DGA Detection with

Machine Learning and Deep Learning

**Lixi Zhou, Haimin Zhang, Shiqi Dai**

{zhou.lix, zhang.haim, dai.shi}@husky.neu.edu

CSYE 7245, Fall 2018, Northeastern University

# Abstract (10 %)

## Research idea

DGAs can constantly generate large amounts of domains to evade blacklist detection. It is hard to block the connection between hackers and zombie computers in real-time with traditional ways. In order to solve this problem, we decided to use machine learning algorithms to detect DGAs and compare the performance of these algorithms.

## Methods

We first performed feature engineering.

Then applied preprocessed data to machine learning models like a random forest, support vector machine, naive Bayes classifier, CNN, LSTM.

## results

# Introduction (10 %)

Internet security vendors have provided several strategies to intercept DGA traffic. In traditional, security providers would first decode the algorithm by applying reverse engineering. Generating a list of domains with a given seed, then preregister, sink-holed or put them into a DNS blacklist to prevent potential C2 traffic. Another common strategy is to find similar domain groups by using their statistical properties to determine if DGA generates a domain. The main disadvantage of traditional strategies is the lack of capability to be used for real-time detection and protection.  
  
Therefore, several strategies based on machine learning are introduced. FANCI is one of them. FANCI stands for Feature-based Automated NXDomain Classification and Intelligence, and it was introduced in 2018 by Schüppen, Teubert, Herrmann, and Meyer. It is a system for detecting infections with domain generation algorithm based malware by monitoring non-existent domain responses. FANCI mainly uses two supervised learning algorithms, random forests and support vector machine. Because of the using of RF and SVM, all the domain data(text) has to be represented by features. Schüppen et al.,(2018) described 21 features and used three different categories to group their features. They are structural features, linguistic features, and statistical features. Structural features have to subcategories as inherent structural features and non-self-explanatory structural features. Non-self-explanatory structural features can be some boolean type features or calculated ratio features. Linguistic features are used to measure the deviations from common linguistic patterns of domain names. Statistical features are n-gram frequency distribution and entropy which are common approaches in the feature engineering of domain data. According to Schüppen et al.,(2018), FANCI is based on supervised learning classifiers. It requires training with labeled data. Thus the first module is a training module. The output of the training module is a trained model, then the next module--classification module will use the model to classify new input data. Before classifying, classification module will also perform some preprocessing like feature extraction. In the end, the intelligence module will supply intelligence based on classification results, in particular, find infected devices and identify new DGAs or unknown seeds. FANCI is a very flexible system. There are two main usage scenarios, all-module using and distributed using.  
  
However, the above traditional machine learning approaches have to use manually picked features to create classifiers like FANCI. They usually have two significant drawbacks: First, hand-crafted features are easy to circumvent by hackers. Second, getting hand-crafted features is relatively time-consuming at the runtime. Thus deep learning/neural network approaches have been taken seriously nowadays. Specific neural networks require less feature engineering and perform better at the run-time. They work directly on raw domain names with a minimal transformation. In other words, if a new family of DGA shows up, the classifier can be retrained right away without the need for manual feature engineering. Also, neural network models act like the black box so it is hard for hackers to reverse and beat. Second, deep learning models have better "true positive"/"false positive" rate and real-time performance. In test cases, neural network classifiers are usually able to achieve satisfying accuracy.

For this research, we have three datasets in total: one dataset for Benign Domains, Alexa Top 1 Million Sites, which are a combination of good domains; two datasets for DGA Domains (<https://www.kaggle.com/cheedcheed/top1m>), Bambenek Consulting provided malicious algorithmically-generated domains (<http://osint.bambenekconsulting.com/feeds/dga-feed.txt>) and 360 Lab DGA Domains (<https://data.netlab.360.com/feeds/dga/dga.txt>). Combing these three datasets and shuffling them to generate the dataset for the project.

# 

Figure 1. Original dataset

# Methods

# Feature Engineering

For the machine learning part, only the attribute of the domain itself is not enough for a machine learning algorithm. It needs some features. Applying features engineering first. Based on our knowledge and reference materials, three kinds of features will be generated: Structural Features; Linguistic Features; Statistical Features. For the first part of feature engineering:

|  |  |  |
| --- | --- | --- |
| Features | Ex: prata.pt | Ex: tbaxcrnxirtmuusq.eu |
| DNL (Domain Name Length) | 8 | 19 |
| NoS (Number of Subdomains) | 1 | 1 |
| SLM (Subdomain Length Mean) | 5.0 | 16.0 |
| HwP (Has www Prefix) | 0 | 0 |
| HVTLD (Has a Valid Top Level Domain) | 1 | 1 |
| CSCS (Contains Single-Character Subdomain) | 0 | 0 |
| CTS (Contains Top Level Domain as Subdomain) | 0 | 0 |
| UR (Underscore Ratio) | 0.0 | 0.0 |
| CIPA (Contains IP Address) | 0 | 0 |

Table 1. Structural Features

From Table 1, nine structural features are generated. For the example, prata.pt, DNL (The length of the domain name) is 8. It only has 1 subdomain, so its NoS value is 1. The length of the subdomain (SLM) is the length of ‘prata’, which equals to 5.0. It does not have www Prefix, so its Hwp value is 0. ‘.pt’ is a valid top-level domain, so its HVTLD domain is 1. It does not contain single-character subdomain, so the CSCS value is 0. So does the CTS. The ratio of underscore (UR) for example is 0 also. And it does not have an IP address.

|  |  |  |
| --- | --- | --- |
| Features | Ex: prata.pt | Ex: tbaxcrnxirtmuusq.eu |
| contains\_digit (Contains digit) | 0 | 0 |
| Vowel\_ratio (The ratio of vowel) | 0.4 | 0.25 |
| Digit\_ratio (The ratio of vowel) | 0.33 | 0.0 |

Table 2. Linguistic Features

Based on linguistic analysis, three linguistic features are generated from the domain. Whether a domain contains a digit (contains\_digit), the ratio of the vowel in a domain and the ratio of the digit. The value of these linguistic features can be known from Table 2.

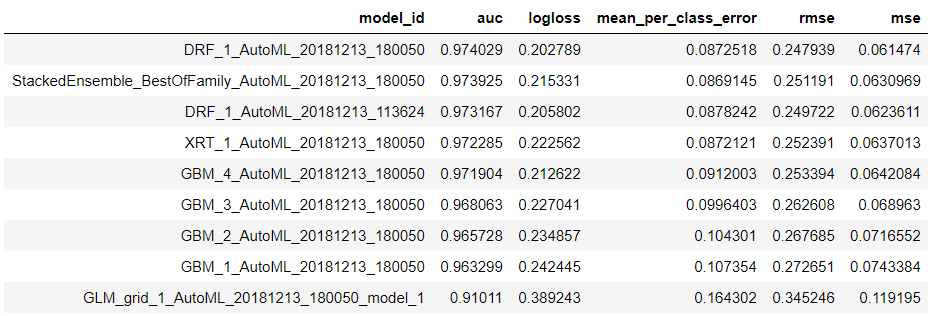
|  |  |  |
| --- | --- | --- |
| Features | Ex: prata.pt | Ex: tbaxcrnxirtmuusq.eu |
| RRC (The ratio of repeated characters in a subdomain) | 0.25 | 0.33 |
| RCC (The ratio of consecutive consonants) | 0.4 | 0.625 |
| RCD (The ratio of consecutive digits) | 0 | 0 |
| Entropy (The entropy of subdomain) | 1.92 | 3.5 |

Table 3. Statistical Features

There are also 4 statistical features will be generated. From Table 3. RRC represents the ratio of repeated characters in a subdomain. RCC represents the ratio of consecutive consonants, RCD represents the ratio of consecutive digits and Entropy means the entropy of subdomain.

## AutoML: H2O

H2O has an industry leading AutoML functionality that automatically runs through all the algorithms and their hyperparameters to produce a leaderboard of the best models. One disadvantage of the H2O Auto Machine Learning is that because it needs to use a lot of models to find the best model of the dataset, so it will take a lot of time to find the best model. As for the project, the size of the whole dataset is about 2.5 million. The first time we run H2O Auto Machine Learning, H2O took about 40 minutes to find the best model with the accuracy of 95%. For this training, only using half of the whole dataset. By putting the whole 2.5-million dataset into H2O Auto Machine Learning function, it takes about 2 hours to finish the fitting and predicting part. Luckily, the latest training reached the accuracy of 97.40%.



Figure, the leaderboard of H2O Auto Machine Learnin

Random Forest(RF)

Random forest is a bunch of decision trees. It can be seen as an ensemble model. A random forest model will take all predicting results from its inner decision trees as a vote.

## SVM(Support Vector Machine)

SVM is able to classify input data by using the computed hyperplane which is trained from the training set. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples.

## Naive Bayes Classifier

## Neural Network Models

### Data Representation

Our data are domains which can be seen as semi-structured data.

### CNN(Convolutional Neural Network)

Typically, CNN is used on image/audio data. It plays a vital role in cognitive computing like image/voice recognition. But there are some approaches to text classification that use CNN as the classifier.

### LSTM(Long Short-Term Memory Neural Network)

Long short-term memory (LSTM) units are units of a recurrent neural network (RNN). An RNN composed of LSTM units is often called an LSTM network.

### Bias and Variance Trade-off

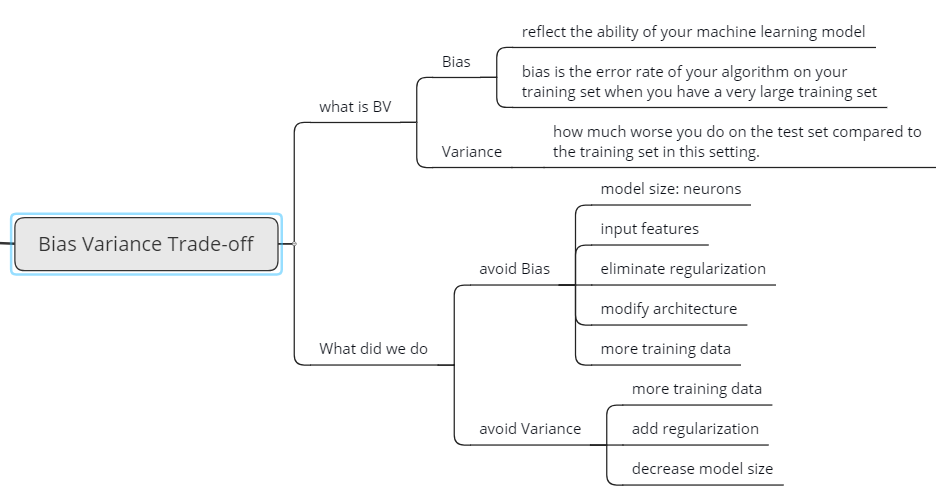
During training a neural network model, we always faced difficulties on fine-tuning. There are so many configuration options for us. With which specific configuration that returns the best model becomes the most confusing decision. This is why we use bias and variance trade-off as the guide during fine-tuning models. Bias refers to the error rate of the neural network model trained on the training set when the training set is large enough. It reflects the capability of your model, the lower the bias you get, the better the model is. On the other hand, the variance is used to measure how much worse the model performs on the testing set compared to on the training set. Big variance indicates the serious overfitting the model has met. The ideal model is the one with both small bias and variance. In other words, well-fitted models with high accuracy are those perfect model we want in this project.

In our experiments, we faced high bias as well as high variance. Techniques were taken to avoid these situations. To avoid high bias, five ways are recommended:

1. Enlarge the size of your model.  
   This means you can increase the number of neurons in layers. This will directly improve the learning ability of your neural network for more trainable parameters.
2. Improve the quality of input features.  
   Many times, the reason that your model doesn’t work well may not come from the model itself. It is probably because your features lack abilities to represent your data well. For example, in our project, we choose character-level embedding to represent data.
3. Eliminate regularization  
   Some models add regularization method like l2 regularizer and dropout layer, to prevent overfitting. If the regularization is too much, it will directly lead to the lack of capability of learning. At this time, trying to reduce regularization might be a good choice.
4. Modify the architecture  
   High bias may also come from the simplicity of your model architecture. You can try a larger number of layers or layer with more sophisticated functionalities. But the modification on architecture can be difficult. You should think carefully before changing the architecture.
5. Train the model on more data  
   If more trainable data available, try to use more data to train the model. It remarkably reduced the bias in our experiments.

For reducing variance, there are also a few techniques we used in our experiments and would like to share:

1. Decrease model size  
   High variance means overfitting. The model has too many parameters which allow it to learn perfectly from the training data. Thus the training accuracies look great, yet when it comes to testing data, whole new data that never seen before by the model, the true, not that good, accuracies become what you get. Opposite to what we mentioned above, at this time, you may decrease the number of neurons inside your model to reduce its learning ability.
2. Add regularization  
   Another useful method is adding regularization. In our experiments, we applied l2 regularizers on Conv2D layers.
3. More training data  
   Last but not least, you can also try to add more data to your training set. It really helps!

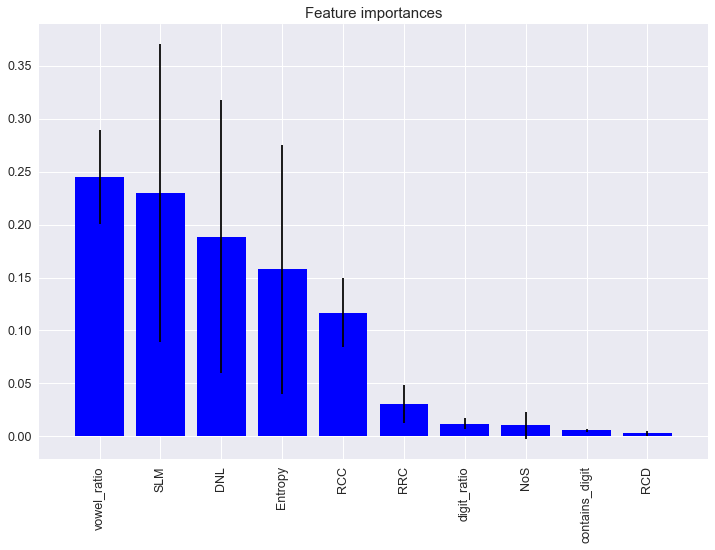


# Results (10 %)

Random Forest

After applying the Random Forest model, training dataset gets 92.23% accuracy and test dataset get 91.65% accuracy. One advantage Random Forest model is that the time for building and fitting a model is not too much. So, it can finish its training part quickly.

Besides the performance of Random Forest model, the graph of importance for each attribute in the model has also been generated. From the figure below, vowel\_ratio, SLM, DNL play the most important roles in the final decision.



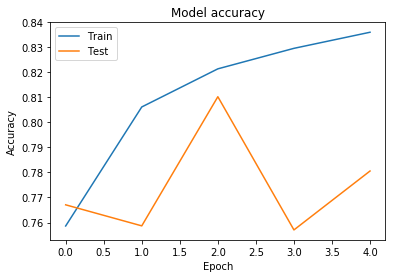
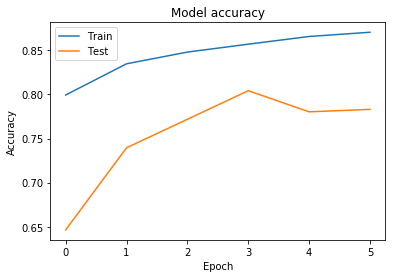
Figure, the importance of each attribute in the Random Forest model

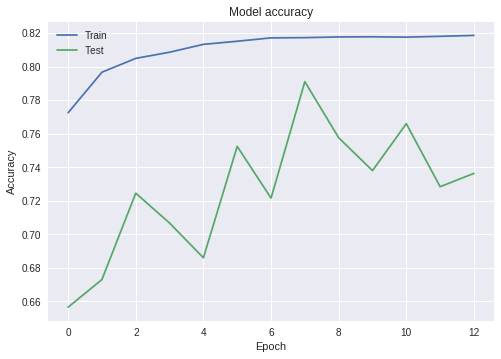
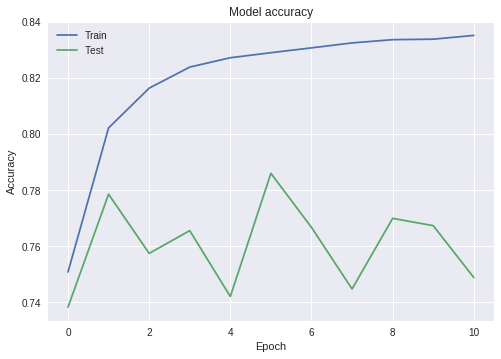
SVM

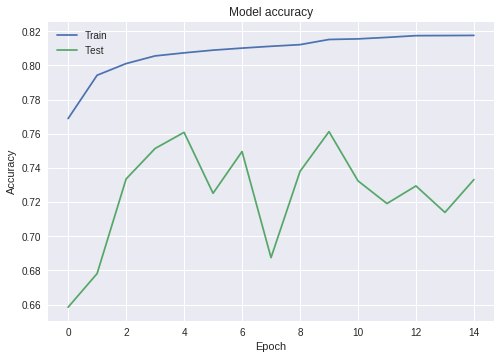
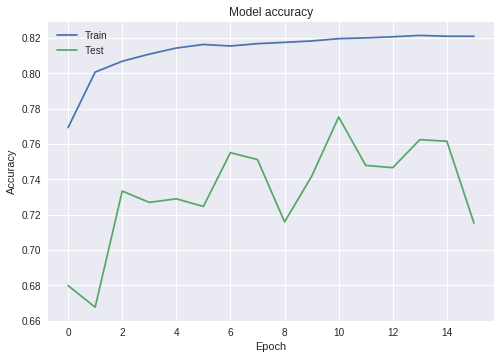
Naive Bayes

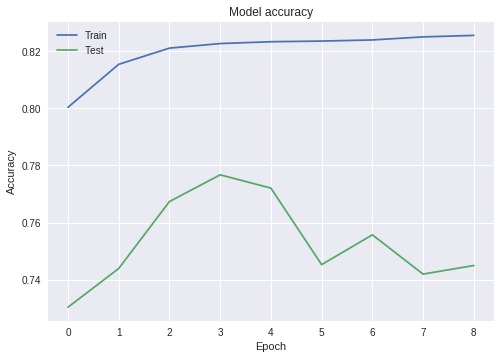
H2O

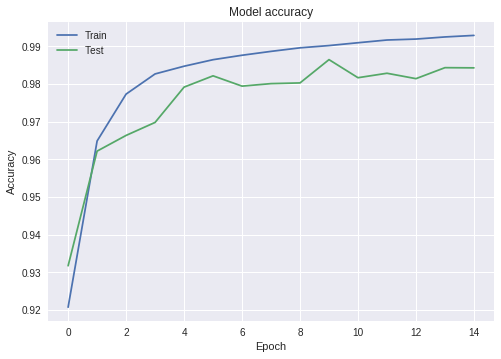
CNN









LSTM  


# Discussion (10 %)

We will talk about the model selection here.

# References (5 %)

Our project is using MIT license

Cite papers, code and have a clear license of your work

[1] Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., & Dagon, D. (2012, August). From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In USENIX security symposium (Vol. 12).  
  
[2] AWS | Alexa Top Sites-Up-to-date lists of the top sites on the web. (n.d.). Retrieved from http s://aws. amazon.com/alexa-top-sites/  
  
[3] Domain generation algorithm. (n.d.). Retrieved from https://www.wikiwand.com/en/Domai n\_ generation\_algorithm  
  
[4] G. (2018, November 05). Google-research/bert. Retrieved from https://github.com/google-res earch/bert  
  
[5] H2O – Data Resource Portal. (n.d.). Retrieved from https://www.northeastern.edu/datares o urces/h2o  
  
[6] H. A., & J. W. (2018, February 22). Using Deep Learning to Detect DGAs. Retrieved from h t tps://www.endgame.com/blog/technical-blog/using-deep-learning-detect-dgas  
  
[7] Koehrsen, W. (2018, June 02). Automated Feature Engineering in Python – Towards Data Science. Retrieved from <https://tow>ardsdatascience.com/automated-feature-engineeri ng-in-pyt hon-99baf11cc219  
  
[8] Plohmann, D., Yakdan, K., Klatt, M., Bader, J., & Gerhards-Padilla, E. (2016, August). A Comprehensive Measurement Study of Domain Generating Malware. In USENIX Security Symposium (pp. 263-278).  
  
[9] Schiavoni, S., Maggi, F., Cavallaro, L., & Zanero, S. (2014, July). Phoenix: DGA-based botnet tracking and intelligence. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 192-211). Springer, Cham.  
  
[10] Schüppen, S., Teubert, D., Herrmann, P., Meyer, U., & Sch, S. (2018, August). FANCI: feature-based automated NXDomain classification and intelligence. In Proceedings of the 27th USENIX Conference on Security Symposium (pp. 1165-1181). USENIX Association.  
  
[11] Tran, D., Mac, H., Tong, V., Tran, H. A., & Nguyen, L. G. (2018). A LSTM based framework for handling multiclass imbalance in DGA botnet detection. Neurocomputing, 275, 2401-2413.  
  
[12] Woodbridge, J., Anderson, H., Ahuja, A., & Grant, D. (2016). Predicting Domain Generation Algorithms with Long Short-Term Memory Networks.  
  
[13] Yu, B., Pan, J., Hu, J., Nascimento, A., & De Cock, M. (2018). Character Level Based Detection of DGA Domain Names.