Using Machine Learning to Assist Output Checking

Josep Domingo-Ferrer, Alberto Blanco-Justicia

Universitat Rovira i Virgili,
Department of Computer Engineering and Mathematics,
CYBERCAT-Center for Cybersecurity Research of Catalonia,
UNESCO Chair in Data Privacy,
Av. Països Catalans 26, 43007 Tarragona, Catalonia
{josep.domingo,alberto.blanco}@urv.cat

December 2021

Outline

1. Introduction
2. Rewriting checking rules for synthetic log generation
3. Generation of synthetic training and test data
4. Experimental work
5. Conclusions and future research

Introduction

- There is an increasing demand by researchers to access high-quality microdata. Privacy legislation, however, limits their release. Usually, controllers release anonymized microdata [5, 2].
    - Yet, anonymization entails information loss. Researchers often require access to the original microdata.
- Some controllers offer safe access centers, which are controlled environments where researches run their analyses under monitoring. The controller's staff check whether any output leaks personal information from the respondents [1, 4].

Introduction

- Highly expert output checkers can follow the so-called **principles-based model**, where checkers collaborate with researchers and take the entire context of the analysis into account to make a decision on whether an output is safe enough to be returned or not.
- An easier alternative that requires less interaction and expertise by the checkers is the **rule-based model**: the checker uses simple rules of thumb to label an output as safe or unsafe. The price paid is a higher probability of false positives and false negatives.

Introduction

Contribution and plan of this paper

- We propose to relieve some of the burden of rule-based output checking by (partially) automating it via a ML approach.
- We create synthetic output checking log files based on subsets of rules from [1] and use them to train ML models.
- Then, we examine how well the rules we used have been learned and, more importantly, how the rules *not* used to generate the log file have also been learned.
- Our results show that our deep learning approach can generalize the rules embedded in the training data, and hence captures the general flavor of safe and unsafe outputs.

Rewriting checking rules for synthetic log generation

We take the rules proposed in [1] to decide whether an output can be safely returned to the researcher.

Example – Class 1: Frequency tables

- Each cell of the table must contain at least 10 units (unweighted).
- The distribution of units over all cells in any row or column is such that no cell contains more than 90% of the total number of units in that particular row or column (group disclosure)
- No cell in a table is formed from units of whom 90% or more originate from one organisation.

Rewriting checking rules for synthetic log generation

We rewrite the rules in terms of the following attributes:

- *AnalysisType* is the kind of analysis the rule refers to, for example, Frequency Table, Magnitude Table, Mode, etc.
- *Output* is the result to be returned to the analyst.
- *Confidential* indicates whether the attributes contributing to the analysis are confidential or not.
- *Context* are a set of attributes that describe the analysis. In the previous example, one of these contextual attributes is the minimum number of units contributing to each of the cells in the table.
- *Decision* is the result of a boolean expression, that takes the previous attributes as inputs and returns whether the output can be released or not.

Rewriting checking rules for synthetic log generation

The previous example results in the following rule:

- RULE 1:
  - AnalysisType: FrequencyTable
  - Output: $[0, 1]$
  - Confidential: YES/NO
  - CellUnits: Integer
  - PercentageRows: $[0, 1]$
  - Decision: YES if (!Confidential OR (CellUnits $>=$ 10 AND PercentageRows $<$ 90%)).
- CellUnits and PercentageRows are contextual attributes.

Generation of synthetic training and test data

- We next discuss how to generate synthetic data from the above rules that can be used to train and test a ML model.
- We need to derive a record schema of fixed length that can describe the decisions made by all the above rules.
- However, there are some outputs in the above rules that have a variable number of components, for example, frequency tables (Rule 1).
    - To deal with that problem, we split those rules into rules that *separately apply to each single output component*.
    - Then we can create post-processing rules whereby the entire output is only returned to the researcher if all its components are labeled as safe.

Generation of synthetic training and test data

- Additionally, each of the rules require different contextual attributes. For example:
  - For Rule 1, we need a context attribute *PercentageRows*.
  - For Rule 2, we need a context attributes *CellUnits*, *PercentageRows*, and *PercentageCellTotal*.
- Therefore, the schema that can describe the decisions made by all rules is formed by the following superset of attributes: *AnalysisType*, *Output*, *Confidential*, *CellUnits*, *PercentageRows*, *PercentageCellTotal*, *SampleSize*, *PercentageSampleTotal*, *Intercept*, *DegreesOfFreedom*, *DegreesOfFreedom2*, *NumberOfVariables* and *Decision*.

Generation of synthetic training and test data

- Given the above schema, a synthetic record to describe an instance of a certain Rule *i* can be generated as follows:
  1. Initialize *AnalysisType* to the analysis corresponding to Rule *i*.
  2. Randomly choose an output that is compatible with the analysis type. *E.g.* in Rule 13_s the output is a correlation coefficient and hence it must lie in the interval $[-1, 1]$.
  3. Randomly set *Confidential* to YES or NO.
  4. Randomly choose context attributes that fit the expected semantics for the analysis type. *E.g.* in Rules $9a_s$ and $9b_s$, *Intercept* must be YES/NO, whereas the other context attributes must be blank.
  5. Finally, compute *Decision* according to the decision algorithm for the rule.

Experimental work

- We took the rules and unified similar ones having the same decision algorithm. This left us with 14 total rules.
- Then, we generated a synthetic data set with $200,000$ training samples, with each of the 14 rules contributing $14,700$ samples, half of which with positive decisions.
- We trained feedforward neural network with 2 hidden layers of 64 units each and obtained a 94.08% accuracy, a 4.2% false positive rate and a 7.4% false negative rate.
- We are interested in a low FPR, since false positives are those that are dangerous for the privacy of the respondents.

Experimental work

- Next, we conducted a series of experiments to find out if models can generalize when exposed to samples generated from rules it has not been exposed to during training.
    1. We generated a testing data set that contains samples for the 14 rules. This testing data set was used for all experiments.
    2. Then, from a number of $1 \leq n \leq 14$, we generated 100 training data sets using random subsets of n rules, which yielded $1,400$ data sets with $200,000$ training samples each.
    3. We trained a NN like the one described above for each of the training data sets and tested it against the previously described testing data set.

Experimental work

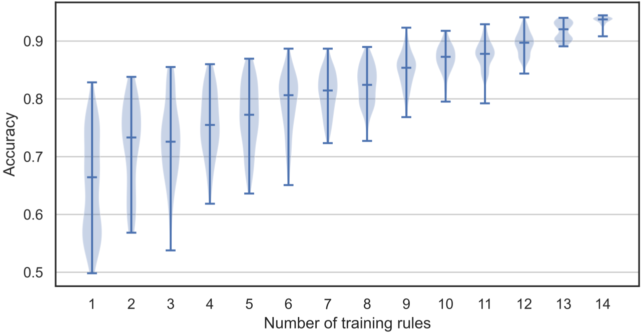

Figure: Accuracy of the models with respect to the number *n* of rules used to generate the training sets

Experimental work

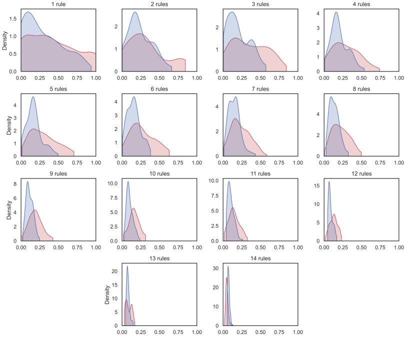

Figure: False positive rate (red) and false negative rate (blue) for a number of rules used to generate the training sets ranging from $n = 1$ to $n = 14$

Experimental work

- We are especially interested in the FPR, since it corresponds to released outputs that might reveal private information about the respondents.
    - While the FPR measures the privacy risk, the FNR measures the utility loss, because it corresponds to outputs that are not released even though they could be usefully returned to analysts without privacy risk.
- As expected, both rates decrease as more rules are considered when generating the training data sets.

Conclusions and future research

- We have presented an approach that leverages machine learning to assist experts in output checking at safe data access centers.
- Our system follows the rule-based model, and we show that it can generalize the rules it is trained on.
- A limitation of the presented research is that it does not use real log files. Future research will strive to gather such data to further validate our approach.
- We also aim at increasing the level of automation of the entire process, leveraging the code of the analysis to derive needed inputs. Also, extending automation to the principles-based model is also an important and daunting challenge.

References I

- S. Bond, M. Brandt and P.-P. de Wolf. *Guidelines for the Checking of Output Based on Microdata Research*. Deliverable D11.8, project FP7-262608 “DwB: Data without Boundaries”, 2015. `https://ec.europa.eu/eurostat/cros/system/files/dwb_standalone-document_output-checking-guidelines.pdf`

- J. Domingo-Ferrer, D. Sánchez and J. Soria-Comas. *Database Anonymization: Privacy Models, Data Utility, and Microaggregation-Based Inter-Model Connections*. Morgan & Claypool, 2016.

- *General Data Protection Regulation*. Regulation (EU) 2016/679. `https://gdpr-info.eu`

References II

- E. Griffiths, C. Greci, Y. Kotrotsios, S. Parker, J. Scott, R. Welpton, A. Wolters and C. Woods. *Handbook on Statistical Disclosure Control for Outputs (version 1.0)*. Safe Data Access Professionals Working Group, 2019. https://ukdataservice.ac.uk/media/622521/thf_datareport_aw_web.pdf

- A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Gießing, E. Schulte-Nordholt, K. Spicer and P.-P. de Wolf. *Statistical Disclosure Control*. Wiley, 2012.

- Ocean Protocol, accessed October 30, 2021. https://oceanprotocol.com