USENIX Security '26 Artifact Appendix: Membership Inference Attacks on Tokenizers of Large Language Models

Meng Tong[1 *] Yuntao Du[2 *] Kejiang Chen[1 ✉] Weiming Zhang[1] Ninghui Li[2]

[1] *University of Science and Technology of China* [2] *Purdue University*
[*] *Equal Contribution* [✉] *Corresponding Author*

A Artifact Appendix

A.1 Abstract

Membership inference attacks (MIAs) are widely used to assess the privacy risks associated with machine learning models. However, when these attacks are applied to pre-trained large language models (LLMs), they encounter significant challenges, including mislabeled samples, distribution shifts, and discrepancies in model size between experimental and real-world settings. To address these limitations, we introduce tokenizers as a new attack vector for membership inference. Specifically, a tokenizer converts raw text into tokens for LLMs. Unlike full models, tokenizers can be efficiently trained from scratch, thereby avoiding the aforementioned challenges. In addition, the tokenizer's training data is typically representative of the data used to pre-train LLMs. Despite these advantages, the potential of tokenizers as an attack vector remains unexplored. To this end, we present the first study on membership leakage through tokenizers and explore five attack methods to infer dataset membership. We conduct extensive experiments verify the dataset-level membership leakage from tokenizers.

A.2 Description & Requirements

A.2.1 Hardware Requirements

The experiments were conducted on a system equipped with at least **32GB of RAM** to support the training of shadow tokenizers and dataset sampling.

A.2.2 Software Requirements

The artifact is designed to run on Linux-based distributions. The following software environment is required:

- **Package Manager:** Conda (Miniconda or Anaconda)
- **Language:** Python 3.12
- **Libraries:** HuggingFace `tokenizers`, `transformers`, and the dependencies specified in `requirements.txt`.

Environment Setup To replicate the environment, execute the following commands:

```
conda create -n MIA python=3.12
conda activate MIA
pip install -r requirements.txt
```

A.2.3 Security, privacy, and ethical concerns

The introduction of a new attack vector for MIAs could be used to infer personal information and lead to privacy violations. However, our experiments solely utilize public,non-sensitive datasets collected by Google. We emphasize that our proposed attacks are never intended to facilitate or encourage the privacy leakage from LLMs.

A.2.4 How to access

We are committed to upholding the principles of open science by making our findings freely accessible. Our codes are available at https://zenodo.org/records/18080228. It is recommended to download the files on linux computer.

A.2.5 Hardware dependencies

None.

A.2.6 Software dependencies

The experimental environment is managed using Conda. Users are required to have Conda (or Miniconda) installed. The installation of conda are referred to https://www.anaconda.com/download. The specific Python dependencies are listed in `requirements.txt`, which is available at https://zenodo.org/records/18080228.

A.2.7 Benchmarks

We utilize real-world web data from the C4 corpus in our evaluation. We have provided the involved datasets at https://zenodo.org/records/18080228.

A.3 Set-up

A.3.1 Installation

To replicate the environment, execute the following commands to set up the Python environment and install the required dependencies:

```
conda create -n MIA python=3.12
conda activate MIA
pip install -r requirements.txt
```

A.3.2 Basic Test

After the environment is correctly configured, run the following command in the terminal to implement a membership inference attack via vocabulary overlap:

```
python train_target_tokenizer.py
python train_shadow_tokenizer.py
python mia_via_vocabulary_overlap.py
```

The vocabularies of target tokenizers are stored in the `trained_tokenizer` directory. Also, the generated shadow tokenizers are saved in the `shadow_tokenizer` directory. The evaluation results and performance of membership inference are saved in the `infer_results` directory.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *Tokenizers are vulnerable to advanced membership inference attacks, as demonstrated by experiment E1 (Section 5.2) and the results in Table 3. For instance, MIA via Vocabulary Overlap and MIA via Frequency Estimation yield AUC scores exceeding 0.70 when evaluated on a target tokenizer with a vocabulary size of 200,000.*

(C2): *Expanding a tokenizer's vocabulary increases its susceptibility to membership inference attacks. This correlation is validated by experiment (E1) and the corresponding results reported in Table 3.*

A.4.2 Experiments

(E1): *[Comparison of MIAs against target tokenizers] [30 human-minutes + 24 compute-hour + 30GB disk]:*
Preparation: To set up the experimental environment, create a dedicated Conda environment and install the dependencies as follows:

```
conda create -n MIA python=3.12
conda activate MIA
pip install -r requirements.txt
```

The datasets are in the `website_data/` folder.

Execution: The experiment follows training target tokenizers, training shadow tokenizers, and executing the membership inference attacks:

```
python train_target_tokenizer.py
python train_shadow_tokenizer.py
python mia_via_compression_rate.py
python mia_via_vocabulary_overlap.py
python mia_via_frequency_estimation.py
python mia_via_merge_similarity.py
python mia_via_naive_bayes.py
```

The training scripts will generate tokenizers in the `trained_tokenizer` and `shadow_tokenizer` folders.
Results: Upon completion of the attack scripts, all experimental outcomes, including inference accuracy and performance metrics, are automatically aggregated. The results are saved in the `infer_results` directory.

A.5 Notes on Reusability

The membership inference attacks can also be executed on custom datasets. To perform this, users should place the target dataset in the `website_data` directory.

A.6 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/usenixsec2026/.