A Artifact Appendix

A.1 Abstract

The artifact is an open-source Python library that implements a novel framework to evaluate the privacy-utility trade-off of synthetic data publishing and to compare it to that of traditional sanitisation techniques. The library provides implementations of two privacy attacks to evaluate privacy gain with respect to the risk of linkability and inference. It further includes implementations of five example generative models, three standard models and two models trained under formal privacy guarantees.

The artifact contains experiment scripts and configuration files to reproduce some of the results presented in the paper. In particular, the example runs described in the README allow the user to partially reproduce the graphs of Section 6 that compares the privacy-utility trade-off of (differentially private) synthetic data to traditional anonymisation.

A.2 Artifact checklist

- **Algorithm:** The privacy (and utility) games introduced in the paper are implemented in the corresponding command line interface (`cli` files.)
- **Data set:** The repository contains a copy of the cleaned-up and pre-processed dataset used for the main set of experiments. The dataset and the required metadata can be found in the data folder under `texas.csv` and `texas.json`, respectively.
- **Run-time environment:** Synthetic Data is also distributed as a ready-to-use Docker image containing Python 3.9 and CUDA 11.4.2, along with all dependencies required by Synthetic Data.
- **Hardware:** When running evaluations for either the `PATE-GAN` or `CTGAN` model it is useful to have a GPU at hand. This significantly speeds up the execution. However, they are not needed for running the example experiments.
- **Execution:** The README includes instructions about how to run three example experiments. The evaluation under the linkage risk model is the most compute- and memory-intensive. On a machine with an Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz with 2 cores (with hyperthreading) this should take around 1h15m.
- **Output:** The example experiments produce output files in a `json` format and can be parsed with the functions provided in `utils/analyse_rersults`. We include a simple jupyter notebook that allows to visualise and analyse the results.
- **Experiments:** The repository includes experiment configuration for three key experiments. See further below of the README of the repository.
- **How much disk space required (approximately)?:** If using the dockerised deployment, its image requires ~14GB of disk space. The experiment outputs need
- **How much time is needed to prepare workflow (approximately)?:** <1h (but strongly depends on the bandwidth of the connection used to pull the Docker image).
- **How much time is needed to complete experiments (approximately)?:** This depends on the compute power available. On a machine with an Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz with 2 cores (with hyperthreading) it should take 3h to run all example experiments.
- **Publicly available?:** The code is publicly available at https://github.com/spring-epfl/synthetic_data_release/tree/v1.1
- **Code licenses (if publicly available)?:** The code is distributed under a BSD-3-Clause License.
- **Data licenses (if publicly available)?:** See https://www.dshs.texas.gov/THCIC/Hospitals/Download.shtm

A.3 Description

The library has two main classes: `GenerativeModels` and `PrivacyAttacks`. For both classes we define a parent class that determines the core functionality that objects of the class need to implement.
`GenerativeModel` provides two main functions. `GM.fit(R)` is called with a raw dataset `R` as input and implements the model's training procedure. `GM.sample(m)` generates a synthetic dataset `S` of size `m`. The library enables easy integration of existing model training procedures. `GM.fit` simply wraps any existing training algorithm and exposes the appropriate API endpoints.

`PrivacyAttack` objects have two functions: `PA.train` and `PA.attack`. `PA.train` trains the adversary's guess function and needs to be run before calling `PA.attack`. `PA.attack(S)`, takes a dataset `S` and outputs a guess about a secret value. In our implementation, we instantiate `PrivacyAttack` with two attacks, a linkage adversary and an attribute inference attack.

The library also includes procedures to estimate the privacy gain of synthetic and sanitised data publishing. These procedures can be found in the corresponding `cli` files.

A.3.1 How to access

The code, some example data and experiment configuration files are publicly available at https://github.com/spring-epfl/synthetic_data_release/tree/v1.1.

A.3.2 Dependencies

For your convenience, Synthetic Data is also distributed as a ready-to-use Docker image containing Python 3.9 and CUDA 11.4.2, along with all dependencies required by Synthetic Data.

Note: This distribution includes CUDA binaries, before downloading the image, ensure to read its EULA and to agree to its terms.

A.3.3 Data sets

The repository contains a copy of the cleaned-up and pre-processed dataset used for the main set of experiments, the Texas hospital dataset. The dataset and the required metadata can be found in the data folder under `texas.csv` and `texas.json`, respectively.

The Texas Hospital Discharge dataset is a large public use data file provided by the Texas Department of State Health Services. The dataset we include here for the experiments consists of 100,000

records uniformly sampled from a pre-processed data file that contains patient records from the year 2013 and 2014. We retain 18 data attributes of which 11 are categorical and 7 continuous.

A.4 Installation

Synthetic Data can either be installed from scratch or run in a dockerised environment. If you want to use the ready-to-use docker container, pull the image and run a container (and bind a volume where you want to save the data):

```
docker pull springepfl/synthetic-data:latest
docker run -it -rm -v "$(pwd)/output:/output"
springepfl/synthetic-data
```

The Synthetic Data directory is placed at the root directory of the container.

```
cd /synthetic_data_release
```

You should now be able to run the examples without encountering any problems.

A.5 Experiment workflow

We provide three example experiments and their configurations to reproduce some of the key claims presented in Section 6 of the paper.

Privacy gain with respect to linkability. First, to run a privacy evaluation with respect to the privacy concern of linkability you can run

```
python3 linkage_cli.py -D data/texas -RC
tests/linkage/runconfig.json -O tests/linkage
```

The results file produced after successfully running the script will be written to `tests/linkage` and can be parsed with the function `load_results_linkage` provided in `utils/analyse_results.py`. A jupyter notebook to visualise and analyse the results is included at `notebooks/Analyse Results.ipynb`.

Privacy gain with respect to inference. To run a privacy evaluation with respect to the privacy concern of inference you can run

```
python3 inference_cli.py -D data/texas -RC
tests/inference/runconfig.json -O tests/inference
```

The results file produced after successfully running the script can be parsed with the function `load_results_inference` provided in `utils/analyse_results.py`. A jupyter notebook to visualise and analyse the results is included at `notebooks/Analyse Results.ipynb`.

Average machine learning utility. To run a utility evaluation with respect to a simple classification task as utility function run

```
python3 utility_cli.py -D data/texas -RC
tests/utility/runconfig.json -O tests/utility
```

The results file produced after successfully running the script can be parsed with the function `load_results_utility` provided in `utils/analyse_results.py`. The jupyter notebook contains code for visualising the results.

A.6 Evaluation and expected results

Privacy gain with respect to linkability. This experiment allows you to compare the privacy gain for five outlier records from the Texas dataset with respect to the risk of linkability for three different privacy mechanisms: traditional sanitisation (`SanitiserNHSk10`), synthetic data produced by a standard Bayesian Network (`BayNet`) and a differentially private version of this model (`PrivBay`). The differentially private model is trained with a privacy parameter of $\varepsilon = 1.0$. All other model hyperparameters match the ones used in the paper.

After loading the results into the notebook named `Analyse Results.ipynb`, you can inspect the per-target privacy gain for five outlier records from the Texas dataset under a linkage attack with varying feature sets. You should observe that, as described in the paper, the privacy gain of most targets is larger under the `BayesianNet` model compared to traditional sanitisation and further increases if the synthetic data is sampled from the differentially private model (compare with Fig.4 in Section 6.1 in the final version of the paper). You can choose under which attack feature set you want to compare the targets' privacy gain.

Note: Due to the sampling uncertainty and randomness of the attack and generative model training process, you should expect slight variations between the observed privacy gain and the exact values reported in the final publication. Furthermore, the observed variance of the per-record privacy gain is likely larger than the one reported in the final publication. This is because the privacy game is run for a smaller number of iterations to reduce the computation time of the experiments. To reduce the reported standard variation, you can modify the parameter `nIter` in the configuration file `tests/linkage/runconfig.json`.

Privacy gain with respect to inference. Similarly, the privacy gain of the same target records under the same models with respect to the risk of inference can be evaluated with the results file written to `tests/inference/`. The results should be comparable to the data presented in Fig. 6 of Section 6.2.

Utility loss under a classification task. This experiment allows you to compare the utility loss of sanitised and synthetic data publishing under a simple classification task as utility function. The details of this experiment are described in Section 6.3.1. Here, we want to compare the accuracy of a machine learning classifier trained on a sanitised or synthetic dataset to that of a classifier trained on the raw data. You should observe how training on data with a higher privacy gain, i.e., (differentially private) synthetic data, leads to a decline in the model's accuracy compare to the raw data.

A.7 Experiment customization

To change the evaluation parameters, you can modify the `runconfig.json` files in the corresponding experiment folders. For instance, to change the size of the raw and synthetic datasets, respectively, you can modify the parameters `sizeRawT` and `sizeSynT` for each experiment file.

To evaluate the privacy gain of synthetic data publishing under a different generative model, a new `GenerativeModel` class has to be integrated. See A.3 for more details on the implementation of this class. If you want to run the evaluation on an entirely new dataset, a metadata file in `.json` format is necessary. You can use the `texas.json` metadata file as a template.