Word encoding for word-looking DGA-based Botnet classification

Sea Ran Cleon Liew* and Ngai Fong Law
* University of Waterloo, Canada
E-mail: srcliew@uwaterloo.ca
The Hong Kong Polytechnic University, Hong Kong
E-mail: ennflaw@polyu.edu.hk

Abstract — **There are two main types of domain name-generating algorithms (DGAs) – random-looking and word-looking. While existing methods can effectively distinguish between the two types of DGAs with high accuracy, classifying different types of word-looking DGAs has proven to be challenging, as they are often mistaken for legitimate domains. To address this issue, previous methods used character encoding with long short-term memory networks (LSTM) or convolutional neural networks (CNN) to model the character distribution of different word-looking DGAs. Since most word-looking DGAs are constructed using various dictionaries, we propose using word encoding instead of character encoding. Word encoding can provide a better characterization as it is based on the usage of different words in the dictionaries and their associations. Experimental results show that the classification accuracy for word-based DGAs increases by more than 7% (from 87% to 94%) using word encoding as compared to character encoding.**

I. INTRODUCTION

Domain name generation algorithms (DGAs) are techniques to generate a large number of domain names to facilitate the establishment of covert communication between bots and the command and control (C&C) server. By constantly changing domain names in communication, it makes the tracking and identification of malicious network traffic difficult. In this way, DGAs help to prevent the takedown of the C&C servers [1].

There are two main types of DGA: random-looking and word-looking [2-3]. Random-looking DGAs are produced by concatenating random strings of characters. In contrast, the word-looking DGAs are generated from concatenating words taken from a dictionary. Various methods have been proposed for detecting and classifying DGAs. Detection means to determine if the domain names are legitimate or generated by the DGA-based botnets, while classification identifies which method has been used to generate the DGA. Hence detection is a binary problem while classification is a multi-class problem. As reported in [4], the accuracy of the binary detection and multi-class classifications are over 90% and around 70% respectively using machine learning-based (ML) algorithms. Thus, it is much more difficult to classify the DGAs than to detect these domains from legitimate domains [5-7].

Various methods, including ML and deep learning-based (DL) approaches, have been proposed for detecting and classifying these domain names. In ML approaches, features about character distributions in the domain names have been used. Examples include vowels and consonant ratios or the numerals and English characters ratios [8]. Most of the extracted features were primarily targeted at random-looking domains and are effective in characterizing the randomness in the domain names. It, however, resulted in inferior performance in characterizing word-looking domain names [6]. In contrast, existing DL methods explore character encoding to characterize the character distribution and association within the domain names. Thus, DL methods are more effective in characterizing word-looking domain names than ML methods. Despite that, word-looking DGAs are produced by using different dictionaries. Instead of using character encoding, word-level encoding can better characterize the nature of the words used in the dictionary and their associations. We thus explore the use of word encoding and DL techniques for word-looking DGA classification.

This paper is organized as follows. First, a comparative study of existing methods is given in Section II. Then, our proposed method is described in Section III. Experimental results are given in Section IV. Finally, we conclude our work in Section V.

II. STUDY OF EXISTING METHODS

Domain name generation algorithms (DGAs) detection and classification are important problems in network security, as

these algorithms generate a large number of domain names to facilitate covert communication between bots and command and control servers. Table 1 illustrates the differences between the DGA detection and the classification problems. In detection, we only target to distinguish the legitimate and the DGAs. However, in classification, we want to know which algorithm has been used to generate the DGAs too.

A way to detect DGAs is to check if the domain names can be resolved into IP addresses successfully. This is based on the fact that only a small number of the generated domain names are registered. If it cannot be resolved, an error signal called NXDomain response will be received. Thus, one approach to DGA detection is to analyze network traffic characteristics and detect the error signal returned through network communication. However, the network traffic analysis can only detect DGAs, but not classify them. Classifying DGAs requires analyzing the patterns of the algorithmically generated domain names. To address this issue, both ML and DL methods have been used for DGA classification [5-12].

Feature extraction is a crucial step in ML methods. Features representing expert knowledge are extracted from the domain names to characterize the nature of domain names. Common types of features include statistical features, information theory features, and lexicographic features. For example, features like vowel-consonant ratio, TF-IDF features, n-gram distributions, the longest consecutive consonant/number/vowel sequences, pronounceability score, and entropy [9, 13-14] have been used to characterize the nature of the algorithmically generated domain names. While these features are somewhat effective in characterizing random-looking DGAs, accurately classifying word-looking DGAs remains a significant challenge. As reported in [4], the average F1 scores for classifying 11 word-looking DGAs are 0.54, 0.57, and 0.68 for kNN, decision tree, and random forest respectively. This suggests that the extracted features are not sufficient for characterizing different DGA classes. This is consistent with findings from other authors [6]. One main reason for this is that words in word-looking DGAs of different dictionaries are indeed valid English words. Thus, extracted features across different dictionaries would likely be similar.

DL has also been used for DGA classification. Unlike ML approaches, DL does not require feature extraction. Instead, the domain name is treated as a string of characters. With the use of a sufficient number of examples, a learning model is trained to distinguish and characterize the DGAs. As domain names are made up of characters, character encoding is commonly used to convert the domain names to numerical sequences. This process involves breaking down the domain name into individual characters and representing these characters as numerical values.

Convolutional neural networks (CNN) and long short-term memory networks (LSTM) are popular DL models [15-17]. LSTM is often used for acquiring patterns in long sequences [7, 10-11, 18]. CNN, on the other hand, uses a filter kernel with varied sizes to characterize sequential relationships [19-20]. Previous studies have shown that combining CNN and LSTM can improve detection and classification performance [6, 21]. According to a study in [5], the average F1 scores for classifying eleven word-looking DGAs are 0.87 and 0.90 for character-based Bi-LSTM and CNN-Bi-LSTM respectively. Comparing the performance of ML and DL approaches, we can see that the DL approach has shown a significant improvement in accurately classifying word-looking DGAs. It is promising to investigate whether the deep learning techniques can be improved further to classify the word-looking DGAs.

Domain name	Detection (Binary)		Classification (multi-class)	
	Class	Label	Class	Label
Polyu	Legit	0	legit	0
Pointreply	DGA	1	suppobox2	1
Duringsuppose	DGA	1	pizd	2
Somewhatlemon	DGA	1	nymaim	3

Table 1: DGA detection and classification problems.

III. PROPOSED METHODS

In existing character-based DL models for DGA classification, characteristics among the connecting characters are considered. However, in word-looking DGAs, words from different dictionaries are concatenated. Table 2 shows details regarding the dictionary used in constructing word-looking DGAs. As all dictionaries contain valid English words, character-based encoding may not be effective in distinguishing the English words in different dictionaries.

To better capture the linguistic and semantic structure [22, 23] of the word-looking DGAs, a word-based encoding method can be used. In this section, we discuss a DL method for characterizing the word relationships in word-looking DGAs. In particular, we consider both CNN and LSTM models. These models can learn to identify patterns and relationships among words in a way that is not possible with character-based encoding. In this way, the word-looking DGA classification can be done more accurately.

To extract the relevant parts of domain names from URLs, we followed the procedures outlined in [20]. If the URL contains a second-level domain name, we extract the second-level part. If the URL contains a third-level domain name, we check if the second-level domain name is from a popular dynamic domain name service such as "no-ip.com" or "ddns.net". If it is, we extract the third-level domain part. If the second-level domain name is not from a popular dynamic domain name service, we extract the longer string consisting of the second-level and the third-level domain names. This procedure helps extract the most relevant parts of the domain name while avoiding any unnecessary information.

DGAs	Dictionaries
gozi	gpl: 4379 words, luther: 1537 words nasa: 558 words, rfc: 2460 words
matsnu	Use two dictionaries: verb dictionary (878 words) and noun dictionary (1008 words)
nymaim	Use two dictionaries: the first has 2450 words and the second has 4387 words
pizd	The dictionary contains 384 words
rovnix	Uses the US Declaration of Independence as the dictionary
suppobox	The dictionary in all versions contains 384 words

Table 2: Dictionaries used in different DGAs

The next step in processing the domain names is to decompose them into words, which can help capture the semantic meaning of the domains. For example, the domains "preparetwenty" and "coveredpublicandfrom" can be decomposed into {"prepare", "twenty"} and {"covered", "public", "and", "from"} respectively. These words are valid English words that carry meaning and can provide important information for DGA classification. The subsequent DL model will be trained to characterize the association among the connecting words. Both CNN and Bi-LSTM will be used for DGA classifications. CNNs are effective at learning local features and patterns in the data [15-17], while Bi-LSTM can capture sequence dependencies and relationships among the words [22-23].

The proposed CNN and Bi-LSTM structure for DGA classification consists of an embedding layer with an M number of units, which maps each word to a vector representation. For CNN, the convolutional layer has 128 filters and a kernel size of 2, which applies the filter to the word embeddings to extract local features. The Bi-LSTM layer consists of 200 units to capture long-term dependencies and relationships among the words in the domain names. The output of the convolutional layer or the Bi-LSTM layer is then passed through three fully connected layers with 256, 128, and 128 units respectively, before being fed into the output layer with L units, corresponding to the L number of classes in the DGA classification.

IV. Experimental results

UMUDGA is a public dataset designed for detecting and profiling algorithmically generated domain names in DGA-based botnet detection [4]. It contains over 30 million domain names from 50 DGA classes. In addition to the domain names, UMUDGA contains also basic statistical features, n-gram features, and NLP (natural language processing) features. In our study, we focus on the word-looking DGAs classification. To train and evaluate our proposed models, the dataset contains 10,000 legitimate domain names and 10,000 algorithmically generated domains for each of the word-looking DGA classes. This dataset allows us to test the effectiveness of our models in classifying different types of word-looking DGAs.

To evaluate and compare the performance, the precision, recall, and F1 scores are used. Precision is defined as the ratio between TP and (TP+FP) where TP is the number of samples that are in class A and are identified as A, and FP is the number of samples that are not in class A but are identified as A. Recall is the ratio between TP and (TP+FN) where FN is the number of samples that are in class A but is identified as not in A. In a perfect classification, both precision and recall should be equal to 1. In practice, increasing precision may decrease recall. The F1 score is the weighted average of precision and recall.

Table 3 shows the average F1 scores of the proposed CNN and Bi-LSTM models. We can see that both models achieved an average F1 score of 0.94 for DGA classifications. We compared our proposed models with both ML models and other character-based DL models. For DL models such as CNN (character), LSTM.MI [24], and Bi-LSTM [5], their average F1 scores are smaller than 0.88. Hence, our proposed models can achieve an increase of at least a 7.7% improvement in the F1 score. The improvement was much larger as compared to the ML models such as RF, DT, and kNN. Hence word encoding is beneficial for word-looking DGAs classification.

Table 4 and Table 5 show respectively the percentage improvement in the F1 scores of the proposed CNN and Bi-LSTM models against different ML and DL models for different DGAs. It clearly shows that deep learning models can always outperform machine learning models. Compared to

other character-based DL models, our proposed word encoding CNN and Bi-LSTM improve the F1 scores for all classes.

Besides, F1 scores, precision, and recall are also considered. Fig. 1 and Fig. 2 show respectively the boxplot of the precision and recall achieved by different models. Consistent with the F1 scores results, DL models have better performance than the ML models. In addition, word encoding methods perform better than character encoding methods.

	F1 score	
	Legitimate	DGAs
Proposed (CNN)	0.805	0.936
Proposed (Bi-LSTM)	0.773	0.944
CNN (character)	0.711	0.854
LSTM.MI [24, 5]	0.711	0.866
Bi-LSTM [5]	0.721	0.874
Random Forest (RF) [4]	0.662	0.682
Decision Tree (DT) [4]	0.582	0.572
kNN [4]	0.484	0.540

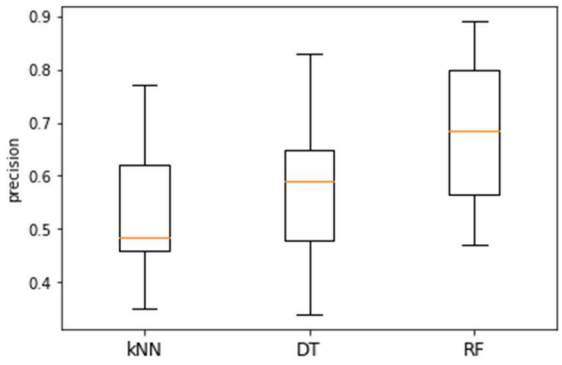
Table 3: The average F1 scores for the proposed CNN and Bi-LSTM models in comparison with other character-based deep learning models and machine learning models.

	kNN [4]	DT [4]	RF [4]	CNN (char)	LSTM .MI [24, 5]	Bi-LSTM [5]
legit	67	38	21	13	13	11
goziGpl	89	41	9.2	30	19	12
goziLuther	83	94	42	6.7	5.8	2.4
goziNasa	143	150	88	18	15	7.4
goziRfc	183	158	98	23	21	14
matsnu	20	20	7.9	6.7	6.1	8.0
nymaim	130	64	29	14	11	12
pizd	87	98	65	4.2	4.4	7.7
rovnix	92	68	45	12	9.6	6.8
suppobox1	102	106	70	2.1	3.0	6.5
suppobox2	24	24	18	2.1	3.1	4.2
suppobox3	23	20	11	1.0	0.6	0.7
Average	87	73	42	11	9.2	7.7

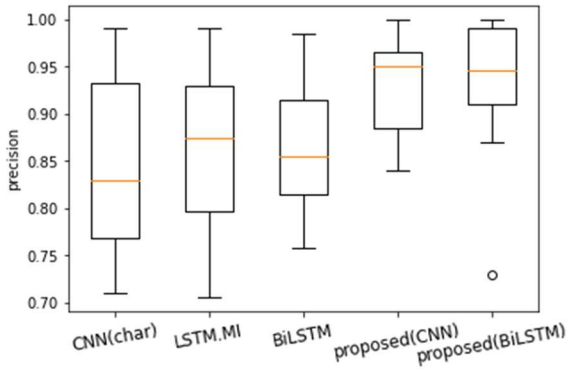
Table 4: The percentage improvement in the F1 score of the proposed CNN model against different ML and DL models for both legitimate and DGA classes.

	kNN [4]	DT [4]	RF [4]	CNN (char)	LSTM .MI [24, 5]	Bi-LSTM [5]
legit	60	33	17	8.5	8.3	6.8
goziGpl	93	44	12	33	21	15
goziLuther	83	94	42	6.7	5.8	2.4
goziNasa	158	156	92	21	17	9.8
goziRfc	186	160	100	25	23	15
matsnu	20	20	7.9	6.7	6.1	8.0
nymaim	133	65	30	15	12	13
pizd	87	98	65	4.2	4.4	7.7
rovnix	92	68	45	12	9.6	6.8
suppobox1	106	111	74	4.2	5.1	8.7
suppobox2	24	24	18	2.1	3.1	4.2
suppobox3	23	21	11	1.0	0.6	0.7
average	88	74	43	12	9.7	8.2

Table 5: The percentage improvement in the F1 score of the proposed Bi-LSTM model against different ML and DL models for both legitimate and DGA classes.

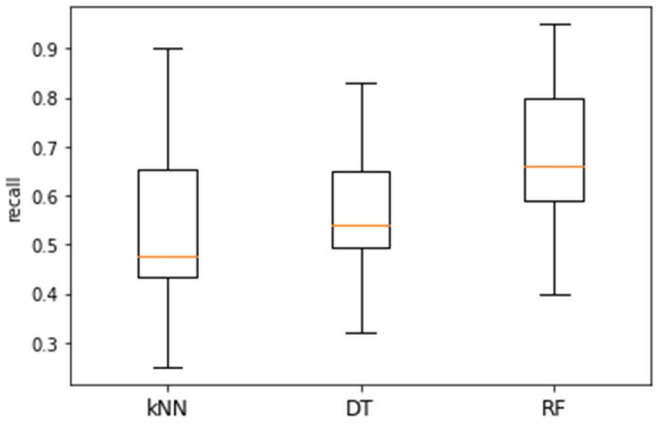


(a)

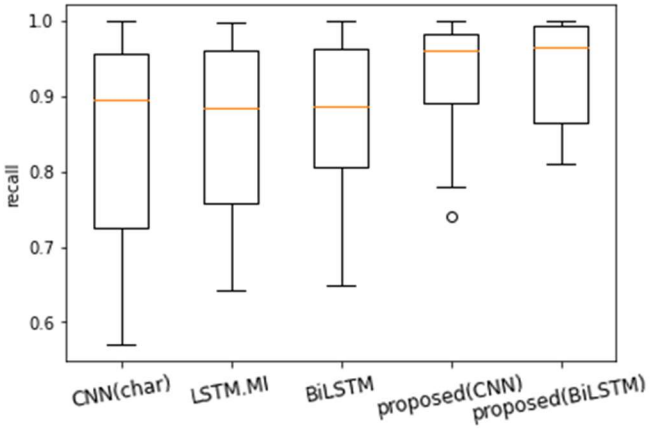


(b)

Fig 1: Boxplots of the precision achieved by (a) ML models such as kNN, DT, and RF, and (b) DL models such as character encoding CNN, LSTM.MI, Bi-LSTM, the proposed CNN model, and the proposed Bi-LSTM model.



(a)



(b)

Fig 2: Boxplots of the recall achieved by (a) ML models such as kNN, DT, and RF, and (b) DL models such as character encoding CNN, LSTM.MI, Bi-LSTM, the proposed CNN model, and the proposed Bi-LSTM model.

V. CONCLUSIONS

Existing state-of-the-art DGA classifications are ineffective at classifying word-looking DGAs. It is because word-looking DGAs are constructed by concatenating words from dictionaries. Features used in machine learning models are not designed to identify such patterns. The character encoding adopted by existing deep learning models achieves a better performance. To further improve the performance, the word encoding technique is applied to study the association of different words in the domain names. Experimental results show that the proposed word encoding-based CNN and Bi-LSTM models achieve better classification performance than both the machine learning models and character encoding-based deep learning models.

REFERENCES

[1] G. Vormayr, T. Zseby, and J. Fabini, "Botnet Communication Patterns", IEEE Communications Surveys & Tutorials, 19, 2768-2796, 2017.

[2] Z. Wang, and Y. Guo, "Neural networks based domain name generation", Journal of Information Security and Applications, 61, 102948, 2021.

[3] T. Wang, L. Chen, and Y. Genc, "A dictionary-based method for detecting machine-generated domains", Information Security Journal: A Global Perspective, Vol. 30, Issue 4, 2020.

[4] M. Zago, M. G. Pérez, and G. M. Pérez, "UMUDGA: A dataset for profiling DGA-based botnet", Computers & Security, 92, 101719, 2020.

[5] A. Cucchiarelli, C. Morbidoni, L. Spalazzi, and M. Baldi, "Algorithmically generated malicious domain names detection based on n-grams features", Expert Systems with Applications, 170, 114551, 2021.

[6] F. Ren, Z. Jiang, X. Wang, and J. Liu, "A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network", Cybersecurity, 3, 1-13, 2020.

[7] P. Vij, S.D. Nikam, and A. Bhatia, "Detection of Algorithmically Generated Domain Names using LSTM", International Conference on COMmunication Systems & NETworkS (COMSNETS), 1-6, 2020.

[8] H.P. Vranken, and H. Alizadeh, "Detection of DGA-Generated Domain Names with TF-IDF", Electronics, 11, 414, 2022.

[9] A.O. Almashhadani, M. Kaiiali, D. Carlin, and S. Sezer, "MaldomDetector: A system for detecting algorithmically generated domain names with machine learning", Computers & Security, 93, 101787, 2020.

[10] Y. Qiao, B. Zhang, W. Zhang, A.K. Sangaiah, and H. Wu, "DGA Domain Name Classification Method Based on Long Short-Term Memory with Attention Mechanism", Applied Sciences, 9, 4205, 2019.

[11] J.P. Selvi, R.J. Rodríguez, and E. Soria-Olivas, "Toward Optimal LSTM Neural Networks for Detecting Algorithmically Generated Domain Names", IEEE Access, 9, 126446-126456, 2021.

[12] Z. Wang, Y. Guo, and D. Montgomery, "Machine learning-based algorithmically generated domain detection", Computers and Electrical Engineering, 100, 107841, 2022.

[13] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-Away Traffic to

Bots: Detecting the Rise of DGA-Based Malware", USENIX Security Symposium, 24, 2012.

[14] L. Bilge, S. Şen, D. Balzarotti, E. Kirda, and C. Krügel, "Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains", ACM Transactions on Information Systems Security, 16, 14, 2014.

[15] Y. Liu, Z. Zhou, Y. Yang, N.F. Law and A.A. Bharath, "Efficient Source Camera Identification with Diversity-enhanced Patch Selection and Deep Residual Prediction", Sensors, 21(14), 4701, 2021.

[16] A. Zhang, Z. Zou, Y. Liu, Y. Chen, Y. Yang, Y. Chen, N.F. Law, "Automated Detection of Circulating Tumor Cells using Faster Region Convolution Neural Network", Journal of Medical Imaging and Health Informatics, 9(1), 167-174, 2019.

[17] M. Irshad, N.F. Law, K.H. Loo and S. Haider, "IMGCAT: an Approach to Dismantle the Anonymity of a Source Camera using Correlative Features and an Integrated 1D Convolutional Neural Network", Array, 18, 100279, 2023.

[18] J. Woodbridge, H. Anderson, A. Ahuja, and D. Grant, "Predicting Domain Generation Algorithms with Long Short-Term Memory Networks", ArXiv, abs/1611.00791, 2016.

[19] Z. Feng, C. Shuo, and W. Xiaochuan, "Classification for DGA-Based Malicious Domain Names with Deep Learning Architectures", International Journal of Intelligent Information Systems, Vol. 6, Issue 6, 67-71, 2017.

[20] B. Yu, D.L. Gray, J. Pan, M.D. Cock and A.C. Nascimento, "Inline DGA Detection with Deep Networks", 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 683-692, 2017.

[21] H. Mac, D. Tran, V. Tong, L.G. Nguyen, and H.A. Tran, "DGA Botnet Detection Using Supervised Learning Methods", Proceedings of the Eighth International Symposium on Information and Communication Technology, 211-218, 2017.

[22] S.R.C. Liew, and N.F. Law, "BEAM – an Algorithm for Detecting Phishing Link", Proceedings of 2022 APSIPA Annual Summit and Conference, 598-604, 2022.

[23] S.R.C. Liew and N.F. Law, "Use of Subword Tokenization for Domain Generation Algorithm Classification", Cybersecurity (accepted), 2023, doi: 10.1186/s42400-023-00183-8.

[24] D. Tran, H. Mac, V. Tong, H.A. Tran, and L.G. Nguyen, "A LSTM based Framework for Handling Multiclass Imbalance in DGA Botnet Detection", Neurocomputing, 275: 2401, 2018.