DNSFilter
David Elkind, Chief Data Scientist

Where we're going, we don't need (negative) labels

Positive-Unlabeled Machine Learning for Computer Security

Background

A common problem in machine learning for computer security is the scarcity of benign labels. Threat feeds and services such as VirusTotal and Hybrid Analysis provide labels for known malicious portable executables and websites, but there is no equivalent service for labeling **benign** software and websites. Dedicating expert time to reviewing and labelling benign software is can be very expensive, especially when one desires a very large dataset.

This poster is inspired by the talks given at CAMLIS 2023, each of which mentioned in passing that they were not able to find a large-scale, authoritative resource for benign labels to support training their machine learning models and had to resort to some heuristic approach to creating labels.

Whereas the standard machine learning approach requires labels for both the positive and negative class (PN learning), positive unlabeled (PU) learning requires labels only for the positive class. The second class is simply unlabeled data – a mix of positives and negatives in some unknown proportion.

PU Learning Method

We use the TED^n procedure (Garg *et al.*) to simultaneously estimate the proportion of positives among the unlabeled data and to train a model that is reliably able to detect malicious objects.

The TED^n procedure is composed of three parts.

1. The model is trained for 0 or more warmup iterations in a PN fashion (as if the unlabeled data were all truly negative examples)
2. Best Bin Estimator (BBE) estimates the proportion of positives among the unlabeled data by estimating smallest value such that all of the model's predictions greater than that value are positive. The authors provide a statistical guarantee for this estimator.
3. Conditional Value Ignoring Risk (CVIR) filters the training data to exclude any unlabeled data which have a model prediction greater than the BBE estimate. In this way, the training data have the most likely positive values removed & the model bias is decreased.

TED^n Training terminates when training error stabilizes.

Data & Experiments

The goal of this research is to show that PU learning is more effective than PN learning when there is not a reliable source of negative labels.

This research applies positive-unlabeled learning to two computer security problems

1. Detecting malicious domain names (Namgung *et al.*)
2. Detecting malicious portable executable files (Anderson & Roth, 2018)

In experiments 1 and 2, we simulate PU data by assigning all negatives and a random sample of positives to the unlabeled class.

In experiment 3, we combine Ember's unlabeled and negative partition to create a new, enlarged unlabeled partition.

The same neural network model architectures are used for each experiment; we only vary the training procedure used.

Results

The experimental results are displayed visually. Dotted lines show the 95% confidence interval for the benchmark PN model, which is trained without all PN data and without any label corruption.

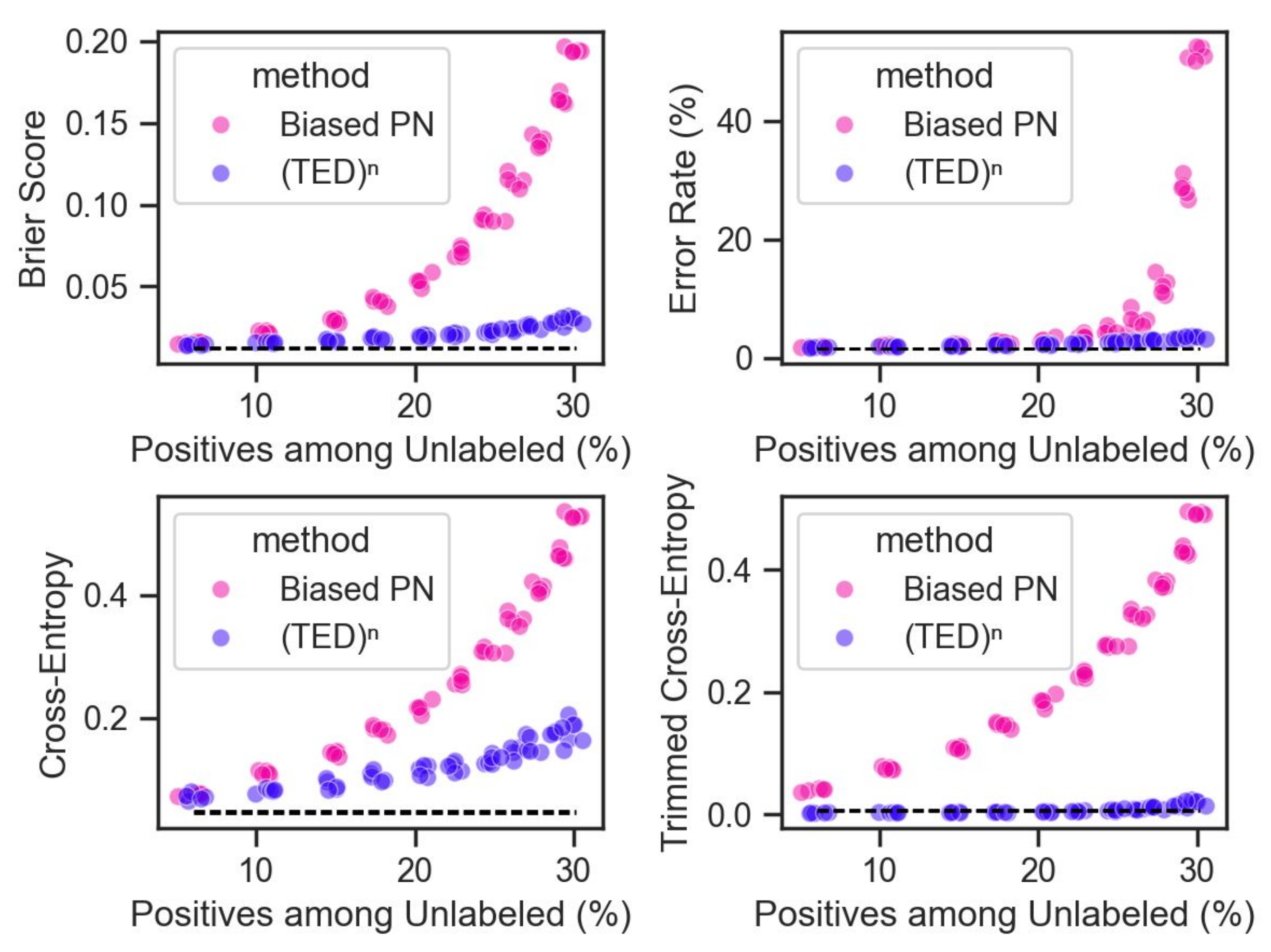
We also perform hypothesis tests in the form of two-sample, paired t-tests. We chose the paired test due to the dependent nature of the data – each fold from 5-fold cross-validation has the same underlying training data between the Biased PN and TED^n procedure.

The results for the DGA model (Experiment 1) clearly show that the TED^n procedure performs better than the Biased PN method. Statistical tests reveal that in all but one case, the TED^n model has a statistically significant improvement over the Biased PN method.

The results for Ember (Experiments 2 and 3) are somewhat more mixed. Hypothesis tests reveal that only the Brier score and trimmed cross-entropy are statistically superior to the Biased PN method.

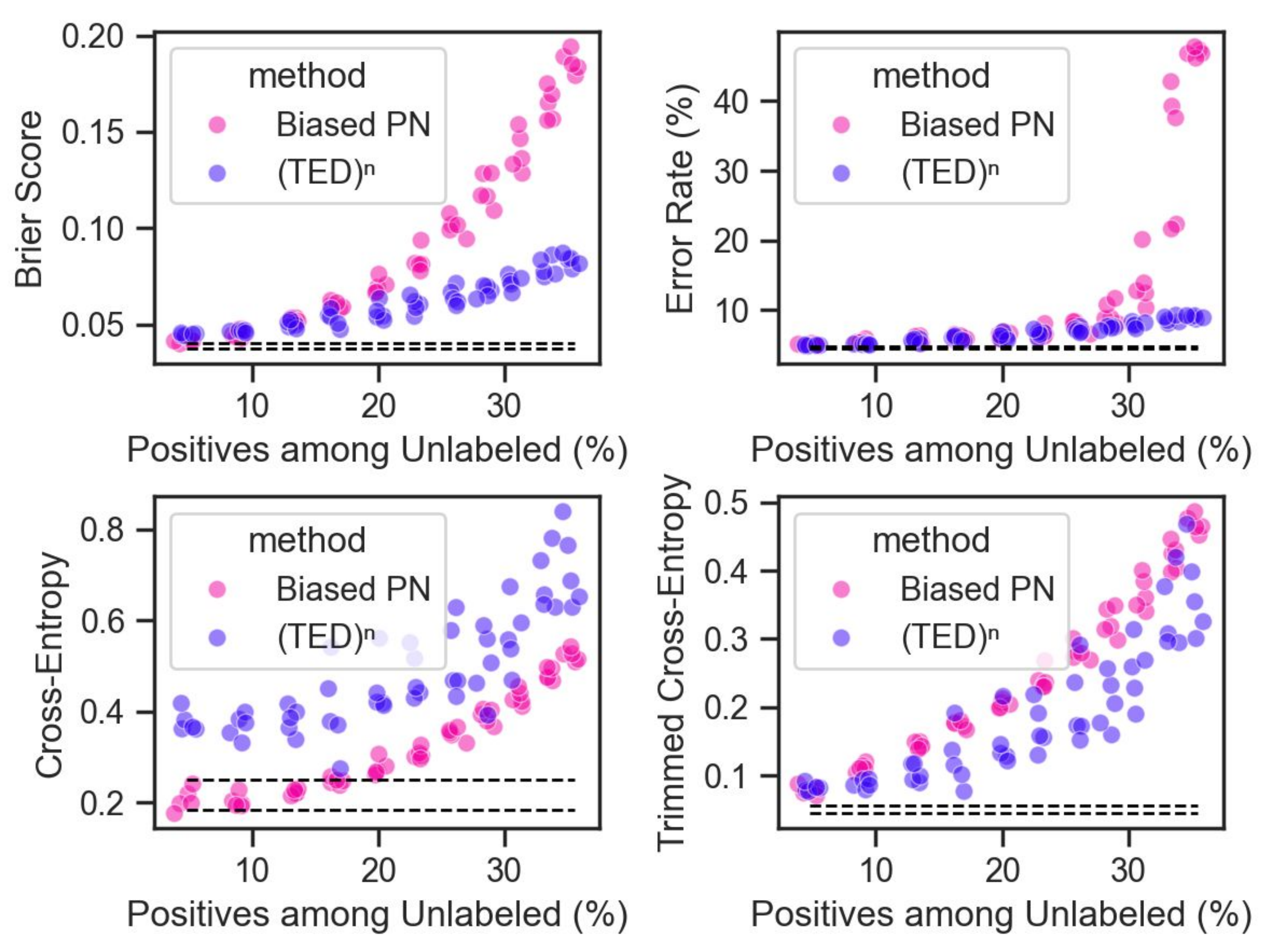
DGA Results

Experiment 1 (DGA)

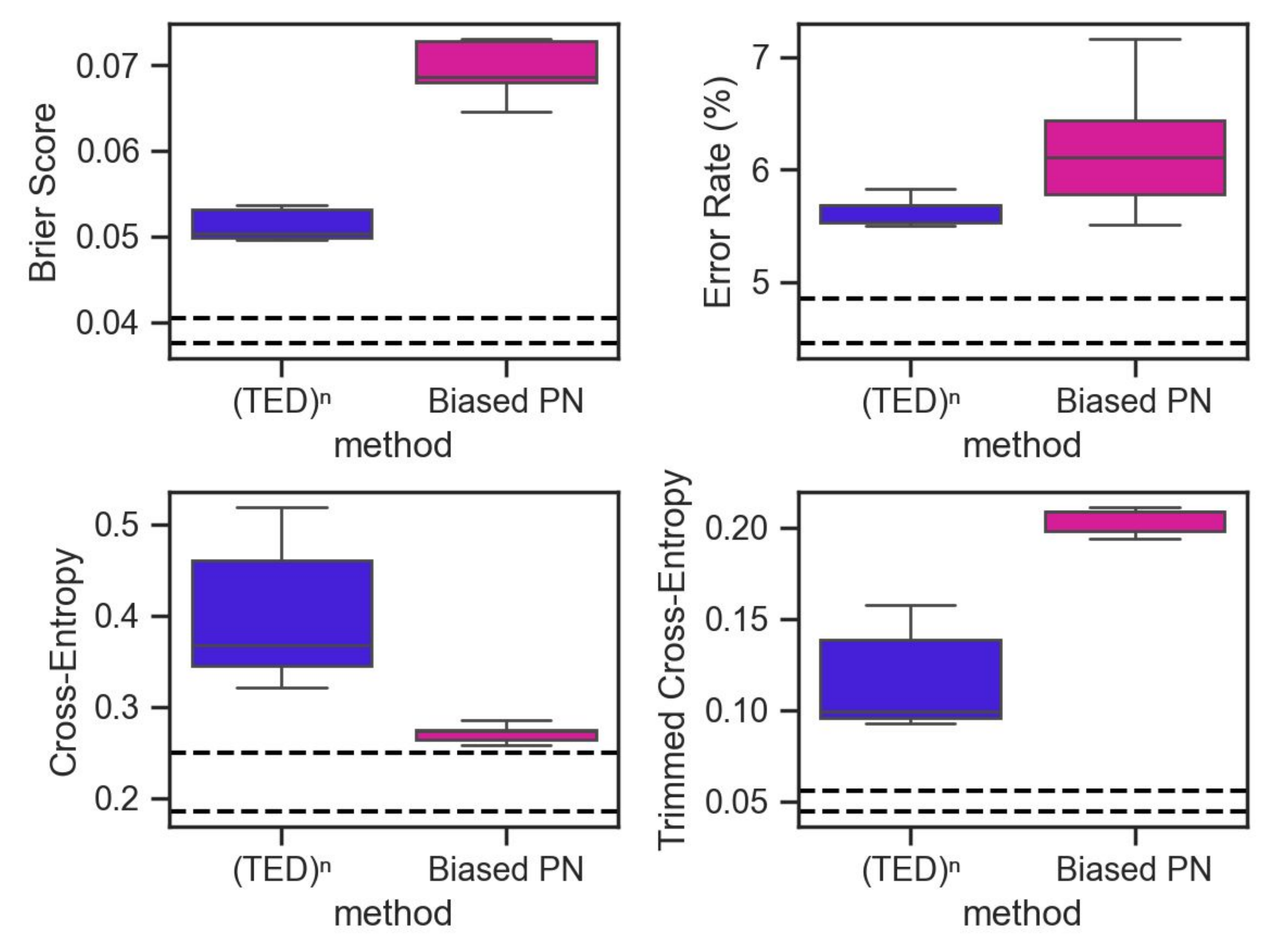


Ember Results

Experiment 2 (Ember)



Experiment 3 (Ember)



Conclusions

In all cases, the most pronounced improvements in model quality occur when there is a larger proportion of positives among the unlabeled data. This is intuitive -- when the proportion of labeling errors is smaller, the labeling errors will have a smaller influence on what the model learns.

The DGA results demonstrate that the TED^n procedure can be extremely effective. The results for the DGA task demonstrate an almost uniform improvement to models in the PU setting. We attribute this success to the overall effectiveness of the modeling strategy: 3-grams are tremendously effective as a method of representing domain name strings for DGA detection.

On the other hand, the TED^n procedure produces mixed results on the EMBER task. In turn, we believe that the lower quality of the model applied to EMBER data degrades the TED^n procedure, because the model is not precisely able to screen out positive instances from the unlabeled data. We attribute this to the poor performance of the neural network model; even our unbiased PN benchmark model (trained without any label noise) performs worse than the original EMBER model from 2017. Anderson & Roth report a ROC AUC of 0.99911 for their gradient boosted tree model trained on the EMBER data. Our benchmark model only obtains ROC AUC 0.9852 (the mean of 5-fold cross validation) and a 95% confidence interval of [0.9841, 0.9863].

We conjecture several potential causes for this discrepancy.

- We use 5-fold cross-validation of the training data to construct a validation set used for early stopping. The 20% reduction in training data size could impair the model.
- This research uses neural networks because the iterative nature of NNs complements the iterative TED^n training procedure. Anderson & Roth used gradient boosted decision trees; perhaps this finding is further evidence that neural networks are less effective than gradient boosted decision trees on tabular data. We used NNs because they easily fit with TED^n
- The EMBER code is a few years out of date, so we were forced to use newer library versions and kludge a few lines of code. This may have degraded the quality of the feature vectors.

Future work should study the efficacy of the TED^n procedure in conditions that are closer to the real-world setting of industry practitioners. Both datasets are very small by the standards of industry. An industrial-scale machine learning model for computer security would use hundreds of millions to billions of samples. The DGA dataset has 1.4 million samples and the EMBER dataset has 1.1 million samples. Collecting more data is often the shortest path to improving model quality, and the TED^n procedure removes the costs for labeling benign data.

References

Anderson, Hyrum S., and Phil Roth. "EMBER: an open dataset for training static PE malware machine learning models." arXiv preprint arXiv:1804.04637 (2018).

Garg, Saurabh, et al. "Mixture proportion estimation and PU learning: A modern approach." *Advances in Neural Information Processing Systems* 34 (2021): 8532-8544.

Namgung, Juhong, Siwoon Son, and Yang-Sae Moon. "Efficient deep learning models for DGA domain detection." *Security and Communication Networks* 2021.1 (2021): 8887881.