Liew and Law *Cybersecurity (2023)6:49*
https://doi.org/10.1186/s42400-023-00183-8

Cybersecurity

RESEARCH

Open Access

Use of subword tokenization for domain generation algorithm classification

Sea Ran Cleon Liew[1*] and Ngai Fong Law[2]

Abstract

Domain name generation algorithm (DGA) classification is an essential but challenging problem. Both feature-extracting machine learning (ML) methods and deep learning (DL) models such as convolutional neural networks and long short-term memory have been developed. However, the performance of these approaches varies with different types of DGAs. Most features in the ML methods can characterize random-looking DGAs better than word-looking DGAs. To improve the classification performance on word-looking DGAs, subword tokenization is employed for the DL models. Our experimental results proved that the subword tokenization can provide excellent classification performance on the word-looking DGAs. We then propose an integrated scheme that chooses an appropriate method for DGA classification depending on the nature of the DGAs. Results show that the integrated scheme outperformed existing ML and DL methods, and also the subword DL methods.

Keywords Botnet detection, Domain names, Network security, Machine learning-based botnet detection

Introduction

Network attacks are common nowadays. A botnet is a group of Internet-connected devices such as Internet-of-Things (IoTs) and computers that are controlled by third-party software to perform specific tasks without the knowledge of device owners (Negash and Che 2015; Kambourakis et al. 2019). The instructions are sent from a "command and control" (C&C) server. In the past, the C&C server had a fixed IP address or domain name for communicating with the bots. However, such communication can be blocked easily by approaches such as a blacklist. To better avoid detection, recent bots and C&C servers employ domain name generation algorithms (DGA) for communication (Vormayr et al. 2017). The algorithm generates a large set of domain names, but only a few are registered. The bots would try resolving these domain names successively until they are able to connect to the C&C server. By using this approach, the C&C server is hidden which makes its detection difficult.

There are different DGA generation schemes such as hash-based, arithmetic-based, wordlist-based, and permutation-based methods (Wang and Guo 2021; Wang et al. 2020). The hash-based and arithmetic-based methods produce random-looking domain names such as "ukphbhncsdpgo.com" that appear to be different from legitimate domains. The wordlist-based and permutation-based methods produce word-looking domain names by joining or permuting words from a dictionary to create domain names. Several approaches have been proposed for detecting and classifying these domain names, including machine learning-based and deep learning-based methods (Almashhadani et al. 2020; Berman 2019; Cucchiarelli et al. 2021; Qiao et al. 2019; Ren et al. 2020; Selvi et al. 2021; Vij et al. 2020; Vranken and Alizadeh 2022; Wang et al. 2022; Hoang and Vu 2022). A recent review paper (Saeed et al. 2021) provides an overview of these methods. Detection involves determining whether a domain name is generated by a DGA scheme, whereas classification aims to identify the DGA method

*Correspondence:
Sea Ran Cleon Liew
srcliew@uwaterloo.ca
[1] University of Waterloo, Waterloo, Canada
[2] Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong

used to generate the domain name. As reported in Zago et al. 2020a, binary detection achieves over 90% accuracy, while multi-class classification achieves around 70%. This indicates that identifying the DGA method used to generate domain names is more challenging than distinguishing them from legitimate ones (Cucchiarelli et al. 2021; Ren et al. 2020; Vij et al. 2020). Especially some classes have extremely low classification accuracy. Therefore, there is much room for further improvement in multi-class classification.

In this paper, we study the classification performance of existing approaches on random-looking and word-looking domains. In machine learning-based approaches, features about the character distributions in the domain names have been used. Examples include the vowels and consonant ratios or the numerals and English characters ratios (Vranken and Alizadeh 2022). 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 (Ren et al. 2020). In contrast, existing deep learning methods are more effective in characterizing word-looking domain names. To further improve the existing deep learning methods, we propose using subword tokenization to study the domain names. The rationale is that subword tokenization is better suited for studying relationships among connecting words than character tokenization (Liew and Law 2022).

In addition, we propose an integrated scheme containing two classifiers. One classifier focuses on characterizing the random-looking DGA while the other focuses on word-looking domains. We also develop a metric to distinguish between random-looking DGA and word-looking domains. In summary, we study the following research questions in this paper:

- Are there any performance differences between existing machine learning (ML) and deep learning (DL) approaches in DGA multi-class classification? Which type of domain names can be more accurately classified by ML? Which type of domain names can be more accurately classified by DL?
- Is subword tokenization better than character tokenization in characterizing word-looking domain names?
- How can we determine whether ML or DL is better suited for detecting the class of a given DGA?
- Can we combine ML and DL approaches for achieving better DGA multi-class classification performance?

The main contributions of this paper are:

- We examine the effectiveness of integrating subword tokenization into DL models to characterize the word-looking DGAs. This represents an improvement over existing DL approaches which primarily use character tokenization only.
- We examine the performance difference between ML and DL in DGA multi-class classification and find out the types of DGA better classified by ML and DL. An algorithm is developed to determine whether a testing DGA is better classified by ML or DL.
- A scheme is developed to integrate the advantages of feature-extracting ML and subword tokenization DL in DGA multi-class classification.

The remainder of this article is organized as follows. First, we give an overview of the problem and compare the performance of the existing state-of-the-art machine learning and deep learning methods in "Background" section. Next, we describe the proposed subword tokenization for building a deep learning model in "The proposed subword-based deep learning model (SW-CNN and SW-LSTM)" section. "Proposed integrated scheme" section then shows the integrated scheme and how we quantify the nature of the DGAs. Experimental results are given in "Experimental results" section. Finally, we conclude our work in "Conclusions" section.

Background

There are two problems associated with DGA: DGA detection and DGA classification (Zago et al. 2020a). DGA detection is a binary classification problem. The aim is to distinguish if the domain name is legitimate or algorithmically generated. The DGA classification is a multi-class classification problem. It aims to further classify the domain names into several types of DGA methods that have been used to generate them. Both machine learning (ML) and deep learning (DL) methods have been used for DGA detection and classification (Almashhadani et al. 2020; Berman 2019; Cucchiarelli et al. 2021; Qiao et al. 2019; Ren et al. 2020; Selvi et al. 2021; Vij et al. 2020; Vranken and Alizadeh 2022; Wang et al. 2022). Interested readers may refer to paper (Saeed et al. 2021) for a recent survey of ML and DL methods for DGA detection.

In ML methods, feature extraction is a crucial step to characterize the nature of the domain names. Features representing expert knowledge are extracted from the domain names to define the algorithmically generated domains' characteristics. Common features can be divided into statistical features, information theory features, and lexicographic features. For example, vowel-consonant ratio, n-gram distributions, the longest consecutive consonant/number/vowel sequences,

pronounceability score, and entropy (Almashhadani et al. 2020; Antonakakis et al. 2012; Bilge et al. 2014) have been used to characterize the nature of the algorithmically generated domain names. A detailed list of features can be found in Vranken and Alizadeh (2022).

After features are extracted, various ML models have been used for DGA detection and classification. Table 1 provides a summary of recent methods, including the features used, the ML models, the dataset sizes, and the detection/classification results (Almashhadani et al. 2020; Cucchiarelli et al. 2021; Vranken and Alizadeh 2022; Wang et al. 2022; Zago et al. 2020a). Generally, DGA detection has a much better performance than its classification. For DGA detection, the accuracy or the F1 score is always higher than 0.95 for all the methods. For DGA classification, the F1 score is much lower than that. It is between 0.297 and 0.823. We can see the difficulty in DGA classification. The extracted features can distinguish DGA from legitimate domains. However, they are not sufficient in characterizing different DGA classes. Further research on feature extraction should be carried out to improve the multi-class classification performance.

Deep learning approaches have also been used for DGA detection and classification. Unlike machine learning, no feature extraction is done. Rather, the domain name is considered a string of characters. With the use of a sufficient number of examples, a learning model is trained to distinguish and characterize the DGA. In this way, there is no need for manual feature extraction. As domain names consist of characters, tokenization, and embedding are required to convert the domain names to numerical sequences. In literature, the most popular method is to use character tokenization in which a domain name is decomposed into a sequence of characters. These characters are encoded independently and mapped to integers in the embedding layer. The resultant numerical sequences are then fed into the DL models for training.

Convolutional neural networks (CNN) and long short-term memory (LSTM) are popular DL models. LSTM is often used for acquiring patterns in long sequences in different applications (Qiao et al. 2019; Selvi et al. 2021; Vij et al. 2020; Woodbridge et al. 2016; Xu et al. 2022). For CNN, a filter kernel with varied sizes is used to characterize sequence relationships (Berman 2019; Feng et al. 2017; Yu et al. 2017). CNN can also be combined with LSTM to improve detection and classification performance (Ren et al. 2020; Mac et al. 2017). Table 2 provides a summary of recent methods, including the model structures, the dataset sizes, and the detection/classification results. Like the ML results, classification is more challenging than detection. In DGA detection, the F1 score (in Eq. (4)) is always higher than 0.97. However, in terms of classification, the score can be as low as 0.7 for some datasets. There is much room for further improvement in multi-class classification. Also, in most of the experimental settings, the number of benign domain names is a lot more than the number of DGA in each class (Berman 2019; Qiao et al. 2019; Ren et al. 2020; Selvi et al. 2021;

Table 1 A summary of recent ML methods for DGA detection and classification

Detection or classification problem	Features	ML models	Dataset	Results
Detection (Almashhadani et al. 2020)	Lexical features	DT, SVM, kNN	85,000 benign 85,000 DGA from 20 classes	F1 scores: 0.9437 (DT) 0.9411 (SVM) 0.9443 (kNN)
Detection (Wang et al. 2022)	Distance-based features (KL distance, edit distance, Jaccard index)	SVM, NN	10,000 benign, 10,000 DGA from 12 classes	Accuracy close to 1
Classification (Vranken and Alizadeh 2022)	TF-IDF of the n-grams in domain names	SVM, MLP, RF, DT, kNN	583,9543 benign 492,800 DGA from 57 classes	F1 scores: 0.7573 (SVM) 0.7759 (MLP) 0.6284 (RF) 0.6443 (DT)
Detection and Classification (Zago et al. 2020a, b)	Lexical features	Adaboost, NN, RF, SVM, DT, kNN	10,000 benign 50 DGA classes, each has 10,000	F1 scores: Detection 0.556–0.989 Classification 0.297–0.769
Detection and Classification (Cucchiarelli et al. 2021)	n-gram features	MLP	10,000 benign 50 DGA classes, each has 10,000	F1 scores: Detection 0.964 Classification 0.823

Table 2 A summary of DL methods for DGA detection and classification

Detection or classification problem	DL models	Dataset	F1 score
Detection (Berman 2019)	CNN embedding + CNN (1D) + fully connected layers	1 million benign 852,116 DGA from 50 classes	0.9933
Detection (Selvi et al. 2021)	LSTM: embedding + LSTM + fully connected layers	32,000 benign 32,000 DGA	0.9762
Classification (Qiao et al. 2019)	LSTM with attention: embedding + LSTM + attention + fully connected layers	910,313 benign 765,091 DGA from 15 classes: 759,091 DGA from 14 random-looking classes 6000 from 1 word-looking class	0.9458
Detection and Classification (Vij et al. 2020)	LSTM: embedding + LSTM + fully connected layers	109,935 benign 109,935 DGA from 11 classes (all are random-looking DGAs)	Detection: 0.9804 Classification: 0.7192
Detection and Classification (Ren et al. 2020)	CNN-BiLSTM with attention: embedding + CNN + LSTM + attention + fully connected layer	1 million benign 308,230 DGA from 24 classes: 19 arithmetic-based 2 wordlist-based 3 part-wordlist-based	Detection: 0.9879 Classification: 0.8300
Detection (Yang et al. 2022)	Subword tokenization and transformer	10,000 benign and 10,000 DGA from 9 classes: (one wordlist-based DGA)	Detection: 0.9697

Vij et al. 2020). Hence, the overall result may not reflect the performance in each DGA class.

It is important to identify the advantages and limitations of existing approaches in DGA classification. There are two types of DGAs, namely the random-looking DGA and the word-looking domain names (Selvi et al. 2021). We will study if these ML and DL models have similar performance on both DGA types. Based on the published classification results, a summary of current state-of-the-art results is made in Table 3.

In most of the experimental setups, there are more random-looking DGAs than word-looking DGAs. Table 3 shows that most setups use 10% or fewer word-looking DGAs to build the model. If the number of word-looking DGAs is small, the training in the DL model may not be able to characterize these word-looking DGAs very well. If there are more word-looking DGAs, DL models like BiLSTM and CNN can provide a good characterization for these domains. As in Cucchiarelli et al. (2021) which contains 11 word-looking DGA classes, the F1 score for word-looking DGAs is higher than that for random-looking DGAs.

In general, machine learning-based methods can characterize the random-looking DGAs better than word-looking DGAs. This is consistent with findings from other authors (Ren et al. 2020). Despite that, an exception is the n-gram features. As the n-gram features capture specifically the relationship among consecutive characters distribution, it models the word relationship better than other types of manual features, such as vowel-consonant ratios.

Based on the summary, we see there is a performance difference between existing ML and DL approaches. When more word-looking DGAs are used for training, DL is better for word-looking DGAs than random-looking DGAs. By combining the use of ML and DL, the classification performance of word-looking and random-looking domain names can be improved. For the existing DL methods, character tokenization is used. Since n-gram is good for classifying word-looking domains, inspired by this idea, we investigate if subword tokenization can be used to better capture the characteristics of word-looking DGAs.

The proposed subword-based deep learning model (SW-CNN and SW-LSTM)

While existing character tokenization can, to a certain extent, characterize the relationship among connecting tokens, word-based tokenization can better preserve the linguistic and semantic structure (Liew and Law 2022) in the word-looking DGAs. However, the random-looking DGA does not have any semantic meaning and thus word-based tokenization would not be able to give any meaningful representation. Hence, we consider a subword tokenization method to tokenize the domain names. Words that can be found in a dictionary are formed as tokens, while other random parts are decomposed into characters for tokenization.

Figure 1 shows the schematic diagram of our proposed subword-based DL model. The model contains two branches for characterizing the domain names, one branch considers character tokenization, and the

Table 3 A summary of the F1 score for word-looking and random-looking DGAs from existing state-of-the-art DGA classification methods

Dataset	Characteristics	Methods	F1 scores	
			Word-looking DGAs	Random-looking DGAs
1	11 word-looking DGAs, 39 random-looking DGAs, Ratio (W/R) = 0.282 (Zago et al. 2020b)	ML (lexical features, RF) (Zago et al. 2020a)	0.6820	**0.7360**
		ML (n-gram) (Cucchiarelli et al. 2021)	**0.9084**	0.7848
		BiLSTM (Cucchiarelli et al. 2021)	**0.8745**	0.6989
		CNN-BiLSTM (Cucchiarelli et al. 2021)	**0.9010**	0.7026
2	2 word-looking DGAs, 19 random-looking DGAs, Ratio (W/R) = 0.105	ML (SVM) (Ren et al. 2020)	0.3670	**0.6180**
		LSTM (Ren et al. 2020)	0.5500	**0.6600**
		CNN (Ren et al. 2020)	0.5135	**0.7053**
		CNN-BiLSTM (Ren et al. 2020)	0.4503	**0.7009**
		CNN-BiLSTM with attention (Ren et al. 2020)	**0.8854**	0.7853
3	4 word-looking DGAs, 53 random-looking DGAs, ratio (W/R) = 0.0075	ML (MLP) (Vranken and Alizadeh 2022)	**0.7887**	0.7750
		ML (RF) (Vranken and Alizadeh 2022)	0.3444	**0.6498**
		ML (SVM) (Vranken and Alizadeh 2022)	**0.8371**	0.7513
		LSTM (Vranken and Alizadeh 2022)	0.7331	**0.8348**
4	1 word-looking DGA, 14 random-looking DGAs Ratio (W/R) = 0.071	LSTM (Qiao et al. 2019)	0.1626	**0.9445**
		LSTM with attention (Qiao et al. 2019)	0.1743	**0.9458**
5	11 random-looking DGAs	LSTM (Vij et al. 2020)	–	**0.7192**

Bold values indicate the type of DGA that have a better result for each method

W and R denote respectively the number of word-looking and random-looking classes

other branch considers subword tokenization. These two branches are merged at the end to achieve a better domain name classification. Note that each URL will go through both branches so that both character and word relationships can be extracted.

The relevant parts of the domain names will first be extracted from the URLs in the pre-processing block. The domain names are extracted as follows (Yu et al. 2017). If the URL contains a second-level domain name, the second-level part is extracted. If it is a third-level domain name, the second-level domain name is checked to see if it is from a popular dynamic domain name service such as "no-ip.com", "dnsdynamic.org" or "ddns.net". If so, the third-level domain part is extracted. If not, the longer string from the second-level and the third-level domain name is extracted. For example, the URL "ab1cf5d50e7da6.com" will be processed to give "ab1cf5d50e7da6", while "akboavenifbiuc.ddns.net" produces "akboavenifbiuc" only.

The output from the pre-processing block will be fed into two branches for further analysis. The first branch uses character tokenization while the second uses subword tokenization. In character tokenization, each character is encoded independently. For example, the tokens for "shopee.ph" are {'s', 'h', 'o', 'p', 'e', 'e'}. Each character token is then mapped to an integer. The character set includes the English alphabet, numbers, and special characters. For example, the above token may become {19, 8, 15, 16, 5, 5} after encoding. The number of tokens varies with different domain names. However, inputs to the deep learning models need to be uniform with equal

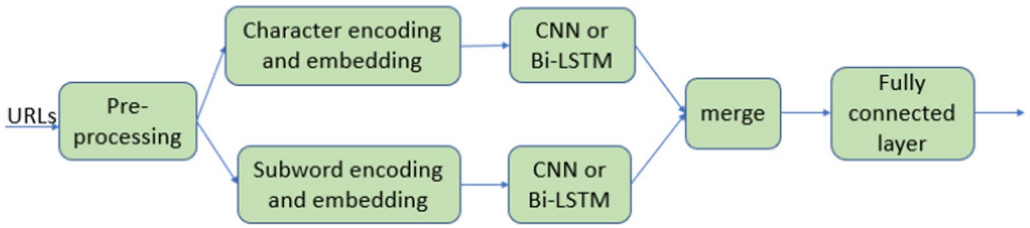

Fig. 1 The schematic diagram of the proposed SW-CNN and SW-LSTM DL models

lengths. Padding is thus required so that all resultant embedding vectors have uniform lengths.

The steps for subword tokenization are like those for character tokenization, except the subword replaces the character as tokens. For example, the URL "shopee.ph" becomes "shopee" after pre-processing. Using subword tokenization, tokens are {'shop', 'ee'}. The tokens preserve common words that can be found in a dictionary. Some more examples are shown in Table 4.

For random-looking DGAs such as those in the "ramnit" class, the extracted subwords are short in length. However, for word-looking DGA, the extracted subwords are English words carrying semantic meaning. Thus, the subsequent deep learning model can be used to characterize the relationship among the connecting words.

As discussed in "Background" section, both convolutional neural networks (CNN) and Bi-directional Long Short-Term Memory (Bi-LSTM) have been used for DGA detection and classification in the literature (Berman 2019; Qiao et al. 2019; Ren et al. 2020; Selvi et al. 2021; Vij et al. 2020; Woodbridge et al. 2016; Feng et al. 2017; Yu et al. 2017; Mac et al. 2017). These two models will be investigated in this study to model the word relationships. The proposed structure is shown in Fig. 1. We called our proposed subword models using CNN and Bi-LSTM as SW-CNN and SW-LSTM respectively. In summary, the proposed models capture the relationships between words and characters that are combined to generate the final DGA classification.

Proposed integrated scheme

Subword-based DL methods (SW-CNN and SW-LSTM) produce meaningful representations for word-looking DGA. However, they may not be the best for characterizing random-looking DGA. It is thus advantageous to use an integrated scheme so that an appropriate model can be adopted for classifying different types of DGAs. Figure 2 shows the integrated scheme which contains two classifiers, the ML model and the DL model. In the training phase, both the ML and DL models are trained. The ML model can be either a random forest, XGBoost, or other classifiers that characterizes the random-looking DGA well. The DL model is either the proposed SW-CNN or SW-LSTM which focuses on characterizing word-looking DGA. In the testing phase, a randomness indicator is obtained which classifies the domain names into either the random-looking or word-looking type so that an appropriate model is adopted for final classification. By using this proposed

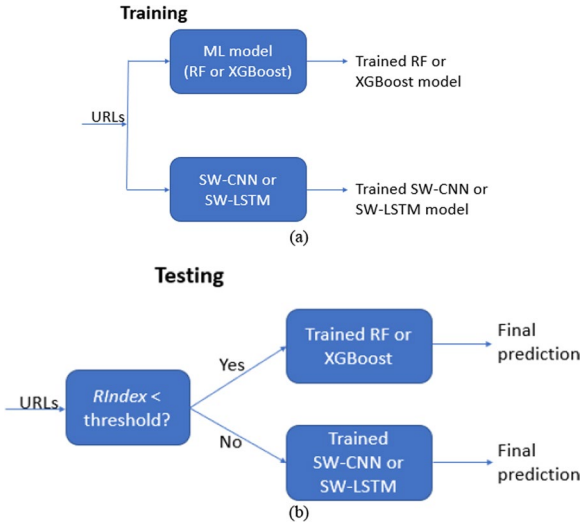

Fig. 2 The proposed integrated scheme, **a** training phase, and **b** testing phase

Table 4 Examples of character and subword tokenization for different DGA types

DGA Type	Class name	URL	*RIndex*
Word-looking	pizd	animalforget.net Character tokenization: "a", "n", "i", "m", "a", "l", "f", "o", "r", "g", "e", "t" Subword tokenization: "animal", "for", "get"	$N(Tokens_{char})$=12 $N(Tokens_{subword})$=3 *RIndex* = 0.75
Word-looking	rovnix	thesetobewarfarebecomes.net Character tokenization: "t", "h", "e", "s", "e", "t", "o", "b", "e", "w", "a", "r", "f", "a", "r", "e", "b", "e", "c", "o", "m", "e", "s" Subword tokenization: "these", "to", "be", "war", "fare", "be", "com", "es"	$N(Tokens_{char})$=23 $N(Tokens_{subword})$=8 *RIndex* = 0.65
Random-looking	ramnit	ukphbhncsdpgo.com Character tokenization: "u", "k", "p", "h", "b", "h", "n", "c", "s", "d", "p", "g", "o" Subword tokenization: "uk", "p", "hb", "hn", "cs", "dp", "go"	$N(Tokens_{char})$=13 $N(Tokens_{subword})$=7 *RIndex* = 0.46

integrated method, the DGA can be classified more appropriately depending on its nature.

The randomness index, *RIndex*, is used to indicate the nature of the DGAs. It can be constructed by comparing the subword and character tokenization. In particular, the change of the number of tokens in subword tokenization with reference to the number of tokens in character tokenization can be employed. It is defined as,

$$RIndex = \frac{N(Tokens_{char}) - N(Tokens_{subword})}{N(Tokens_{char})} \quad (1)$$

where $N(Tokens_{char})$ and $N(Tokens_{subword})$ denote respectively the number of tokens in the character tokenization and subword tokenization. *RIndex* must be larger than 0 as $N(Tokens_{char}) \geq N(Tokens_{subword})$. It is also smaller than 1. If the domain contains mostly words, the number of subword tokens would be small which thus gives a large *RIndex*. On the other hand, if the domain is random, the number of subword tokens and character tokens would be similar. In this case, *RIndex* would be small. As shown in Table 4, the *RIndex* of the word-looking DGAs is 0.75 and 0.65. In contrast, for the random-looking DGA, it is 0.46 which is smaller in value. Hence, *RIndex* can indicate the nature of the domain names.

Experimental results

UMUDGA is a public dataset designed for profiling algorithmically generated domain names in botnet detection. It has a collection of over 30 million domain names from 50 DGA classes (Zago et al. 2020a, b). Out of the 50 classes, 39 produce random-looking domain names, and 11 produce word-looking domain names. In our experimental testing, multi-class DGA classification is considered. To construct the dataset, 10,000 legitimate domain names and 10,000 algorithmically generated domain names for each of the 50 classes are collected. The multi-class classification problem becomes classifying the domain names into one of the 51 classes. This setting is the same as that in Cucchiarelli et al. (2021) and Zago et al. 2020a).

To evaluate and compare the performance, the precision, recall, and F1 scores are used. They are defined as follows,

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

where TP, FP, and FN stand for true positive, false positive, and false negative respectively for each class. For a class A, TP is the number of samples that are in A and are identified as A. FP is the number of samples that are not in A but are identified as A. FN is the number of samples that are in A but being identified as not in A. Precision tells the percentage of samples that are correctly classified in the classification results. Recall means the percentage of samples in a class that can be correctly classified. In a perfect classification, both precision and recall are 1. In practice, increasing precision may decrease recall. F1 score is thus used to quantify the performance. It is the weighted mean of precision and recall.

We will first compare the performance of subword tokenization and character tokenization in characterizing the word-looking and random-looking DGAs. We will then evaluate the performance of our proposed integrated scheme. Additionally, we will study the effectiveness of using *RIndex* to identify the nature of the domain names.

Performance of subword tokenization in SW-CNN and SW-LSTM

In this part, we study the performance of using the subword tokenization in Fig. 1 as compared to the existing state-of-the-art approaches which use character tokenization only. Table 5 summarizes the performance of the proposed SW-CNN and SW-LSTM and compares it with the existing CNN (Ren et al. 2020) and LSTM (Cucchiarelli et al. 2021) approaches.

We can see that the use of subword encoding improves the overall classification performance. In CNN, the overall average accuracy improves from 0.7304 to 0.7589 by incorporating the subword encoding. For Bi-LSTM, the accuracy increases from 0.7304 to 0.7568. Hence, the subword encoding can help one performs a better DGA classification.

In order to have a further understanding of the classification performance, we examine the performance of the two models on both random-looking and word-looking DGAs. As shown in Table 5, the subword information is beneficial for characterizing word-looking DGAs. For both CNN and BiLSTM, the addition of the subword encoding can improve the F1 score, precision, and recall in the word-looking DGA significantly. It yields an improvement range of 9.59% to 13.64%. The average F1 for word-looking DGA improves from 0.8536 to 0.9364 in SW-CNN and 0.8345 to 0.9436 in SW-LSTM. Thus, subword tokenization is capable of modeling the word relationship which can be used to characterize the word-looking DGA well.

While Table 5 shows the average performance, a boxplot is employed to show the distribution of the F1 score

Table 5 A summary of the performance of the proposed subword DL models: SW-CNN and SW-LSTM and their character counterparts

	Average precision	Average F1	Average recall
Proposed SW-CNN			
Word-looking DGA (average of 11 classes)	0.9355	0.9364	0.9391
Improvement char-CNN (Ren et al. 2020)	9.59%	9.70%	9.89%
Random-looking DGA (average of 39 classes)	0.7059	0.6728	0.6931
Improvement over char-CNN (Ren et al. 2020)	1.70%	1.89%	0.64%
Proposed SW-LSTM			
Word-looking DGA (average of 11 classes)	0.9473	0.9436	0.9409
Improvement char-LSTM (Cucchiarelli et al. 2021)	13.64%	13.07%	11.65%
Random-looking DGA (average of 39 classes)	0.7031	0.6731	0.6921
Improvement char-LSTM (Cucchiarelli et al. 2021)	0.59%	1.40%	0.52%

from each DGA class. Figure 3 shows the boxplot of the F1 score obtained from the proposed SW-CNN and SW-LSTM as compared to the existing CNN and LSTM approaches in both random-looking and word-looking DGAs. For random-looking DGA, the performance is similar no matter whether subword information is used or not. However, for word-looking DGA, the F1 score is significantly improved. As shown in the boxplot, the third quantile in the F1 score for both SW-CNN and SW-LSTM has been improved significantly by using the subword information. Results show that the performance of each of the word-looking DGA classes can be improved by using the proposed subword approaches. Boxplots for precision and recalls look similar and thus are not shown.

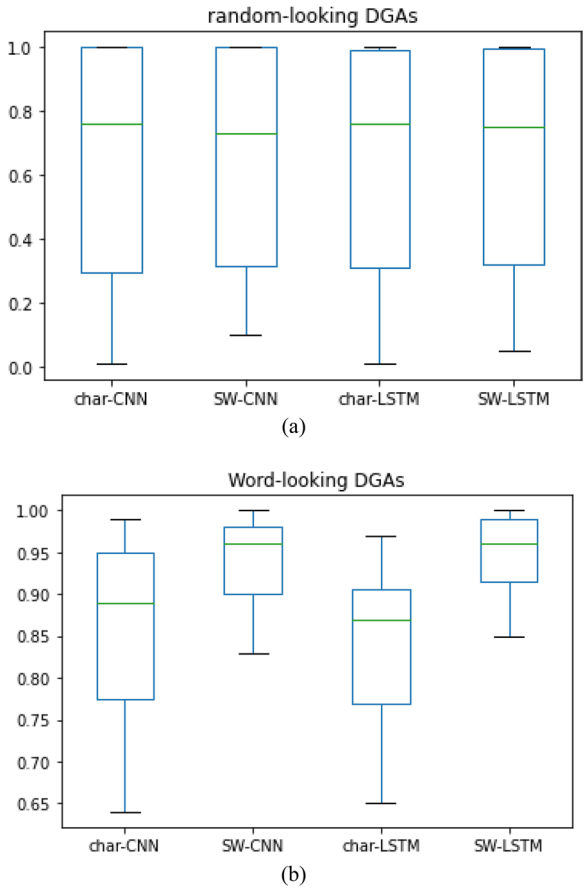

Fig. 3 The boxplots of the F1 score for **a** the random-looking DGAs and **b** the word-looking DGAs. Note that SW-CNN and SW-LSTM denote our proposed subword DL models, while char-CNN and char-LSTM denote the CNN and LSTM models using character tokenization

Performance of the integrated scheme

The integrated scheme requires the setting for RIndex. Figure 4 shows the boxplot of the distribution of the *RIndex* value on the word-looking and random-looking DGAs. We can clearly see that word-looking DGAs and random-looking DGAs have vastly different distributions on the *RIndex* value. The third quartile of the *RIndex* value in random-looking DGA is 0.4545 which is much smaller than the first quartile of the *RIndex* value in word-looking DGA (0.6471). Hence, by using the *RIndex* value, one can easily distinguish if the domain is random or is formed by concatenating words in the dictionary. In the experiment, we will set the threshold of the RIndex value to 0.55.

From "Background" and "Performance of subword tokenization in SW-CNN and SW-LSTM" sections, the performance of the models varies depending on the type of DGA being classified. In particular, the extracted

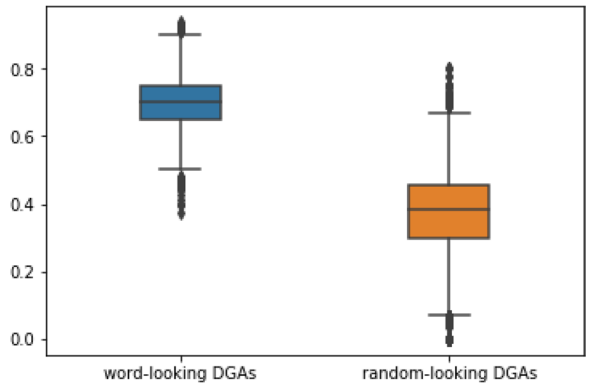

Fig. 4 The boxplot of the distribution of *RIndex* for word-looking and random-looking DGAs

features in machine learning approaches (Zago et al. 2020a) perform better on the random-looking DGA while our proposed SW-CNN and SW-LSTM perform better on the word-looking DGA. The advantage of these two approaches is integrated into the proposed integrated scheme to achieve a better DGA classification. In the following, we will examine the performance of the integrated scheme using random forest (RF) and XGBoost.

Integrated scheme with random forest

In this sub-section, we consider integrating the random forest classifier with the proposed SW-CNN and SW-LSTM models. Table 6 shows the classification results. The overall performance of the integrated scheme is better than the RF, SW-CNN, SW-LSTM, or CNN-BiLSTM.

Table 6 A summary of the performance of the proposed integrated schemes RF + SW-CNN (random forest with SW-CNN) and RF + SW-LSTM (random forest with SW-LSTM)

	Average precision	Average F1	Average recall
Proposed RF + SW-CNN			
Overall	0.7893	0.7812	0.7895
Improvement over RF (Saeed et al. 2021)	7.86%	7.92%	7.93%
Improvement over SW-CNN	4.00%	6.59%	5.52%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	3.90%	4.46%	3.92%
Random-looking DGAs	0.7487	0.7377	0.7472
Improvement over RF (Saeed et al. 2021)	0.44%	0.23%	0.43%
Improvement over SW-CNN	6.06%	9.65%	7.81%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	4.15%	5.00%	4.23%
Word-looking DGAs	0.9300	0.9327	0.9391
Improvement over RF (Saeed et al. 2021)	37.13%	36.76%	35.51%
Improvement over SW-CNN	− 0.59%	− 0.40%	0%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	3.58%	3.52%	3.70%
Proposed RF + SW-LSTM			
Overall	0.7904	0.7833	0.7907
Improvement over RF (Saeed et al. 2021)	8.01%	8.21%	8.05%
Improvement over SW-LSTM	4.44%	6.71%	5.60%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	4.04%	4.74%	4.08%
Random-looking DGAs	0.7487	0.7382	0.7467
Improvement over RF (Saeed et al. 2021)	0.44%	0.30%	0.36%
Improvement over SW-LSTM	6.49%	9.67%	7.89%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	4.15%	5.07%	4.16%
Word-looking DGAs	0.9409	0.9409	0.9409
Improvement over RF (Saeed et al. 2021)	38.73%	37.96%	35.77%
Improvement over SW-LSTM	− 0.68%	− 0.29%	0%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	4.79%	4.43%	3.90%

A comparison was made with random forest (RF), SW-CNN, SW-LSTM, and CNN-BiLSTM (Cucchiarelli et al. 2021)

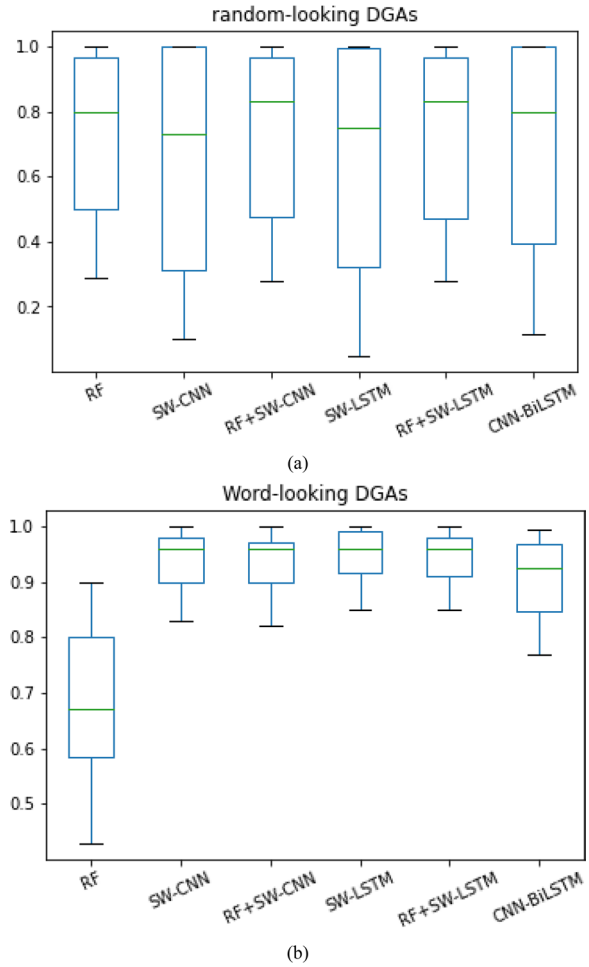

Fig. 5 The box plots of the F1 score for **a** the random-looking DGAs and **b** the word-looking DGAs

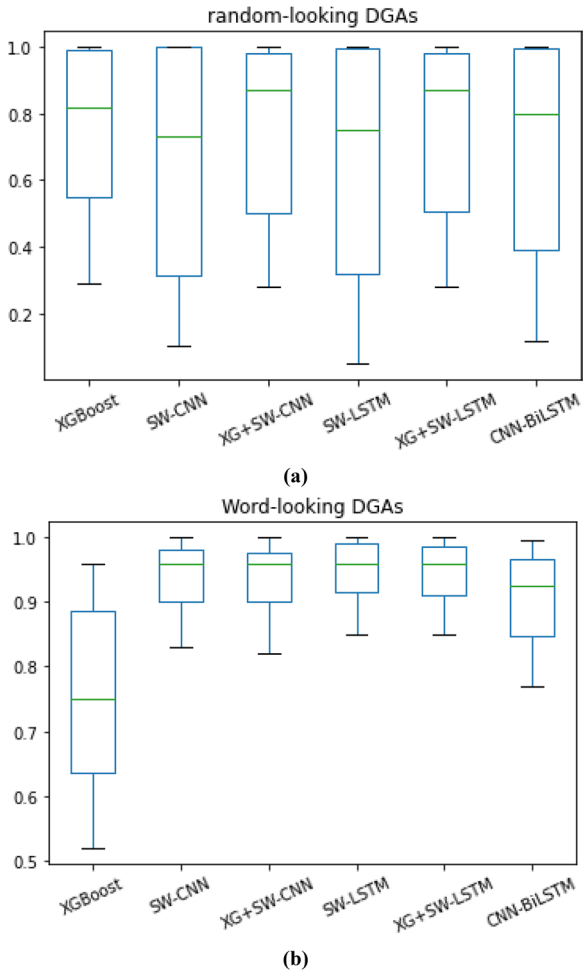

Fig. 6 The box plots of the F1 score for **a** the random-looking DGAs and **b** the word-looking DGAs

In the CNN case, the improvement of the integrated scheme over the RF, SW-CNN, and CNN-BiLSTM are 7.86%, 4.01%, and 3.90% respectively. For LSTM, the improvements are 8.01%, 4.44%, and 4.04% respectively. We also examine the performance of the random-looking and word-looking DGAs. We can see that the integrated approach is able to produce a good classification for both DGA types. For the word-looking DGAs, the integrated approach has significant improvement over the random forest approach as the word tokenization can better characterize the word relationship. For the random-looking DGAs, the performance of the integrated scheme matches that of the feature-extracting ML approach. As shown in the boxplot in Fig. 5, some random-looking DGA classes have bad classification performance in the SW-CNN and SW-LSTM. However, the integrated approach can improve these classes because our integrated scheme chooses the ML approach for classification. Thus, the overall classification performance of the random-looking DGAs in the integrated approach is better than the SW-CNN and SW-LSTM.

Integrated scheme with XGBoost

In this sub-section, we consider integrating the XGBoost classifier with the proposed SW-CNN and SW-LSTM models. Table 7 shows the classification results and Fig. 6 shows the boxplot. Similar to the case of the random forest, the integrated scheme performs better than the individual classifiers as well as the CNN-BiLSTM which uses character tokenization. By comparing the integrated scheme of the random forest (Table 6) and the XGBoost (Table 7), the performance of the XGBoost is slightly better than that of the random forest.

Table 7 A summary of the performance of the proposed integrated schemes XG + SW-CNN and XG + SW-LSTM

	Average precision	Average F1	Average recall
Proposed XG + SW-CNN			
Overall	0.8016	0.7950	0.7997
Improvement over XGBoost	5.13%	4.98%	5.07%
Improvement over SW-CNN	5.62%	8.47%	6.88%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	5.52%	6.31%	5.27%
Random-looking DGAs	0.7641	0.7554	0.7605
Improvement over XGBoost	− 0.25%	− 0.54%	− 0.51%
Improvement over SW-CNN	8.24%	12.28	9.72%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	6.29%	7.51%	6.08%
Word-looking DGAs	0.9300	0.9327	0.9391
Improvement over XGBoost	25.07%	24.48%	24.45%
Improvement over SW-CNN	− 0.40%	− 0.30%	0%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	3.58%	3.52%	3.70%
Proposed XG + SW-LSTM			
Overall	0.8023	0.7970	0.8010
Improvement over XGBoost	5.22%	5.24%	5.24%
Improvement over SW-LSTM	6.01%	8.58%	6.98%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	5.61%	6.57%	5.44%
Random-looking DGAs	0.7636	0.7559	0.7603
Improvement over XGBoost	− 0.31%	− 0.47%	− 0.54%
Improvement over SW-LSTM	8.60%	12.30%	9.85%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	6.22%	7.59%	6.05%
Word-looking DGAs	0.9427	0.9418	0.9409
Improve over XGBoost	26.54%	25.57%	24.69%
Improvement over SW-LSTM	− 0.49%	− 0.19%	0%
Improvement over CNN-BiLSTM (Cucchiarelli et al. 2021)	4.99%	4.53%	3.90%

A comparison was made with XGBoost, SW-CNN, SW-LSTM, and CNN-BiLSTM (Cucchiarelli et al. 2021)

Conclusions

In this study, we have performed a comprehensive analysis of existing state-of-the-art methods in algorithmically generated domain name classification. The existing methods have limited performance in characterizing word-looking domain names due to the use of character tokenization. Hence, subword tokenization is proposed for characterizing these domain names. Experimental results have proved that the subword tokenization in the proposed CNN and LSTM models significantly improves the classification performance of these word-looking domain names as compared to their character counterparts. As domain names contain both word-looking and random-looking types, an integrated scheme that combines the advantages of feature extracting machine learning classifier and the subword-based deep learning model is proposed. Our experimental results demonstrate the advantages of the integrated scheme and it outperforms both the machine learning-based classifiers and deep learning classifiers.

Acknowledgements
The authors would like to thank the reviewers for their constructive comments that have greatly improved the paper.

Author contributions
Both authors contributed to the design and implementation of the research, to the analysis of the results, and to the writing of the manuscript.

Funding
N/A.

Availability of data and materials
All datasets used are public datasets.

Declarations

Competing interests
The authors declared that they have no competing interests.

Received: 30 May 2023 Accepted: 6 August 2023
Published online: 07 September 2023

References

Almashhadani AO, Kaiiali M, Carlin D, Sezer S (2020) MaldomDetector: a system for detecting algorithmically generated domain names with machine learning. Comput Secur 93:101787

Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, Dagon D (2012) From throw-away traffic to bots: detecting the rise of DGA-based malware. In: USENIX security symposium, p 24

Berman DS (2019) DGA CapsNet: 1D application of capsule networks to DGA detection. Information 10:157

Bilge L, Şen S, Balzarotti D, Kirda E, Krügel C (2014) Exposure: a passive DNS analysis service to detect and report malicious domains. ACM Trans Inf Syst Secur 16:14

Cucchiarelli A, Morbidoni C, Spalazzi L, Baldi M (2021) Algorithmically generated malicious domain names detection based on n-grams features. Expert Syst Appl 170:114551

Feng Z, Shuo C, Xiaochuan W (2017) Classification for DGA-based malicious domain names with deep learning architectures. Int J Intell Inf Syst 6(6):67–71

Hoang XD, Vu XH (2022) An improved model for detecting DGA botnets using random forest algorithm. Inf Secur J Glob Perspect 31(4):441–450

Kambourakis G, Anagnostopoulos M, Meng W, Zhou P (2019) Botnets: architectures, countermeasures and challenges. CRC Press

Liew SRC, Law NF (2022) BEAM—an algorithm for detecting phishing link. APSIPA ASC

Mac H, Tran D, Tong V, Nguyen LG, Tran HA (2017) DGA botnet detection using supervised learning methods. In: Proceedings of the eighth international symposium on information and communication technology, pp 211–218

Negash N, Che X (2015) An overview of modern botnets. Inf Secur J Glob Perspect 24(4–6):127–132

Qiao Y, Zhang B, Zhang W, Sangaiah AK, Wu H (2019) DGA domain name classification method based on long short-term memory with attention mechanism. Appl Sci 9:4205

Ren F, Jiang Z, Wang X, Liu J (2020) A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. Cybersecurity 3:1–13

Saeed AMH, Wang D, Alnedhari HAM, Mei K, Wang J (2022) A survey of machine learning and deep learning based DGA detection techniques. In: Qiu M, Gai K, Qiu H (eds) Smart computing and communications. SmartCom 2021. Lecture notes in computer science, vol 13202

Selvi JP, Rodríguez RJ, Soria-Olivas E (2021) Toward optimal LSTM neural networks for detecting algorithmically generated domain names. IEEE Access 9:126446–126456

Vij P, Nikam SD, Bhatia A (2020) Detection of algorithmically generated domain names using LSTM. In: International conference on communication systems and networks (COMSNETS), pp 1–6

Vormayr G, Zseby T, Fabini J (2017) Botnet communication patterns. IEEE Commun Surv Tutor 19:2768–2796

Vranken HP, Alizadeh H (2022) Detection of DGA-generated domain names with TF-IDF. Electronics 11:414

Wang Z, Guo Y (2021) Neural networks based domain name generation. J Inf Secur Appl 61:102948

Wang T, Chen L, Genc Y (2020) A dictionary-based method for detecting machine-generated domains. Inf Secur J Glob Perspect 30(4):205–218

Wang Z, Guo Y, Montgomery D (2022) Machine learning-based algorithmically generated domain detection. Comput Electr Eng 100:107841

Woodbridge J, Anderson H, Ahuja A, Grant D (2016) Predicting domain generation algorithms with long short-term memory networks. arXiv: abs/1611.00791

Xu L, Magar R, Farimani AB (2022) Forecasting COVID-19 new cases using deep learning methods. Comput Biol Med 144:105342

Yang C, Lu T, Yan S, Zhang J, Yu K (2022) N-trans: parallel detection algorithm for DGA domain names. Future Internet 14:209

Yu B, Gray DL, Pan J, Cock MD, Nascimento AC (2017) Inline DGA detection with deep networks. In: 2017 IEEE international conference on data mining workshops (ICDMW), pp 683–692

Zago M, Pérez MG, Pérez GM (2020a) UMUDGA: a dataset for profiling DGA-based botnet. Comput Secur 92:101719

Zago M, Gil Pérez M, Martínez Pérez G (2020b) UMUDGA: a dataset for profiling algorithmically generated domain names in botnet detection. Data Brief 30:105400

Publisher's Note