DGA DETECTION

Identification of domain names generated from Domain Generating Algorithms (DGAs)

Author: Suyash Sathe

Contact: 0470372165

Date: 13 December 2019

Contents

1. Introduction

The main objective of the DGA (**Domain Generating Algorithm**) is to allow communication between a malware and its command and control server. It is a technique used in computer malware programs to generate large number of domain names. These domain names are generated periodically out of which one is the address of the control server and is used to communicate with the command and control server. The infected computer scans through these domains iteratively and hits the target domain which is the address of its command and control server.

The objective of this project is to select the best machine learning model to predict the domain names that are generated from DGAs and prevent the plausible cyber-attack. In this analysis, a dataset containing approximately 130,000 domains names was explored and analysed to obtain the best set of features that are crucial for predicting the domain names generated from DGAs. These features are converted into vectorized format for the purpose of building machine learning models.

This vectorized representation of features was then utilised to train 3 different classification models including Logistic Regression, Random Forest and SVM, and the performance of these models was evaluated based the accuracy score and the recall value. After comparing the models, the best model is utilised to predict the domain names that are generated from DGAs.

2. Pre-Processing and Feature Generation

Starting step for any natural language processing problem is to pre-process the textual information and transform the same into a format which could be easily utilized by the machine translation algorithms to learn information from the corpus and assist in finding an optimal solution for the NLP task in hand. Majorly the different steps and tasks associated in the process of text processing are as below.

2.1 Raw Data Analysis

It is always best to understand and get the high-level information about the data on which we are trying to implement any machine learning algorithms. It is required as it guides in the direction of selecting the most appropriate machine learning algorithm which could be used for that problem and dataset. Different types of analysis were performed on the dataset as part of the research work including checking the dimensionality and shape of the dataset to get an overview of the dataset, analysing the statistical details of the data of URLs, comparison between legitimate and DGA generated domain names and analysis of distribution of length of domain names.

It is clearly observed from the distribution of URL lengths that the distribution of length of legitimate URLs is slightly right skewed. This indicates that the legitimate URLs are short in length typically 10-20 characters. As the length of the URL increases beyond 20 characters, the probability of it being generated by a DGA becomes more.

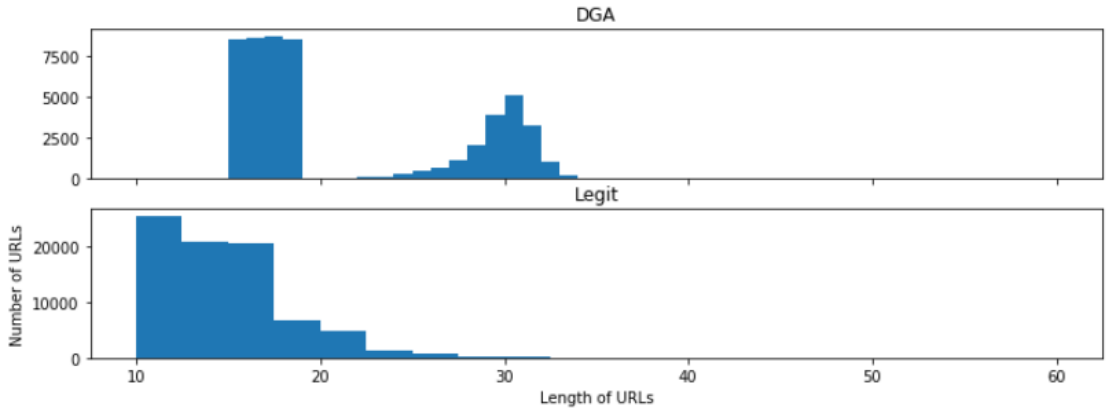

Fig 1: Distribution of length of domain names

The above hypothesis is supported by the distribution of the length of DGA host names. It is observed from the above graph that the length of DGA generated domain names is symmetrically distributed around the length of 30 characters. This indicates that if the length of the URL is around 30 characters, then it is highly likely generated using a DGA. However, this is not always true since there are a significant number of URLs whose length is between 15-20 characters.

2.2 Data Transformation and Reduction

This is one of the most crucial steps in the complete text pre-processing phase. Text in its raw format is difficult to be used by the machine learning algorithms hence it becomes very important for it to be transformed into a format which could be used easily by the algorithms along with providing the ability to perform some statistical analysis and operations.

The first step in the data vectorization process is to tokenize the data. Since the *domain* column of the *dga_domains* columns miss the vital information including the domain extension (.com, .net, etc.) which may be crucial to detect the DGA generated domain name, a customised function *makeTokens* is defined to tokenize the URL into the components. The *makeTokens* function creates tokens by splitting the URL by "/", "-" and "." if present in the URL. This function is used as a tokenizer in the *CountVectorizer* object to represent the tokens or features into vectorized format. It also removes the unnecessary tokens like ".com" which may become redundant features if used to train the model.

Another important step is to convert the word tokens into vectorized form for the machine learning algorithm to use these vectors easily for processing. In this project, CountVectorizer is used to convert the extracted features into vectorized format. All the above steps can be collected and implemented in one go as a part of the default CountVectorizer object. This object performs all the above operation together and returns the sparse matrix of features. However, a custom tokenizer (*makeTokens*) is used in this project instead of a default CountVectorizer object because as it yielded better model performance.

The result achieved after performing all the steps on raw dataset is getting a sparse matrix which could be used for further model building purposes. This can be converted to array object or dataframe object but due to computational limitations, it is better to use the sparse matrix when required in the model building.

3. Model Development

The vectorized representation of the data is utilised for the development of model. Before the model implementation, Sklearn's **train_test_split** function is used to split the data into training set and testing set. The data is split in the ratio of 8:2 where the training data is 80% while the testing data is 20%. In this section 3 types of models are analysed including:

- Logistic Regression Model
- Random Forest Model
- SVM Model

3.1 Logistic Regression Model

In this attempt, an instance of the logistic regression model is created and fed with the sparse vector word representation of the labelled training dataset in the form of *CountVector* sparse matrix. This model is used to predict the labels of the test dataset and the output is recorded.

3.1.1 Model Implementation

Firstly, the tokenised features of the labelled data set are extracted to train the model on these features so that the model learns the hidden weights of the text data distribution. The logistic model instance is created and fed with the sparse vector word representation of the labelled data in the form of the sparse matrix.

Once the model learns the data parameters, it is used to predict the classes of the test data set.

Cross validation

While doing so, 10-fold cross validation is also performed to generalize the model about the sample data point estimation. In this step, the training data is divided into 10 equal parts and 9 out of these 10 parts are passed to train the model. The trained model is used to predict the labels for the 10^{th} part and misclassification rate is computed. Every time, we pass different 9 parts of the data such that the model predicts the label for each data point at least once. The average of misclassification rate of all the 10 parts is used to evaluate the overall performance of the model. Cross validation is done to get an idea of how the model will generalize to about of sample data point estimation. This improves the generalization of the model, prevents overfitting and usually improves the accuracy.

In this step, the average accuracy recorded using 10-fold cross validation (**0.905**) is very close to the accuracy obtained using the traditional 80-20 split (**0.916**) indicating that the dataset is evenly distributed. Since the data is evenly distributed and cross-validation is computationally expensive, it is feasible to use 80-20 train-test split for model training and validation.

3.1.2 Model Assumptions and Advantages

The logistic regression model assumes that all the features of the data set are independent even if they are correlated. However, it is a great classifier as it does not assume the normality, homoscedasticity or linearity in the data.

The advantages of logistic regression model are:

1. Easy to implement.
2. Computationally efficient.
3. No unrealistic assumptions like the normality of the data.
4. No need to scale the input features or tuning of any parameters.

3.2 Random Forest Model

Random Forest is another machine learning algorithm for performing classification tasks. In this step, a Random Forest model with default parameters is implemented on the training set. The default number of trees in this case is 10.

3.2.1 Model Implementation

Random Forest is an extension of bagging that in addition to building decision trees based on multiple samples in the training data. The split points are chosen by finding the attribute and the value of that attribute that results in the lowest cost. An implementation of *RandomForestClassifier* function is used to implement the random forest model. Like logistic regression model, the *RandomForestClassifier* is fed with the sparse matrix of the tokens. This algorithm uses *Gini index* as the cost function to evaluate the split points. This model is trained on the training dataset and its performance is evaluated on the test dataset.

3.2.2 Model Assumptions and Advantages

Random Forest is a powerful algorithm in Machine Learning. It is based on the Ensemble Learning technique. Following are the advantages and disadvantages of the Random Forest model:

1. Random Forest is suitable for both categorical and continuous variables.
2. It takes care of the missing values automatically.
3. It is robust to outliers.
4. It can efficiently handle non-linear parameters.

Disadvantages of this algorithm are:

1. This algorithm requires high computational power and resources.
2. Longer training time.

3.3 SVM Model

Support Vector Machine (SVM) is a linear model for classification which creates a hyperplane in an n-dimensional space to separate the data for each class. This model is trained on the training dataset to create the linear hyperplane boundaries for each class in the entire corpus of the data set.

3.3.1 Model Implementation

SVM is an easy to implement classifier and is almost as memory efficient as logistic regression. First, we trained the SVM on the labelled training data and then it is used to predict the labels for the testing data. The performance of this model is evaluated by evaluating the accuracy of the model on the true labels of the test dataset. The kernel type used in this model is *linear* assuming the data is linearly separable.

3.3.2 Model Assumptions and Advantages

The biggest assumption of SVM is that the data is linearly separable. In real life, this may not be the case. However, SVM is great at finding a linear boundary in an n-dimensional space when 'n' is large. SVM assumes the data to be two-class classification problem. So, we need to provide a one versus rest approach for each class for a multi-class SVM.

The advantages of SVM are:

1. It is a special case of optimized perceptron.
2. It is easy to implement.
3. Highly effective for high dimensional data.
4. Can perform nonlinear classification by mapping vectors into even higher dimensions.

The disadvantages of SVM are:

1. Not great for large data sets.
2. Solves K binary classification problems for K class classification.

4. Model Comparison and Performance Evaluation

As part of this project there were quite several different experiments which were tried and tested for their accuracy. The metrics used to compare the performance of these models are **accuracy** and **recall** score. Accuracy in this classification problem is the number of domain names that are predicted by the model correctly over all the predictions.

An important consideration in building an appropriate ML model for predicting the DGA generated domain names is minimising the number of *False Negatives*. In this scenario, *False Negatives* are the number of domain names that are incorrectly predicted as *legitimate* or *dga* by the model.

Recall tells us what proportion of domain names that were generated from DGA were detected correctly by the algorithm. Therefore, the *recall* of a desirable model will be as close to 100% as possible.

Model	Accuracy	Recall
Logistic Regression	0.916	0.909
Random Forest	0.911	0.896
SVM	0.919	0.912

It is observed that SVM model yields the best value of accuracy and recall score. However, the difference between the performance metrics of SVM and Logistic Regression Model is very minute. SVM solves the *quadratic programming problem* (QP) to find a separation hyperplane which makes it computationally very intensive. Even though this model yields the best results; this model is not the most suitable when the dataset has large number of samples. Hence, logistic regression is a more suitable model for this classification problem.

5. Conclusion

From the above results and the overall attempt at identification of the domain names generated from Domain Generating Algorithms (DGAs), it is learned that the traditional machine learning methods all perform in almost the same range. Though SVM is a suitable classifier for such type of problems with high dimensional vector spaces, it is computationally quite expensive over a large sample of dataset and logistic regression offers an easy to implement and highly generalizable approach.

This is an active area of research and newer ways to optimize the representations of tokens like the *word2vec* transformations using a deep neural network offers solutions for faster computations instead of the count vector vectorization.

The key take-away from this project is that the logistic regression is a great generaliser and a very good classifier as it requires very less computational resources.