Domain Generation Algorithms detection through deep neural network and ensemble

Shuaiji Li
DiDi Research America
Mountain View, CA, USA
sl6486@nyu.edu

Tao Huang
DiDi Research America
Mountain View, CA, USA
taohuang@didichuxing.com

Zhiwei Qin
DiDi Research America
Mountain View, CA, USA
qinzhiwei@didichuxing.com

Fanfang Zhang
DiDi Research America
Mountain View, CA, USA
zhangfangfangff@didichuxing.com

Yinhong Chang
DiDi Research America
Mountain View, CA, USA
yinhongchang@didichuxing.com

ABSTRACT

Digital threats such as backdoors, trojans, info-stealers and bots can be especially damaging nowadays as they actively steal information or allow remote control for nefarious purposes. A common attribute amongst such malware is the need for network communication and many of them use *domain generation algorithms* (DGAs) to pseudo-randomly generate numerous domains to communicate with each other to avoid being take-down by blacklisting method. DGAs are constantly evolving and these generated domains are mixed with benign queries in network communication traffic each day, which raises a high demand for an efficient real-time DGA classifier on domains in DNS log. Previous works either rely on group contextual/statistical features or extra host-based information and thus need long time window, or depend on lexical features extracted from domain strings to build real-time classifiers, or directly build an end-to-end deep neural network to make prediction from domain strings. Pros and cons exist for either way in experiments. In this paper, we propose several new real-time detection models and frameworks which utilize meta-data generated from domains and combine the advantages of a deep neural network model and a lexical features based model using the ensemble technique. Our proposed model obtains performance higher than all state-of-art methods so far to the best knowledge of the authors, with both precision and recall at 99.8% on a widely used public dataset.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**; • **Computing methodologies → Neural networks**; **Ensemble methods**; *Supervised learning by classification.*

KEYWORDS

URL classification; DGA; Deep neural network; Recurrent neural network; Ensemble; Lexical features

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308558.3316498

ACM Reference Format:
Shuaiji Li, Tao Huang, Zhiwei Qin, Fanfang Zhang, and Yinhong Chang. 2019. Domain Generation Algorithms detection through deep neural network and ensemble. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW'19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3308558.3316498

1 INTRODUCTION AND REVIEW

With the rapid advancement in technology and the proliferation of digital devices in today's world, the amount of data being generated, shared and stored is more than ever before. Whilst this brings increased convenience in the form of, for example, online shopping and banking services, cybercriminals are also aware of the value of such data. Digital threats such as backdoors, trojans, info-stealers and bots are constantly evolving and can be especially damaging as they actively steal personal information or allow malicious attacks. One common attribute amongst such malware is the need for network communication. Such malware typically communicate with an attacker controlled *Command and Control* (C&C) server via C&C channels. In order for the malware to communicate with its C&C server, it has to know the server's IP address. One straightforward way is hard-coding a list of IP addresses or domain names in the malware. However, this renders the malware vulnerable to simple IP or domain *blacklisting*. In order to avoid detection and prevent being taken down, some malware families adopted *domain generation algorithms* (DGAs). DGAs are used to generate a large number of pseudo-random domains from a specified seed. The malware and C&C servers share the same seed and pseudo-random algorithm. For example, a DGA could use a date as its seed, as discovered by researchers who reverse-engineered the Torpig malware [27], or even a character of a tweet, as they later found in an upgraded version of Torpig. Malware employing DGAs can generate a set of new domains from each seed and attempt to connect to these domains. As the malware will continually try resolving and connecting to each domain, only a small subset of these domains need to be pre-registered by an attacker for this whole scheme to work. Because any seed value and algorithm can be used in DGA schemes, it is difficult to predict the domains that each malware will use. Therefore, this method effectively renders IP/domain blacklisting useless.

The domains being queried in a network can be acquired via logs of *Domain Name Service* (DNS) servers or a monitoring server. One can build a system to analyze each queried domain and give a

verdict, either malicious or benign, that guides further actions such as communication block or remediation. Due to the high volume of DNS traffic, and the target to block malware's C&C channels as early as possible, a *real-time* DGA classifier becomes important in such case, as blacklisting becomes inefficient, and reverse engineering [27, 28] [1] of the DGA might only work usefully for a while as attackers would update the seeds and algorithms. To build such classifiers, many features can be extracted, such as lexical features, contextual features, or host-based features [25]. Mixed feature type models exist in previous works [3, 32, 33]. However, as to a real-time system in DGA classification or general malicious URL detection, the most straightforward and commonly used features are still lexical features that describe the URL itself [1, 2, 5, 22, 24].

Artificial neural networks (ANNs) are a family of machine learning models that prove to achieve high performance on many difficult problems [12, 17, 21]. ANNs can be used to learn the patterns in the malicious URL strings and lexical features extracted. For example, Le et al. [23] used embedding to transform URLs to matrices and use a deep convolutional neural network (CNN) to classify malicious URLs. In a wide range of tasks involving time series and sequential data, *Recurrent Neural Networks* (RNNs) prove to be useful [15, 29]. RNNs are a class of neural networks with loops (so called *recurrences*) that recursively combine the input vectors with the network state vectors to produce a new state vector, thus allowing temporal information to persist throughout the learning process, and therefore successfully learn some patterns in sequential data. In particular, *long-short term memory* (LSTM) [14, 18] is a special kind of RNN frequently used for data with long-term temporal dependencies. It solves the *vanishing gradient* problem that a vanilla RNN might have [2] by employing cells that preserve information and use 3 "gates" (input, output and forget) to control the information flow. A trained RNN-based model like LSTM is fast in inferencing and prediction, and hence naturally suitable in real-time malicious URL or DGA classification. Bahnsen et al. [4] used LSTM networks directly to the URL string sequence to predict if the URL is a Phishing URL. In DGA classfication, since URLs are randomly generated, RNN models can be developed to learn and detect the "randomness" patterns. Woodbridge et al. [31] used a 128-units single layer LSTM (with input embedding layer) model to learn the "randomness" of different generation algorithms (such as *ramnit, shifu, suppobox, beecone, simba* etc.) and make binary predictions (DGA or non-DGA) or multi-class prediction (the specific algorithm prediction). They reached 98% true positive rate (TPR) and 0.1% false positive rate (FPR) on a public database of DGA from Bambenek Consulting [10]. Within an industry setting, assuming 1 million URLs are queried in the network on a daily basis, and 0.1% of them are real malicious DGAs. A 0.1% FPR will produce roughly 1,000 false positive cases, which might require further inspections from domain experts. We aim to reduce this workload, and at the same time, maintain the generalizability of the algorithm.

The main drawback of these two models is that they solely rely on an LSTM on the string itself, and neglect a wide range of lexical features that can be extracted and assist the classification, like n-grams, length, special characters, etc, as in [5, 13, 20, 22, 24, 26, 32, 34]. Potential failures caused by such drawback lie in two folds. First, sole URL string based deep learning model might "over-learn" the negatives and positives in training set, and generalize not so well in reality [3]. Furthermore, low interpretability of neural networks make it hard to improve or amend such models. Second, some lexical features, like n-grams, special characters, domain length, etc. might convey patterns that can really help the classification, yet these models did not embed them explicitly. RNN model might finally learn these helpful features in an implicit way with abundant data, but combining these features directly in the model might be more efficient. Also, combining string based RNN model and lexical features model might help to "neutralize" the flaws of each individual model and make performance improvement, as we will show in this work. In this paper, we propose a new framework that utilize RNNs and lexical features to form a mixed model to improve the DGA prediction performance. The rest of the paper is arranged as following: in Section 2, we will propose two RNN based models, probability network and representation network, an n-gram based model, and an ensemble model that combines these models and extra lexical features extracted. In Section 3, we will show our experiment settings. Experiment results and discussions will be in Section 4.

2 METHODOLOGY

Our methodology consists of two new deep neural network models and an n-gram-based Xgboost model. Each of them has its unique advantages and disadvantages, and they are finally integrated into an ensemble model that combines the best of all. We will demonstrate the prediction performance in Section 4.

2.1 RNN-based model

We propose two network architectures that are similar but motivated by two different concepts and thus learned differently — a probability network and a representation network.

2.1.1 Probability network. Intuitively, the whole deep learning model tries to capture the pattern in the URLs in order to make prediction of the probability that a given URL is DGA. A graphical depiction of our probability network model is shown in Fig 1. We combine extra lexical features like top-level-domain (TLD) and string length, two commonly used lexical features in previous works in malicious URL identification [3, 13, 20]. A TLD is one of the domains at the highest level in the hierarchical Domain Name System of the Internet. For example, in 'photos.google.com', 'com' is the TLD. String type inputs need to be tokenized first, and get truncated or padded to a uniform length. Subsequently, in the arena of machine learning, one-hot encoding and embedding are two techniques popular in transforming the tokenized string input to a reasonable numerical representation that can be fed into learning models. The one-hot encoding transforms all the character-wise tokens from the original input to sequences of binary representation. As a consequence, it can be slow and space-consuming with high redundancy. On the other hand, embedding condenses the information from the original sequence and can be optimized together with the subsequent models, but it can also lose some useful information

[1] After reverse-engineer, defenders can register anticipated domains and even hijack the C&C sever the attackers use, known as *sinkholing*

[2] the problem which make "memorizing" long-range dependencies difficult

[3] which indeed happened when we deploy such model, as false positives happen on benign domains rarely visited

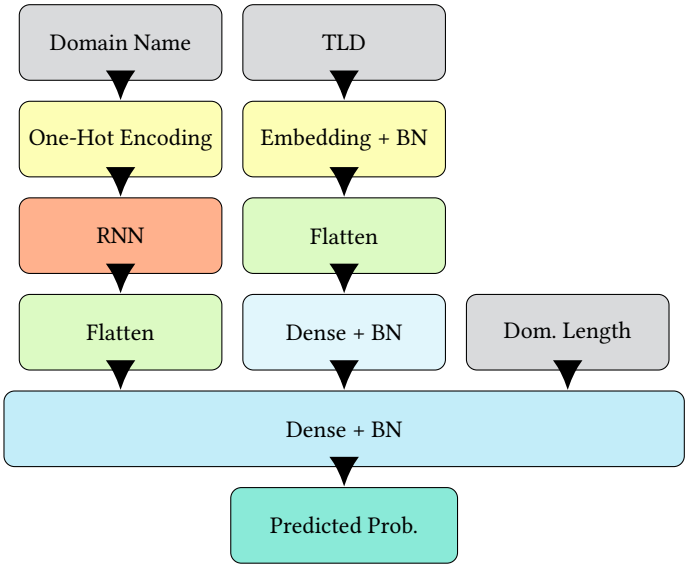

Figure 1: Proposed probability network based on RNN. "BN" here stands for the batch normalization.

from the original sequence. In our probability network, we feed in the full information from the domain name by using one-hot encoding. Meanwhile, we use an embedding layer to the TLD input, as a TLD can only take its value in a predefined list with thousands of legal TLDs [4] and thus is relatively less complex compared to domain names.

An RNN such as LSTM is later used to learn the DGA patterns in the domain name sequence. Its outputs, concatenated with the domain length and the output from TLD processing, are fed into one or multiple fully connected layers before the probability prediction is generated from a final *sigmoid* activation.

2.1.2 Representation network. For text type data problems, such as NLP tasks, pre-training a representation network using RNN models was proved to be effective for improving performances as downstream models can be built up based on these represented vectors [11]. Similarly, when we leverage an ensemble model as shown in Fig 2, we could build an RNN model that maps the URL string (and TLD) to a low-dimensional space, where DGAs are "clustered" and separated from non-DGAs, to ease the learning of the downstream ensemble model. In fact, our *probability network* can be seen as a 1-dimensional representation of the URL strings.

The representation network is constructed similarly to the probability network we proposed, but we intended to make it light-weight, which is very important when we have enormous training data. Similar to probability network, domain names and TLDs are separated and processed, and finally concatenated before feeding into fully connected layers. To make it light-weight, first we replace the one-hot encoder of domain names with an embedding layer to accelerate the rest of domain name learning; second we choose the GRU [9] to learn embedded domain names instead of an LSTM model[5]. Domain length input is canceled in this network. The final low-dimensional output vector of this network is saved as a group of new features for our ensemble model.

[4]Internet Corporation for Assigned Names and Numbers (ICANN) manages most of the legal TLDs

2.2 n-gram based model

In addition to the deep neural network models, we build an Xgboost [8] model to learn the pattern of DGAs. Similar to the idea of [33], we first construct a bigram occurrence count table for each possible character bigram from the training data. The intuition is that generative algorithms such as DGAs are unlikely to generate domain names that preserve a bigram distribution that is exactly the same to legitimate domain names. For a benign domain, it is common to see some "regular" combinations, such as 'go' within 'Google', or 'be' within 'YouTube', either of which contains legitimate lexical items or follow general pronunciation habit in English or some other languages. A malicious DGA, in contrast, tends to disrupt the order of characters[3] so as to circumvent the patterns captured in blacklists. Hence, a DGA domain with counts on less frequent character bigrams is likely to be detected through an n-gram based model.

Once we obtain the character bigram features, we can train a gradient boosting based classifier (Xgboost in this case) to make a prediction. The reason of choosing Xgboost over conventional gradient boosting decision-tree (GBDT), or Adaboost, is that Xgboost can be trained in distributed fashion using multi-cores, which drastically speeds up the training process. In [1], trigram features as opposed to bigram features were used to train a random forest (RF) DGA classifier. In our experiment, we found that bigram features outperform unigram or trigram features in all the evaluation metrics. We also show that gradient boosting based models perform better than RF regardless the choice of n in character n-grams ($n < 4$). Although both RF and Xgboost belong to ensemble learning methods, they differ in multiple aspects. For RF, the order of trees is not important since each tree is built independently using bootstrap samples of the training data. Xgboost, on the other hand, adds new trees to already built ones by continuously minimizing the pseudo-residual over decision-tree space. According to [6, 7], as long as the hyper-parameters are carefully tuned, gradient boosting based models are more robust than RF over a regular feature space with a dimension less than 4,000 because of the above difference in the training objective. In our case, the experiment results in roughly a thousand of bigram features.

2.3 Ensemble

As explained previously, RNN-based models are good at learning temporal information in sequential data. Our RNN-based models are featureless, with syntactic and semantic representations only taken into account implicitly through the training process. Global patterns in domain names and long term correlations among characters can be captured for malicious and benign domains. However, as mentioned in [31], the global "randomness" of the generated domains is hardly captured by the RNN if a training set has inadequate instances of some generation algorithms. Meanwhile, n-gram based model is better in detecting malicious domains that share character n-gram (local patterns, as opposed to full string patterns)

[5]Compared to LSTM, GRU has no memory unit and is therefore faster to train.

distributions with known DGAs in the training set. It does not retain any semantic meaning of a sequence. Therefore, it is possible that a benign domain is classified as DGA by the n-gram model because it shares some bigram tokens with some malicious domains in the training set even though the bigram's position in the string, or leading/following bigrams are different. This yields a high false positive rate (FPR), which costs more further investigations and, of course, higher labor costs, as we found in production test.

To mitigate the influence of global and local information, and to maintain the efficacy of the model with lower false positive rate (FPR) and false negative rate (FNR), we propose a novel ensemble model that encompasses the virtue of both RNN-based and n-gram based models with some extra URL-based features. Inspired by [3–5, 26, 32, 33] etc., the extra features we make from a domain include:

- Length of name s ($s \in$ {domain name, TLD})
- Whether the length of a name s is larger than a threshold c_s ($s \in$ {domain name, TLD})
- Number of numerical characters contained in domain name, tld (also whether tld contains any numerical character)
- Number of special characters contained in domain name and whether TLD contains any special character

A graphical description of the ensemble model is shown in Fig 2. We concatenate the DGA probability predictions from RNN-based model and the probability predictions from Xgboost model described above, together with the additional features we make out from domain strings, and then feed them to a top logistic regression model for the final DGA probability prediction

$$\begin{aligned} p_i &= \frac{1}{1+\exp\left[-(\mathbf{w}^\top \mathbf{x}_i + b)\right]} \\ &= \frac{1}{1+\exp\left[-(w_1 x_{1i} + w_2 x_{2i} + \ldots + w_d x_{di} + b)\right]} \end{aligned} \tag{1}$$

where $\mathbf{w}$ is the linear weights, $\mathbf{x}_i = (x_{1i}, x_{2i}, \cdots, x_{di})^\top$ is the concatenated vector containing extra features and output from the upstream components, and b is the model bias.

At the training stage, both RNN-based model and n-gram based model are trained separately first. On the second stage, their predictions on training data are used in the top logistic regression model training to solve for the ensemble coefficients. By default the coefficients are regularized with ℓ_2 norm, and the coefficient of the regularization term is found by grid search.

3 EXPERIMENT SETUP

In this section, we describe the implementation specifics of all the models developed in Section 2.

3.1 RNN-based model

Probability network. In the probability network model, as an experiment, we implement an LSTM network which is composed of one layer of 256 units LSTM. LSTM output sequences are flattened before being sent to the fully connected dense layers. TLDs are first embedded to 5-dimensional vectors and then fed to a 32 units dense layer with rectifier activation. LSTM model output, TLD output and domain length were concatenated and then fed to a 32 units dense layer with rectifier activation. The output layer is a 1 unit layer with sigmoid activation. Batch normalization layers are added to some layers as shown in Fig 1. We maximize the likelihood of the training instances by optimizing the binary cross-entropy loss.

Representation network. In the GRU based representation model, domain names are first embedded to 63-dimensional vectors and fed into an RNN that is composed of one layer of 256 GRU hidden units. TLDs are embedded to 7-dimensional vectors and fed to a 32 units fully connected layer. GRU output matrices are then flattened and concatenated with the dense representations of TLDs and fed together into a 64 units dense layer. Two subsequent dense layers, with 32 and 1 dense units each, are employed to transform joint dense representations into probability measures. Rectifier activation is used on all intermediate dense layers, and the last layer is activated through a sigmoid function. Once the representation network is trained, output vectors of the second to last layer (32-dimensional) is exported for the ensemble model.

3.2 n-gram based model

For n-gram based models, we test the performance of RF and Xgboost classifiers on unigram, bigram, and trigram features of domain names, respectively. Character n-gram features are extracted on training set, with random holdout test set to measure the generalizability to detect DGA from benign URLs. To reduce potential over-fitting, for each classifier we grid search the hyper-parameters including number of trees and maximum depth of each tree. We choose logistic loss as the objective function of Xgboost and fix the learning rate to 0.1. Information gain entropy is used as the node splitting criterion for random forest.

3.3 Ensemble layer

Once the RNN-based model and n-gram based model are thoroughly trained, we build a top-level logistic regression model taking input from RNN-based model output and n-gram based model output, together with some URL-based features, as shown in Fig 2. The top-level feature space contains either 10 features.The threshold $c_s = \{45$ (domain names), 7 (TLDs)$\}$, and special characters include multiple symbols such as underscore and period. The layer weights are regularized with ℓ_2 norm and the penalty parameter is grid-searched through cross-validation. From now on, we call the generated features, corresponding to the definition in Section 2.3, as `domain_len, tld_len, is_dom_ge_45, is_tld_ge_7, #_num_char, #_num_char_in_tld, is_num_in_tld, #_spec_char_in_tld, is_spec_in_tld` for simplicity.

4 RESULTS

4.1 Overall performance

Table 1 concludes the test performance of all the models we tested in this paper. Previous popular used SVM-based models presented in [16, 30] did not even generalize well (precision and recall lower than 97%) on smaller test sets (less than 2,000 malicious DGAs), and were therefore excluded from our baseline experiments. For the LSTM model proposed in [31], we used the exactly same setting and reached a similar result as declared in the paper. The dataset we used was composed of benign URLs (from Alexa top 1M domains) and DGAs (from Bambenek Consulting[10], about 942k). 70% of

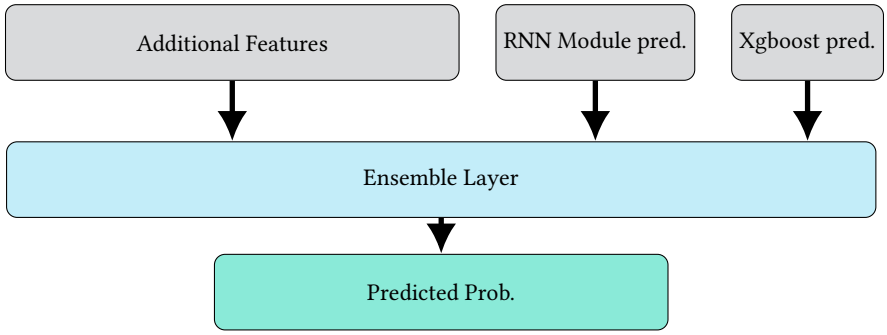

Figure 2: Ensemble model

Table 1: Experiment Results

		Evaluation Metrics				
	Model	Precision	Recall	F1-score	Accuracy	AUC
RNN based	LSTM proposed in [31]	0.9826	0.9819	0.9822	0.9822	0.9970
	Probability Network	0.9940	0.9930	0.9935	0.9937	0.9995
N-gram based	RF (unigram)	0.9637	0.9624	0.9628	0.9629	0.9919
	RF (bigram)	0.9737	0.9772	0.9727	0.9728	0.9965
	RF (trigram)	0.9482	0.9424	0.9434	0.9438	0.9928
	Xgboost (unigram)	0.9669	0.9661	0.9664	0.9665	0.9934
	Xgboost (bigram)	0.9854	0.9850	0.9852	0.9852	0.9986
	Xgboost (trigram)	0.9618	0.9611	0.9614	0.9614	0.9914
Ensemble	Ensemble-probability	**0.9982**	**0.9982**	**0.9982**	**0.9982**	**0.9999**
Ensemble	Ensemble-represent.	0.9967	0.9967	0.9967	0.9967	0.9998

the dataset was used as the training set, and the rest used as the testing set. Our probability network outperforms the LSTM network proposed in [31], with 99.40% precision and 99.30% recall. Among all the n-gram based models, our Xgboost classifier outperforms the RF classifiers proposed by Ahluwalia et al. [1] on all three cases ($n = 1, 2, 3$), with $n = 2$ the best configuration. Overall, the n-gram, tree-based models underperform the probability network we proposed. The ensemble-probability model outperforms the rest models, and show significant improvements over methods in studied literature. The ensemble-probability model achieves 99.82% precision and recall. To the authors' best knowledge, our model obtains the best performance on DGA classification.

Besides, based on Table 1, the FPR of our ensemble-probability model is 0.17%, almost ten times smaller than the LSTM network(1.60%) proposed in [31] [6]. The level of potential human involvement (i.e. domain expert inspections) is effectively lessened. We have also tested the ensemble model on various test sets collected from different time periods (March - May 2018), and it achieved similar performance, a sign of well-preserved generalization ability.

4.2 Case studies

[6]Note here the FPR we get is different from what [31] claimed in their paper; we cannot reproduce the 0.1% FPR in our experiment using our dataset and the code snippet provided in [31]

4.2.1 LSTM-based probability networks. LSTM takes its input from the domain character sequence, and returns an output vector, or a sequence of cell output vectors if we choose to. We wonder: 1) how the prediction is formed as the characters were fed into the LSTM sequentially, and 2) in which point the model raises or decreases its prediction. For the first question, Karpathy et al. [19] visualized LSTM cell outputs in heatmaps and found units (or dimensions, channels) of cell states that have interpretable use in prediction. We could do so as well to explore our probability network, but it turns out there seems no specific human interpretable meaning of any unit that can apply to all domains. For the second question, we can expand subsequences of a domain character-by-character from the first character to the whole string, feed these subsequences to the LSTM based model and observe how the predicted probability is changing. This experiment was conducted on our probability network and the LSTM in [31] respectively on several selected domains and Fig 3 shows the result. The probability network seems to be able find some patterns that are meaningful in human language and thus clear the DGA suspicion of benign domains while the LSTM in [31] fails to do so for these cases. For example, in 'jpopsuki.eu' case, the probability network predicts 'jpopsuk.eu' as DGA, but when the last 'i' completes a meaningful subsequence 'suki'[7] the probability network drops its probability prediction to 'benign'. Similar phenomena are observed when 'n' completes 'xueyuan', 'ng' completes 'xing'[8] and 'o' completes 'pro'. The exact opposite things happen for DGAs as well — when 'n' makes 'nation' and 'ter' makes 'master', the probability network identifies the word and changes its prediction to DGA, probably because in training set there are similar DGAs composed by these words.

However, in next discussion, we will show that for patterns not in the training set, like some Chinese Pinyin sequence, the probability network might make mistakes and in such cases ensemble with lexical feature models can help to make the correct predictions.

4.2.2 Probability Network vs. Ensemble. As shown in Table 2, the weights w_1 (corresponds to the output probability from the probability network) and w_2 (corresponds to the output probability of the n-gram model) are 7.3741 and 9.0226 respectively. The n-gram Xgboost model exhibits a relatively larger impact than the probability network on final predictions of the ensemble layer, even though the

[7]Here 'suki' means 'like/love' in Japanese
[8]Both 'xueyuan' and 'xing' are meaningful in Chinese Pinyin

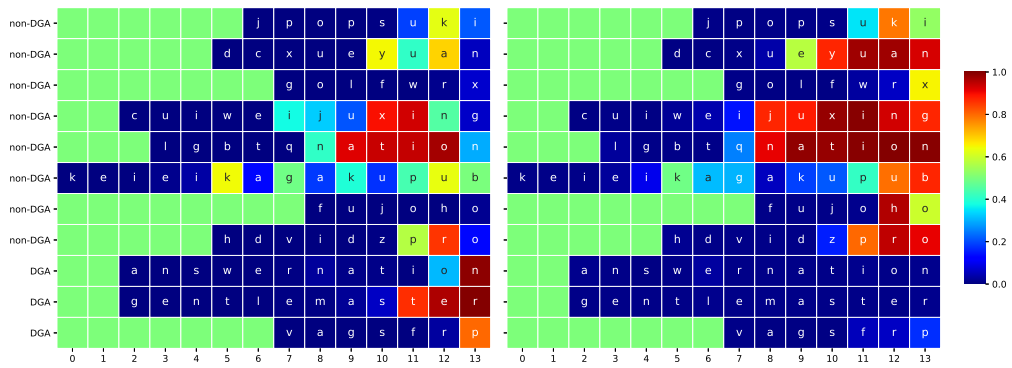

Figure 3: Comparison of LSTM based models on selective domains. The y-axis shows the true label of each domain. The left heatmap shows the prediction of our probability network when the domain is feed in as subsequences expanding character-by-character from the first character to the entire string, and the right graph shows the prediction of the model in [31]

Table 2: Weights of the ensemble layer in the ensemble-probability model

	Prob. Network	N-gram	domain_len	tld_len
Weight	7.3741 is_dom_ge_45	9.0226 is_tld_ge_7	0.0343 #_num_char_in_tld	-0.1086 #_num_char
Weight	-0.00085 is_num_in_tld	0.0037 is_spec_in_tld	-0.0043 #_spec_char_in_tld	-0.1214 b
Weight	-0.0036	-0.9690	-0.9648	-7.4072

probability network outperforms the n-gram model when tested separately.

In the test set, we examined selective URLs, on which the probability network and the n-gram based Xgboost model of the ensemble-probability model predicted differently. Contribution of each component to the final prediction is shown in Table 3. The first two columns show the probability network and n-gram model output probabilities respectively. The third and fourth columns of Table 3 show the component values of these two models (i.e., w_1x_{1i} and w_2x_{2i} respectively), while the fifth column is the sum $w_1x_{1i} + w_2x_{2i}$. The last column shows whether our prediction is correct or wrong. Clearly, a larger $w_1x_{1i} + w_2x_{2i}$ component value means a higher final predicted probability produced by the ensemble model, indicating that the two probability features dominate the top layer's feature space in terms of feature importance.

Diving into the sample cases, we saw that RNN model could correctly identify some benign URLs composed of letters that look "random" and bear no apparent semantic meanings, such as abbreviations from different languages mixed, like 'ccpitnmg' in the sample cases. These domain names look like real DGAs and contain "uncommon" n-grams within the benign cases of the training set from a character perspective. Oppositely, when DGA looks "meaningful" with "common" n-grams that can be found in English (or directly a combination of meaningful words), like 'meskia', 'gentleshould' or 'basketmillion', N-gram model might fail easily. RNN model can, in this case, learns the hidden representation of the domains as a whole through LSTM. N-gram model fails to detect this underlying distribution since bi-gram features are disjoint — the encoded token values are primarily used for nodes split, with weak correlations among n-grams through learning.

However, we noticed that RNN model could have trouble in decoding some sequence patterns, for example, some of the Chinese Pinyin based domains, such as 'xizanglvyou', 'jiafalvshi' and 'ixueyi' [9]. It tends to output a high probability, which make us believe that the LSTM does not generalize its good performance on other similar semantic based domains (like 'gentleshould') to these domains. A probable reason is that our training set of benign URLs (Alex top 1M) contains inadequate examples for Chinese Pinyin patterns. At each cell of the LSTM, the emitted hidden vector contains cell values of current input character with information of previous characters[14, 18]. To always obtain a representative vector that captures the underlying sequence pattern over hidden units, LSTM layers should be trained on a data set that contains enough input sequences with diversified distributions. The authors in [31] have also claimed that lack of training instances generated by some rarely used generation algorithms lead to pool generalization of its LSTM-based model in real production system. For these cases, our N-gram based model makes the right predictions based on their character bi-gram counts.

These cases show the improved robustness of our ensemble model, as compared to individual RNN based and n-gram based model. On some patterns and cases, the two component models can be complementary. Moreover, under this framework, we can easily explore more (complementary) component models and add them to the ensemble.

4.3 Representation of DGAs

Fig 4 shows a 2-dimensional visualization of representation network output for selected test data using t-SNE. Overall, the representation network successfully created the low-dimensional representations of the test domains that are easy to separate and classify. Most benign and malicious cases clearly form its own cluster except for some cases in the lower portion of the graph. This explains why

[9] literally translate: 'Tibet tourism', 'family law lawyer' and 'I learn skill'. The 'v' in 'lv' stands for 'ü' when typewriting Pinyin using standard keyboard

Table 3: Selected ensemble-probability model's prediction components of sample URLs

Domain	RNN prob.	N-gram prob.	RNN comp.value	N-gram comp.value	RNN+N-gram comp.value	Ensemble prob.	True label	Pred.*
xizanglvyou.org	0.9998	0.0134	7.3725	0.1205	7.4930	0.5343	0	✗
jiafalvshi.com	0.8061	0.0952	5.9445	0.8590	6.8036	0.3575	0	✓
ixueyi.com	0.7309	0.0914	5.3897	0.8248	6.2145	0.2121	0	✓
ccpitnmg.org	0.2480	0.6315	1.8291	5.6973	7.5264	0.5170	0	✗
meskia.biz	0.9823	0.0047	7.2434	0.0425	7.2859	0.4400	1	✗
ngxdlr.org	0.2158	0.9670	1.5916	8.7247	10.3164	0.9421	1	✓
ysmyhuh.net	0.9659	0.1778	7.1224	1.6044	8.7268	0.7745	1	✓
gentleshould.net	0.9982	0.0124	7.3612	0.1117	7.4729	0.5378	1	✓
basketmillion.net	0.9948	0.1011	7.3358	0.9123	8.2481	0.7234	1	✓

*: ✓ means our prediction is correct and ✗ means our prediction is wrong.

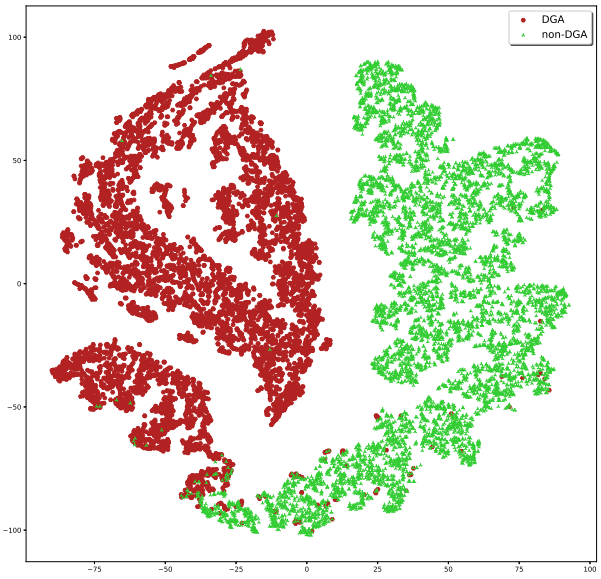

Figure 4: A t-SNE embedding over representation vectors output from GRU based *representation network*. For each group (DGA vs. non-DGA), we randomly selected 8,000 domains from the test data of the group.

ensemble-representation model obtains a very promising performance close to ensemble-probability. Bear in mind that the representation network is initially designed as a much lighter-weight model compared to the probability network, and indeed, the running time in our experiments of the representation network training is only 1/10 of the running time of the probability network training using same machine. For situations where computation power or time frame is limited, we suggest using ensemble-representation model for better efficiency-performance trade-off. Besides, this representation network can be easily plugged-in other learning models and frameworks as part of input processor.

Table 4: Selected edge cases where extra features are influential on final prediction

domain	RNN prob	N-gram prob	RNN+ N-gram comp_val	All comp _val*	Ensem. prob	True Label
eyeota.net	0.9387	0.0614	7.4761	7.3562	0.4872	0
vorobatorakhtopl .com	0.0014	0.8136	7.3511	7.5742	0.5417	1
houmemapoig gofjogra.com	0.0000	0.8146	7.3502	7.6762	0.5669	1

*: As the intercept is -7.4072 from Table 2, if an All comp_val is larger than 7.4072, the model will give a positive prediction and vice versa.

4.4 Extra Features

From Table 2, most extra features have relatively small coefficients after training, and indeed for most of test cases, the predicted label is mostly determined by RNN-based model output(s) and n-gram based model output. However, in some "edge" cases where RNN-based model and n-gram based model disagree and almost "neutralize" each other, the extra lexical features take effect. Table 4 shows 3 cases where such disagreement happens and the extra features help make the correct predictions (mainly length related extra features are making difference in these cases).

5 CONCLUSION

In this paper, we built RNN-based deep neural network models for DGA prediction. Together with n-gram based machine learning model for DGA classification and extra lexical features we made, we proposed to build a top ensemble layer that combines the advantages of the n-gram based model, assistive lexical features and the full-string based model (here the deep neural network models) while at the same time alleviating the disadvantage of each individual component when it is used separately to a great extent. As a result, first, the LSTM based probability network outperforms the literature [31] LSTM model by almost 1 percent in both precision and recall; second, without loss of the generalization capability, the

ensemble model obtains a new state-of-the-art result of 99.82% precision and recall, higher than the standalone n-gram based model and RNN-based probability network model we proposed.

In the discussion of the result, we show that either the full-string based deep learning model or the n-gram based model might not be able to capture all DGAs, as DGA is versatile and the task is complex. We suggest using an ensemble model to balance the two types of models. Our work shows the virtue of each part and proposes a viewpoint of combining them together in a good way. We believe this thought is more important than the specific ensemble implementation (i.e. logistic regression) we propose here, as it is just one way of implementing the thought, and we expect more future works keep revealing the pros and cons of different types of models in DGA prediction and the approaches of composing them to a better meta-model. Our representation network proves to be effective as well as it separates DGAs and non-DGAs in representation space. Meanwhile, DGA is also keep evolving, for example, we find DGAs based on random word sequences instead of character sequences. This brings need for future works on word/character mixed DGA prediction models.

REFERENCES

[1] Aashna Ahluwalia, Issa Traore, Karim Ganame, and Nainesh Agarwal. 2017. Detecting Broad Length Algorithmically Generated Domains. In *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*. Springer International Publishing, 19–34.

[2] Iraj Sadegh Amiri, Oluwatobi Ayodeji Akanbi, and Elahe Fazeldehkordi. 2014. *A machine-Learning approach to phishing detection and defense.* Syngress.

[3] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware.. In *USENIX security symposium*, Vol. 12.

[4] Alejandro Correa Bahnsen, Eduardo Contreras Bohorquez, Sergio Villegas, Javier Vargas, and Fabio A González. 2017. Classifying phishing URLs using recurrent neural networks. In *Electronic Crime Research (eCrime), 2017 APWG Symposium on*. IEEE, 1–8.

[5] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. 2010. Lexical feature based phishing URL detection using online learning. In *Proceedings of the 3rd ACM workshop on Artificial intelligence and security*. ACM, 54–60.

[6] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. 2008. An empirical evaluation of supervised learning in high dimensions. In *In International Conference on Machine Learning (ICML*. 96–103.

[7] Rich Caruana and Alexandru Niculescu-Mizil. 2005. An Empirical Comparison of Supervised Learning Algorithms Using Different Performance Metrics. In *In Proc. 23 rd Intl. Conf. Machine learning (ICML'06)*. 161–168.

[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 785–794.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[10] DGA Domain Feed Bambenek Consulting. 2018. https://osint.bambenekconsulting.com/feeds/dga-feed.txt. Accessed: 2018-03-06.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[12] Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 69–78.

[13] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. 2007. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*. ACM, 1–8.

[14] Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* 12, 10 (Oct. 2000), 2451–2471. https://doi.org/10.1162/089976600300015015

[15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 38 (March 2013), 6645–6649.

[16] F. Haddadi and A. N. Zincir-Heywood. 2013. Analyzing string format-based classifiers for botnet detection: GP and SVM. In *2013 IEEE Congress on Evolutionary Computation*. 2626–2633. https://doi.org/10.1109/CEC.2013.6557886

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[19] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[20] Pranam Kolari, Tim Finin, and Anupam Joshi. 2006. SVMs for the Blogosphere: Blog Identification and Splog Detection. In *AAAI Spring Symposium on Computational Approaches to Analysing Weblogs*. Computer Science and Electrical Engineering, University of Maryland, Baltimore County. Also available as technical report TR-CS-05-13.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[22] Anh Le, Athina Markopoulou, and Michalis Faloutsos. 2010. PhishDef: URL Names Say It All. *CoRR* abs/1009.2275 (2010). arXiv:1009.2275 http://arxiv.org/abs/1009.2275

[23] Hung Le, Quang Pham, Doyen Sahoo, and Steven CH Hoi. 2018. URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection. *arXiv preprint arXiv:1802.03162* (2018).

[24] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2009. Identifying Suspicious URLs: An Application of Large-scale Online Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, New York, NY, USA, 681–688. https://doi.org/10.1145/1553374.1553462

[25] Doyen Sahoo, Chenghao Liu, and Steven CH Hoi. 2017. Malicious URL detection using machine learning: A survey. *arXiv preprint arXiv:1701.07179* (2017).

[26] Stefano Schiavoni, Federico Maggi, Lorenzo Cavallaro, and Stefano Zanero. 2014. Phoenix: DGA-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 192–211.

[27] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. 2009. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*. ACM, New York, NY, USA, 635–647. https://doi.org/10.1145/1653662.1653738

[28] Brett Stone-Gross, Marco Cova, Bob Gilbert, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. 2011. Analysis of a botnet takeover. *IEEE Security & Privacy* 9, 1 (2011), 64–72.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*. MIT Press, Cambridge, MA, USA, 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173

[30] Zhen Wang, Zhongtian Jia, and Bo Zhang. 2018. A Detection Scheme for DGA Domain Names Based on SVM. (2018). https://doi.org/10.2991/mmsa-18.2018.58.

[31] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. 2016. Predicting domain generation algorithms with long short-term memory networks. *arXiv preprint arXiv:1611.00791* (2016).

[32] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Reddy, and Supranamaya Ranjan. 2010. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 48–61.

[33] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Narasimha Reddy, and Supranamaya Ranjan. 2012. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. *IEEE/Acm Transactions on Networking* 20, 5 (2012), 1663–1677.

[34] Wen Zhang, Yu-Xin Ding, Yan Tang, and Bin Zhao. 2011. Malicious web page detection based on on-line learning algorithm. In *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, Vol. 4. IEEE, 1914–1919.