'—– Anyone', '—– Aren', '—– sounds', '—– unsatisf', '—– makeup', '—–ook', '—–Whoever', '—–Uh', '—– defeats', '—– aren', '—– recons', '—– Nothing', '—– Might', '—– skept', '—– keeps', '—–uphem', '—– couldn', '—–Looks', '—– doesn', '—– heck'
3	'—– Anyone nonetheless', '—– Anyone incor', '—– Anyone reintrodu', '—– Anyone REPL', '—– Anyone 503', '—– Anyone poured', '—– Anyone wond', '—– Anyone buttons', '—– Anyone unob', '—– Anyone stole', '—– Anyone rehe', '—– Anyone adv', '—– Anyone compl', '—– Anyone sho', '—– Anyone 444', '—– Anyone harb', '—– Anyone415', '—– Anyoneospons', '—– Anyone withdrawals', '—– Anyone 311'

B Contextual Similarity

In Table 9, we present the full results of the contextual similarity analysis by comparing the context vectors of PII prefixes exclusively extracted by Private Investigator (PV), the corresponding context vectors from the training data (GV), and the context vectors of PII prefixes exclusively extracted by the baselines (OV). For each PII type, the first and second rows show the average cosine similarity between PV and GV, and the average cosine similarity between OV and GV, respectively. As a result, the cosine similarity between PV and GV is consistently higher than that between OV and GV across all PII types, LMs, and datasets. In the third row, we show the proportion of PIIs for which the cosine similarity between PV and GV is greater than that between OV and GV. The results demonstrate that between 60.7% and 100% of PIIs have higher cosine similarity between PV and GV than between OV and GV. These results highlight that Private Investigator-generated prompts are more contextually aligned with the training data than those generated by the baselines.

C Additional Graphs and Figures

Figures 10, 11 and 12 illustrate Venn diagrams for the total PII items that the baseline methods and Private Investigator extracted from the GPT-2, OpenELM, and PHI-2, respectively.

Figure 13 and 14 demonstrate the cosine similarity between the prompts and the *PII-eliciting context*. The prompts are generated using the surrogate LM, GPT-Neo, which is trained on the Enron and TREC datasets, respectively.

Table 9: Cosine similarity of context vectors for PIIs extracted by the Private Investigator and the baselines.

PII Type	Metrics	GPT2		GPT-Neo		OpenELM		PHI-2	
		Enron	TREC	Enron	TREC	Enron	TREC	Enron	TREC
Email Address	Avg. CosSim(PV, GV)	0.739	0.836	0.910	0.933	0.697	0.778	0.602	0.646
	Avg. CosSim(OV, GV)	0.648	0.701	0.866	0.877	0.365	0.326	0.308	0.419
	CosSim(PV, GV) > CosSim(OV, GV)	72.6%	88.9%	83.0%	93.5%	92.0%	97.3%	85.2%	76.6%
Phone Number	Avg. CosSim(PV, GV)	0.730	0.887	0.896	0.964	0.511	0.878	0.475	0.755
	Avg. CosSim(OV, GV)	0.690	0.652	0.877	0.895	0.294	0.232	0.319	0.332
	CosSim(PV, GV) > CosSim(OV, GV)	66.5%	85.7%	73.8%	100.0%	84.9%	98.8%	77.1%	100.0%
Personal Name	Avg. CosSim(PV, GV)	0.702	0.744	0.882	0.881	0.416	0.453	0.497	0.463
	Avg. CosSim(OV, GV)	0.675	0.705	0.870	0.871	0.356	0.401	0.435	0.402
	CosSim(PV, GV) > CosSim(OV, GV)	61.6%	71.4%	68.1%	65.1%	60.7%	62.8%	73.2%	60.0%

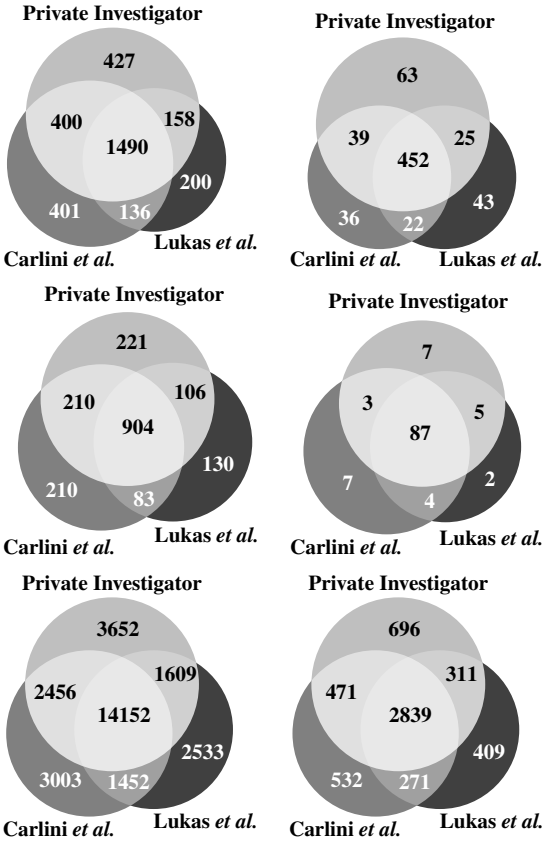


Figure 10: **Visualizing the overlap of PII items extracted from GPT-2 by Private Investigator and two baselines:** Carlini *et al.* (Top-K) and Lukas *et al.* From top to bottom, each row corresponds to the extraction of email addresses, phone numbers, and names. The left column shows results for models fine-tuned on the Enron dataset, while the right shows results for TREC.

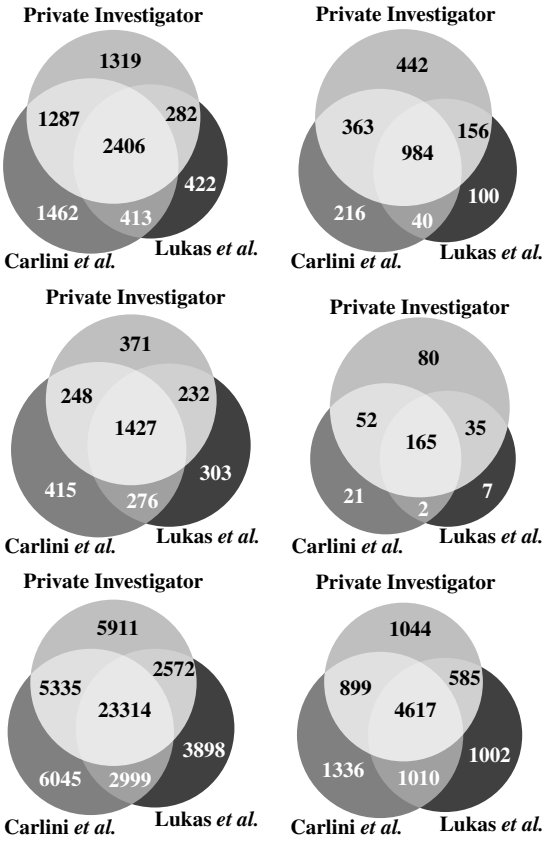


Figure 11: **Visualizing the overlap of PII items extracted from OpenELM by Private Investigator and two baselines:** Carlini *et al.* (Top-K) and Lukas *et al.* From top to bottom, each row corresponds to the extraction of email addresses, phone numbers, and names. The left column shows results for models fine-tuned on the Enron dataset, while the right shows results for TREC.

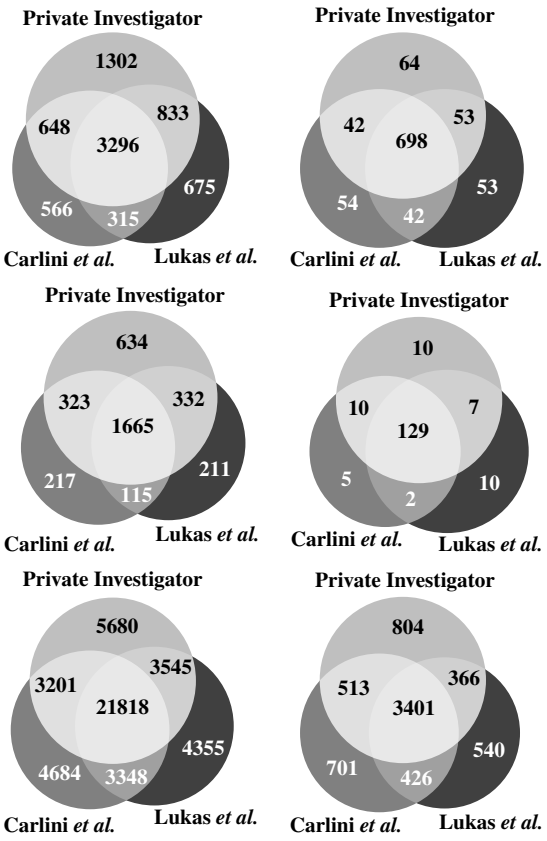
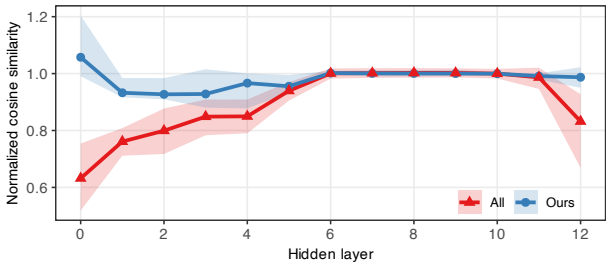
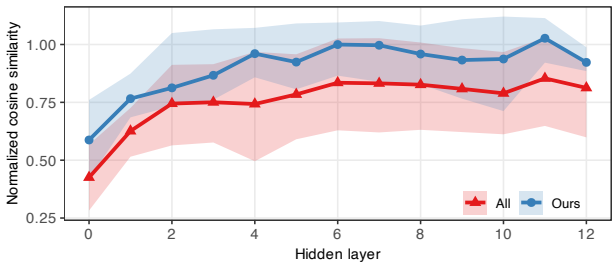


Figure 12: **Visualizing the overlap of PII items extracted from PHI-2 by Private Investigator and two baselines:** Carlini *et al.* (Top-K) and Lukas *et al.* From top to bottom, each row corresponds to the extraction of email addresses, phone numbers, and names. The left column shows results for models fine-tuned on the Enron dataset, while the right shows results for TREC.

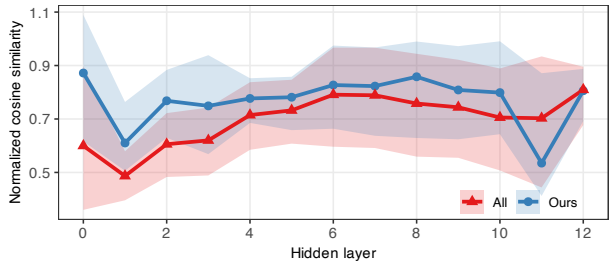


(a) Personal names

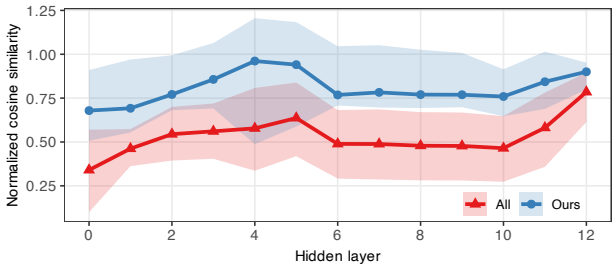
Figure 13: **PII-eliciting directions.** Cosine similarity between the oracle PII-eliciting directions and latent vectors of all single-token prompts (All) or our generated prompts (Ours). We run this analysis on GPT-Neo fine-tuned with Enron. The solid line indicates the median similarity, and the shaded area denotes the interquartile range.



(a) Email addresses



(b) Phone numbers



(c) Personal names

Figure 14: **PII-eliciting directions.** Cosine similarity between the oracle PII-eliciting directions and latent vectors of all single-token prompts (All) or our generated prompts (Ours). We run this analysis on GPT-Neo fine-tuned with TREC. The solid line indicates the median similarity, and the shaded area denotes the interquartile range.