ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX Security ’25 Artifact Appendix: 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

A Artifact Appendix

A.1 Abstract

Private Investigator is an attack framework designed to optimize prompts for querying a Language Model (LM) in order to extract Personally Identifiable Information (PII) used during its fine-tuning process. This artifact provides Docker image to setting up the experimental environment, along with Python programs required to reproduce the results presented in the paper. We tested the artifact on an Ubuntu 22.04 machine equipped with an NVIDIA GeForce RTX 3090 GPU. The experiments require 24 GB of GPU memory, although they can be executed on GPUs with smaller memory capacity by appropriately reducing the batch size. We expect this artifact to reproduce the evaluations shown in §5.2, Table 2, and Figures 2, 10, 11, and 12 of the paper. However, please note that the exact experimental results may vary due to the probabilistic nature of text generation in large language models.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Private Investigator does not have any kind of security risks to the machines that execute it. Furthermore, all the datasets used for the experiments are publicly accessible.

A.2.2 How to access

The artifact is accessible through GitHub (https://github.com/WSP-LAB/PrivateInvestigator) and Zenodo (https://doi.org/10.5281/zenodo.16748364).

A.2.3 Hardware dependencies

Private Investigator requires a Linux machine with an NVIDIA graphics card. Due to the large size of LMs and datasets, we recommend using a machine with at least 32 CPU cores, 128 GB of system memory, 24 GB of GPU memory, and 150 GB of disk space. We tested the Docker image and Python programs on a machine running Ubuntu 22.04 with an NVIDIA GeForce RTX 3090 GPU.

A.2.4 Software dependencies

Private Investigator requires CUDA 12.1 and Python 3.10, along with the following Python libraries: Pytorch 2.1.2, Transformers 4.51.3, and Flair (https://github.com/flairNLP/flair).

The artifact utilizes two datasets (Enron and TREC) and four LM architectures (GPT-2, GPT-Neo, OpenELM, and PHI-2). The Enron dataset can be accessed at https://www.cs.cmu.edu/~enron/, and the TREC dataset is available at http://curtis.ml.cmu.edu/w/courses/index.php/W3C_Email_Corpus. All four LM architectures are publicly shared on Hugging Face (https://huggingface.co/models). All architectures are accessible without any authorization. However the Llama-2 tokenizer–used by the OpenELM model–requires a model access token, which can be obtained from its Hugging Face page: https://huggingface.co/meta-llama/Llama-2-7b. We also share the weights of LMs fine-tuned on the Enron and TREC datasets on Hugging Face: https://huggingface.co/SeonghoKeum.

Instructions for downloading the required software, datasets, and model weights are provided in § A.3.1.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

1. Download the artifact from our GitHub repository: https://github.com/WSP-LAB/PrivateInvestigator.

2. Install Docker and configure it to run Docker without root privileges.

3. Install CUDA Toolkit 12.1. If different version of CUDA is used, update the DockerFile accordingly.

4. To enable GPU access within Docker container, install the NVIDIA Container Toolkit.

5. Build the Docker container by running the script: `build.sh[MODEL_ACCESS_TOKEN`.

6. Launch the Docker container by running the script `launch_container.sh`.

7. Download the datasets by executing the following scripts: `get_enron.sh`, `get_trec.sh`, and `get_commoncrawl.sh`.

8. Download fine-tuned model weights by running the scripts: `get_model_weights.sh` and `get_model_weights_counter.sh`.

For more details, please refer to the `README.md` file included in the artifact.

A.3.2 Basic Test

To verify that all required software dependencies are properly installed, run `basic_test.py` located in the `experiments/` directory. This script specifically tests the TREC dataset and evaluates its perplexity on the OpenELM model fine-tuned on that dataset. If everything is set up correctly, it should print a perplexity value of 3.6.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Private Investigator extracts more PIIs compared to the baselines in 17 out of 24 cases. This is demonstrated by the experiments (E2) and (E3). These experiments are descibed in §5.1 and the results are reported in Table 2 of our paper.

(C2): Private Investigator extracts exclusive PIIs that are not extracted by any other baselines. The claim is proven by Experiment (E4). This experiment is described in §5.2 and illustrated in Figures 2, 10, 11, and 12 of the paper.

A.4.2 Experiments

All scripts and Python files for conducting the experimeents are located in the `experiments/` directory of the artifact. In this section, the directory is omitted in the paths.

(E1): Prompt Generation [600 GPU-hours]: Private Investigator generates prompts and store them under `prompts/` directory.

Preparation: Download all required datasets and model weights, and complete the environemnt setup as described in § A.3.1.

Execution: Run the Python program `run_generate_prompt.py` with the appropriate arguments. It requires the following arguments:

- architecture: gptneo
- model_ckpt: weights/gptneo-enron, weights/gptneo-trec
- surrogate: fine, pre
- target: email, phone, name
- from_scratch: True, False

To generate prompts using the pre-trained surrogate model, omit the model_ckpt argument.

Generating prompts from scratch for all combinations of datasets, surrogate models, and target PII types takes approximately 600 GPU-hours. To reduce computational cost, the program is designed to use intermediate text generation results stored in the `text_generation/` directory by default, which allows prompt generation to complete within an hour. To generate prompts from scratch, explicitly set the from_scratch argument to True.

To generate prompts for all combinations at once, use the following scripts: `run_generate_prompt_interim.sh` (uses intermediate results) `run_generate_prompt_scratch.sh` (generates from scratch).

Results: Generated prompts will be saved in the `prompts/` directory.

(E2): Extract PIIs [200 GPU-hours]: Private Investigator extracts PIIs using the generated prompts.

Preparation: Complete experiment (E1).

Execution: Execute the Python program `run_attack_campaign.py` with the required arguments:

- architecture: gpt2, gptneo, openelm, phi2
- model_ckpt: weights/[gpt2 | gptneo | openelm | phi2]-[enron | trec]
- surrogate: pre, fine
- target: email, phone, name
- c: 0.25
- temp: 1 (for email and phone), 1.4 (for name)

Adjust the eval_batch_size argument to balance GPU memory usage and execution time, if needed.

To run Private Investigator across all combinations of architectures, datasets, and PII types, execute the `run_extract_piis_all.sh` script.

Results: The number of extracted PIIs will be saved in the `all_attack_result.txt` file. These results will be similar to those reported in the Table 2 of the paper. Note that the exact numbers may vary due to the non-deterministic behavior of the LMs.

(E3): Baselines [400 GPU-hours]: Run four baseline attacks: Carlini *et al.* (Top-K), Carlini *et al.* (Temp), Carlini *et al.* (Internet), and Lukas *et al.*.

Preparation: Download all required datasets and model weights, and complete the environemnt setup as described in § A.3.1.

Execution: To run the Carlini *et al.* baselines, execute `baseline_carlini.py` with the following arguments:

- model_path: weights/[gpt2 | gptneo | openelm | phi2]-[enron | trec]
- method: topk, temperature, internet
- target: email, phone, name
- temp: 1 (for email and phone), 1.4 (for name)

The batch_size argument can be adjusted to balance GPU memory usage and execution time.

To runt the Lukas *et al.* attack, execute `baseline_lukas.py` with the following four arguments:

- architecture: gpt2, gptneo, openelm, phi2
- model_ckpt: weights/[gpt2 | gptneo | openelm | phi2]-[enron | trec]
- target: email, phone, name
- temp: 1 (for email and phone), 1.4 (for name)

Similar to the previous attacks, generation batch size could be controlled by the eval_batch_size argument.

To run all experiments at once, execute the following scripts: `run_baselines_email.sh`, `run_baselines_phone.sh`, and `run_baselines_name.sh`.

Results: All baseline results will be saved in the `all_attack_result.txt` file. Note that the results may not exactly match those reported in the paper due to the probability based text generation.

(E4): PII overlap [less than one hour]: Count the number of PIIs that are exclusively or commonly extracted by each attack.

Preparation: Complete the experiments (E1), (E2), and (E3).

Execution: Execute `analyze_pii_overlap.py`.

Results: The numbers of exclusively and commonly extracted PIIs will be saved in the `overlap.txt` file.

Instructions for additional analysis experiments are provided in the `README.md` file included in the artifact.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.