axioms

MDPI

Article

A Unified Learning Approach for Malicious Domain Name Detection

Atif Ali Wagan [1], Qianmu Li [1,*], Zubair Zaland [2], Shah Marjan [2], Dadan Khan Bozdar [3], Aamir Hussain [4], Aamir Mehmood Mirza [3] and Mehmood Baryalai [3]

[1] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[2] Department of Software Engineering, Balochistan University of Information Technology Engineering and Management Sciences, Quetta 87300, Pakistan
[3] Department of Computer Science, Balochistan University of Information Technology Engineering and Management Sciences, Quetta 87300, Pakistan
[4] Department of Computer Science, Muhammad Nawaz Shareef University of Agriculture Multan, Multan 60000, Pakistan
* Correspondence: qianmu@njust.edu.cn

Abstract: The DNS firewall plays an important role in network security. It is based on a list of known malicious domain names, and, based on these lists, the firewall blocks communication with these domain names. However, DNS firewalls can only block known malicious domain names, excluding communication with unknown malicious domain names. Prior research has found that machine learning techniques are effective for detecting unknown malicious domain names. However, those methods have limited capabilities to learn from both textual and numerical data. To solve this issue, we present a novel unified learning approach that uses both numerical and textual features of the domain name to classify whether a domain name pair is malicious or not. The experiments were conducted on a benchmark domain names dataset consisting of 90,000 domain names. The experimental results show that the proposed approach performs significantly better than the six comparative methods in terms of accuracy, precision, recall, and F1-Score.

Keywords: DNS; firewall; malicious; domain; name

Citation: Wagan, A.A.; Li, Q.; Zaland, Z.; Marjan, S.; Bozdar, D.K.; Hussain, A.; Mirza, A.M.; Baryalai, M. A Unified Learning Approach for Malicious Domain Name Detection. *Axioms* **2023**, *12*, 458. https://doi.org/10.3390/axioms12050458

Academic Editors: Oscar Humberto Montiel Ross, Hsien-Chung Wu and Darjan Karabašević

Received: 23 November 2022
Revised: 10 April 2023
Accepted: 24 April 2023
Published: 9 May 2023

1. Introduction

Preserving the integrity of network security is an important consideration for all organizations. As nearly every aspect of business becomes increasingly digital, enterprise network security software can help organizations mitigate the effects of cyber-attacks, particularly by protecting against them, thereby safeguarding their operations and ensuring their competitiveness in a rapidly changing marketplace.

Intrusion detection systems (IDS) and intrusion prevention systems (IPS) have long been part of the network security toolkit to detect, monitor, and block malware and malicious traffic. IDS are designed to detect intrusions into the network and issue warnings when they detect a potential cyber-attack. However, the system itself cannot protect against attacks, and that responsibility is left to human analysts. Meanwhile, IPS work proactively to prevent successful attacks and respond according to predefined rules when an intrusion is detected. One of the most important IPS are firewalls [1]. One such firewall for IP is the DNS firewall. The majority of DNS firewalls are built on regularly updated lists of known malicious domain names. However, this method can only block known malicious communications, leaving a large number of malicious communications that cannot be blocked because they are unknown.

To predict new malicious domains over DNS communications in a timely manner with better DNS data features is important [2]. To resolve this challenge, Marques et al. [2]

proposed a machine learning-based DNS firewall built with new domain names and 34 features. However, they utilized traditional classification models for their proposed approach which have limited capabilities to learn from both numerical and textual features of the domain name at the same time.

To resolve this issue, we propose a unified deep learning model which can learn from both numerical and textual features. This model makes use of both numerical and textual information in the domain name. These pairs are entered into the model and correlating features between the pairs are extracted. Finally, the model uses the correlating features to determine if these pairs represent a malicious domain or not.

Our work makes the following three major contributions:

(1) We consider the domain name textual and numerical features unified representation learning issue in the context of malicious domain name detection tasks.
(2) A new deep learning model is proposed for malicious domain name detection.
(3) Experimental results for the detection of the malicious domain name demonstrate the high performance of the proposed method in comparison with state-of-the-art methods on the malicious domain name dataset.

The remainder of this article is organized as follows: Section 2 describes previous research on malicious domain name detection. The details of our unified learning approach are described in Section 3. In Section 4, the experimental setup is described, followed by experimental results in Section 5. Finally, Section 6 concludes this article.

2. Related Work

A number of approaches for malicious domain name detection have been proposed in the literature. In general, these approaches can be broadly classified into non-machine learning-based and machine learning-based approaches.

2.1. Non-Machine Learning Approaches

The non-machine learning-based approaches use some kind of data obtained from known malicious domain names for making malicious domain name signatures and using those signatures to filter unknown malicious domain names.

Zhang et al. [3] proposed a blocklisting system, the purpose of which is to create blocklists based on the relevance ranking scheme used by the link analysis community. The system creates blocklists for individuals who choose to submit data to a central log-sharing platform. The system evaluates the relevance of the submitter to the attacker based on the attacker's history and the submitter's recent log generation patterns. The blocklist system also incorporates extensive log prefiltering and severity metrics to understand the extent to which attacker alert patterns are consistent with common malware propagation behaviors. Prakash et al. [4] proposed PhishNet, which is based upon two components. The first component uses five heuristics to find new phishing URLs. The second component is made of an approximate matching algorithm that partitions the URL into several components, then these multiple components of the URL are matched with entries in the blocklist individually. Malicious URLs are often short-lived and are updated frequently to avoid being on the blocklist. If these malicious URLs are updated by the same adversary using domain name string manipulation, we can assume that unknown malicious URLs exist in the neighborhood of known malicious URLs. Based on this assumption, Akiyama et al. [5] proposed an effective blocklist URL generation method that uses search engines to discover unknown malicious URLs in the neighborhood of known malicious URLs. Fukushima et al. [6] proposed a blocklisting scheme in which they analyzed malicious URLs' metadata, such as domain, registrar, IP address, IP address block, and autonomous system. Furthermore, they evaluated registrars and IP address blocks associated with malicious URLs. From that evaluation, they made a blocklist with low-reputation IP address blocks and registrars often used in malicious URLs. Sun et al. [7] proposed an automatic blocklist generator (AutoBLG) that discovers new malicious URLs automatically by starting from an existing URL blocklist. The idea of their approach is to extend the space of web page searches, while

decreasing the number of URLs to be analyzed, by applying some prefilters to expedite the process of creating a blocklist.

Unlike the above works, which use signatures for blocking malicious domain name, we propose a machine learning-based approach that learns from malicious and benign domain name data and classifies whether a given domain name is malicious or not.

2.2. Machine Learning Approaches

Machine learning-based approaches make a predictive model for detecting unknown malicious domain names by training on data of malicious and benign domain names.

Saxe and Berlin. [8] used character-level embedding with a convolutional neural network to retrieve associated features from file paths, registry keys, and URLs. The retrieved features are then fed into three fully connected layers to classify file paths, registry keys, and URLs as malicious or normal. Yang et al. [9] proposed a convolutional gated recurrent unit neural network for detecting malicious URLs using characters as text classification features. Their model, instead of using a pooling layer in a convolutional neural network, uses a gated recurrent unit for feature acquisition in the time dimension. Luo et al. [10] proposed an approach for identifying malicious HTTP GET requests using a novel architecture of convolutional neural network for classification; they used natural language processing-based analysis and Auto-Encoder for URL representation and extraction. Mondal et al. [11] proposed an ensemble learning approach to predict the class probabilities of benign or malicious URLs using multiple classifiers. After obtaining probabilities from multiple classifiers, a threshold filter is applied to the probabilities to finally determine whether the URL is malicious or not. Marques et al. [2] empirically investigated the impact of three feature selection methods applied to six classification models on the performance of malicious domain name detection. They conducted experiments on the malicious domains dataset, and found that the decision trees with recursive feature elimination were more suitable for the malicious domain detection task, compared with other baseline methods.

Unlike the above works, we simultaneously consider both numerical and textual feature unified learning for malicious domain name detection.

3. Methodology

3.1. Overview

In order to utilize both numerical features and textual features for detecting malicious domain names, we propose a combined deep learning-based model (hybrid feed forward network (FFN)-long short-term memory (LSTM) [12] model that preserves the advantages of both numerical and textual features. The network structure of the hybrid FFN-LSTM model is shown in Figure 1. The original dataset was split into numerical and textual features that were fed into the model as separate inputs. Individual learners were then built based on numerical and textual features, and the outputs of the individual learners were spliced together in parallel to form the input to a unified learner consisting of a fully connected neural network.

3.2. Numerical Learner

Following Marques et al. [2], we performed the same two pre-processing steps—the label encoder [13] and min-max normalization [14]—as them. The label encoder [13] transforms categorical features by simply mapping each category with an integer value ranging from 0 to n. After the label encoding [13], min-max normalization [14] was applied to scale the different features so that each feature has equal weight for the numerical feature learner. This sets the value of the data points from 0 to 1.

Pre-processed numerical features are sent to the FFN for learning numerical features. The FFN is composed of three types of network layers: an input layer, a hidden layer, and an output layer. Additionally, each of these layers contains n number of neurons. The first layer with n neurons is the input layer, and the input feature vectors are received in this layer. The hidden layer consists of a few layers. In the last output layer, the FFN outputs

the results. We constructed the FFN with a depth of 5 layers. We denote these layers as $\delta = \{\mathcal{L}_n^1, \mathcal{L}_n^2, \mathcal{L}_n^3, \mathcal{L}_n^4, \mathcal{L}_n^5\}$.

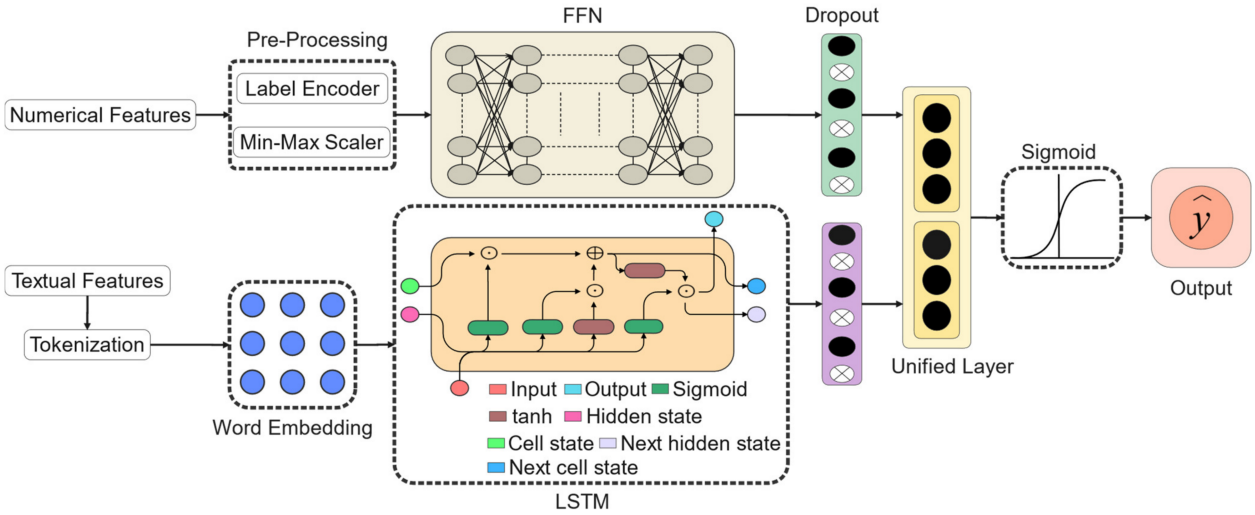

Figure 1. Our proposed malicious domain name detection model.

The FFN output layer output is forwarded to the dropout [15] layer to prevent overfitting. Owing to over-fitting, the classification ability of the FFN is limited. Dropout [15] can effectively solve this problem. Dropout [15], is a mechanism to improve the performance of the FFN by randomly setting the weights of the FFN to 0.

3.3. Text Learner

To construct an accurate malicious domain name detection model, the feature vector representation of textual features is important. Although high-level representations such as numerical features can be useful, they cannot reveal deep hidden semantics in the textual features of the domain name. LSTM [12] is a deep learning architecture, which provides a powerful representation of the textual features. It can learn semantic features automatically. Before LSTM [12] could process textual features, we needed to pre-process textual features with tokenization and represent them as an embedding matrix.

In the tokenization phase, the sequence of words is broken into multiple tokens based on the white space separator character. In this process, each word is called a token.

The embedding layer takes a token index as input and transforms it into a low-dimensional representation. Given a textual features sample tokens n which is essentially a series of words $\left[\mathfrak{f}_1, \ldots, \mathfrak{f}_{|n|}\right]$, our goal is to obtain its matrix representation $n \to \mathcal{N} \in \mathbb{R}^{|n| \times d_m}$, where $\mathcal{N}$ is a matrix consisting of a set of words $\mathfrak{f}_i \to \mathfrak{F}_i, i = 1, \ldots, |n|$ in the given domain name sample. Every word $\mathfrak{f}_i$ can now be represented as an embedding vector, i.e., $\mathfrak{F}_i \in \mathbb{R}^{d_m}$, where d_m is the dimensional vector of words that appear in the textual features of the domain name. In our experiments, we randomly initialize the embedding matrix of malicious domain names textual features, and during the training process it is learned. Thus, domain name textual features matrix representation $\mathcal{N}$ having $|n|$ words sequence can be expressed as follows:

$$\mathcal{N} = \left[\mathfrak{F}_i, \ldots, \mathfrak{F}_{|n|}\right] \tag{1}$$

All domain name textual features are truncated or padded to be the same length $|n|$ for parallelization.

Lower dimensional embedding vectors are sent to an LSTM [12] network which can be regarded as a sequence of LSTM [12] units for feature learning. Let $\mathfrak{F}_1, \ldots, \mathfrak{F}_{|n|}$ be the entered words sequence (e.g., domain name textual features tokens), with having a corresponding labels sequence (e.g., the next domain name textual feature tokens). At

every step t, the LSTM [12] unit takes the $\mathfrak{F}_t$ input, along with the previous hidden and cell state, and computes the next hidden state and cell state using a set of model parameters. The output is predicted using the output state (e.g., the next domain name textual feature token based upon previous ones) at each step t.

Similar to the numerical learned features, the textual learned features using LSTM [12] are fed to the dropout [15] layer to prevent overfitting.

3.4. Unified Learning

The highlight of our method is that we utilize textual and numerical features and automatically determine the optimal ratio of each feature at the fusion stage. We feed the outputs of numerical learner $\mathcal{Q}_i$ and text learner $\mathcal{T}_i$ into a new unified vector $\mathfrak{U}$ that represents both textual and numerical features in a given sample, which can be formulated as follows:

$$\mathfrak{U} = \mathcal{Q}_i \oplus \mathcal{T}_i \tag{2}$$

The new unified vector is then fed into the fully connected layer, resulting in a vector output γ as follows:

$$\gamma = \sigma(W_\gamma \cdot \mathfrak{U} + b_\gamma) \tag{3}$$

where $\cdot$ represent the dot product, the vector γ weight matrix is represented as W_γ, the bias value of the vector γ is represented as b_γ, and the activation function ReLU [16] is represented as $\sigma(\cdot)$. Finally, the γ vector is sent to an output layer.

Each numerical and textual feature is properly represented into a feature vector by previous layers. Using these feature vectors as input, a unified binary classifier is used in the output layer to predict whether these features represent a malicious domain or not. In this study, a sigmoid classifier was used, which computes a probability score for a given domain name sample as follows:

$$\hat{\mathfrak{P}}_i = sigmoid(\gamma) \tag{4}$$

3.5. Training

During the process of training, our model tries to learn the following parameters: the five dense layers of FNN, the word embedding matrices of textual features, the LSTM [12] layer, and the weights and bias of the fully connected output layer. After these parameters are learned, the malicious domain name can be detected. These model parameters are learned by minimizing the following binary cross-entropy loss function:

$$BCE = -\frac{1}{\mathfrak{N}} \sum_{i=0}^{\mathfrak{N}} \mathfrak{G}_i \cdot \log(\hat{\mathfrak{P}}_i) + (1 - \mathfrak{G}_i) \cdot \log(1 - \hat{\mathfrak{P}}_i) \tag{5}$$

where all samples in the dataset are denoted as $\mathfrak{N}$, and the output layer probability score is denoted as $\hat{\mathfrak{P}}_i$, defined by Equation (4), $\mathfrak{G}_i = \{0, 1\}$ indicates whether the i-th sample is a malicious domain or not. As it has previously been shown that Adam [17] is less memory intensive and more computationally efficient than other optimization methods, we decided to minimize the objective function using the Adam [17] optimizer. In order to efficiently calculate parameter updates during the learning process, we use backpropagation [18], a simple implementation of the chain rule of partial derivatives.

4. Experiment Setup

4.1. Dataset

We use a public dataset in this study that has been collected and processed by Marques et al. [19]. The dataset contains approximately 90,000 malicious and non-malicious domain name samples of equal size. Each sample contains 34 features, such as IP, geolocation, open ports, domain name entropy, etc.

4.2. Performance Evaluation

There are four possible outcomes of our model prediction: true positive (TP) indicates that a malicious domain name is predicted as a malicious domain name, true negative (TN) indicates that a non-malicious domain name is predicted as a non-malicious domain name, false negative (FN) indicates that a malicious domain name is predicted as a non-malicious domain name, and false positive (FP) indicates that a non-malicious domain name is predicted as a malicious domain name. Based on these outcomes, the performance evaluation metrics accuracy, recall, precision, and F1-Score can be calculated as given below.

Accuracy is perhaps the most intuitive way of measuring the performance of any binary classification model. The accuracy metric can be interpreted as the percentage of samples correctly classified by the model. Based on the notation introduced earlier, the accuracy is defined in Equation 6 as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

As an alternative measure of classifier performance, precision can be interpreted as the accuracy of positive predictions. Precision is defined by the following equation:

$$\text{Precision } = \frac{TP}{TP + FP} \tag{7}$$

Precision is often used in conjunction with a metric called recall, as precision measurement tends to look very high for models that predict few positives. Recall indicates the proportion of positive instances that are correctly detected by the classifier, as defined by the following equation:

$$\text{Recall } = \frac{TP}{TP + FN} \tag{8}$$

The F1-Score is the harmonic mean of precision and recall, as defined by Equation (9). A large F1-Score can only be obtained if both recall and precision are high.

$$\text{F1} - \text{Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

5. Results

RQ.1 What is the effectiveness of our approach against the state-of-the-art baseline methods?

Our goal is to provide a method that can automatically classify malicious and non-malicious domain names. However, one challenge to the usefulness of this approach is to determine how much its performance is improved over the baseline approaches. By answering this research question, it would become clear how far ahead our method is in detecting malicious domain names compared to state-of-the-art methods. To compare the performance of our method, we chose six baseline methods, which are listed below.

Linear discriminant analysis (LDA) [20–23]: The purpose of the discriminant analysis is to find the linear function of the data that best separates the two data points. Each data point is classified into one of its two groups. When considering the data, the between-group variance is maximized, and the within-group variance is minimized. In LDA [20–23], it is assumed that the data can be represented linearly. However, this often does not reflect realistic relationships between the data, and discriminant analysis is limited in its ability to best classify data points. In simple terms, LDA [20–23] attempts to find a separable subspace of data points whose dimensionality is lower than that of the original data sample. This is achieved by finding a hyperplane that maximizes the mean and minimizes the variance between the two classes.

Support vector machine (SVM) [24]: The SVM [24] method works by finding the hyperplane that divides feature space between two regions. In this way, on one side of the plane are all data points belonging to class one, and all points that belong to class two are

on the opposite part of the plane. The aim is to maximize the margins so that the optimal linearly separable hyperplane can be determined among multiple alternatives. Margin optimization means maximizing the distance between the closest data points (also called support vectors) on either side of the hyperplane. This results in a hyperplane that is as far away as possible from the closest data points (support vectors). In other words, the support vectors can be viewed as a hyperplane parallel to the main hyperplane, helping to create the most efficient hyperplane.

K-nearest neighbor (KNN) [25–27]: To classify a point in the feature space, KNN [25–27] examines the k nearest neighbors of the point and makes predictions based on majority voting from those nearest neighbors. The input variable k is used to determine the number of nearest neighbors to use. This k number is usually an odd number, to avoid situations where new data points are equally distributed in two classes. KNN [25–27], as its name implies, calculates the distance between different points in the feature space. In some situations, it may be useful to weigh the "votes" of adjacent points according to distance, so that nearby points contribute more to the classification than distant points.

Logistic regression (LR) [28]: LR [28] uses optimization methods such as gradient ascent or the more efficient stochastic gradient ascent to find the optimal parameters of a nonlinear function, called a sigmoid, which is very suitable for binary problems. The advantage of using stochastic gradient ascent as an optimization algorithm is that it can learn from new data by several batches and iterations.

Naive Bayes (NB) [29]: The NB [29] is a classifier that is based upon Bayes' theorem of statistical probability. It assumes every single predictor is equally important and independent of one another. That is, given a class variable, it assumes that the absence or presence of a particular feature is independent of the absence or presence of other features. Rather than a simple classification, the NB [29] will report probabilities of an instance belonging to an individual class. The class with the highest posterior probability in our case is the prediction of whether the domain name is malicious or not.

Decision tree (DT) [30]: DTs [30] are essentially a set of questions designed to arrive at a classification decision. As its name suggests, a DT [30] is a tree-like structure, which is composed of several parts, such as root nodes, internal nodes, leaf nodes, and branches. The root node represents the top-level decision node. In other words, it is where the classification tree begins to be traversed. A leaf node is a node that is not split into more nodes; it is where a class is assigned by majority vote. DTs [30] are constructed via an algorithmic approach that makes a classification decision for a given data point by recursively partitioning the available data. In other words, the algorithm partitions the data recursively into subsets with rules that maximize information gain.

Figures 2–5 show the results of our approach in comparison with other baseline approaches in terms of precision, recall, F1-Score, and accuracy. In terms of precision, our approach obtains 0.989%.

The average precision value improvement by our approach over all baseline approaches is 0.051%, 0.049%, 0.019%, 0.04%, 0.034%, and 0.016% compared with LDA [2], [20–23], SVM [2,24], KNN [2,25–27], LR [2,28], NB [2,29], and DT [2,30], respectively. In terms of recall, our approach obtains 0.988%. The average recall value improvement by our approach over all baseline approaches is 0.119%, 0.093%, 0.045%, 0.11%, 0.134%, and 0.036% compared with LDA [2,31], SVM [2,24], KNN [2,25–27], LR [2,28], NB [2,29], and DT [2,30], respectively. In terms of F1-Score, our approach obtains 0.988%. The average F1-Score value improvement by our approach over all baseline approaches is 0.093%, 0.079%, 0.035%, 0.083%, 0.097%, and 0.029% compared with LDA [2,31], SVM [2,24], KNN [2,25–27], LR [2,28], NB [2,29], and DT [2,30], respectively. In terms of accuracy, our approach obtains 0.988%. The average accuracy value improvement by our approach over all baseline approaches is 0.081%, 0.069%, 0.031%, 0.072%, 0.08%, and 0.026% compared with LDA [2,31], SVM [2,24], KNN [2,25–27], LR [2,28], NB [2,29], and DT [2,30], respectively. All of these results indicate that the unified learning approach is more effective compared with the baseline approaches.

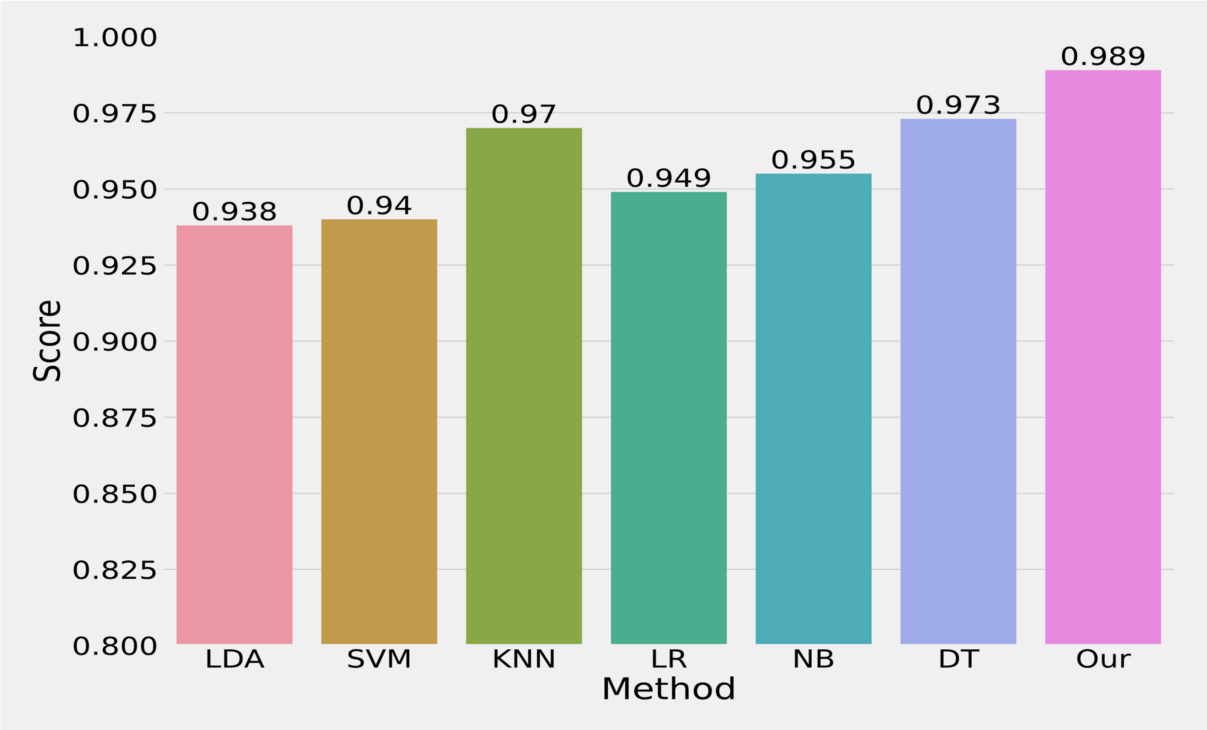

Figure 2. Comparison with baseline approaches on the precision metric.

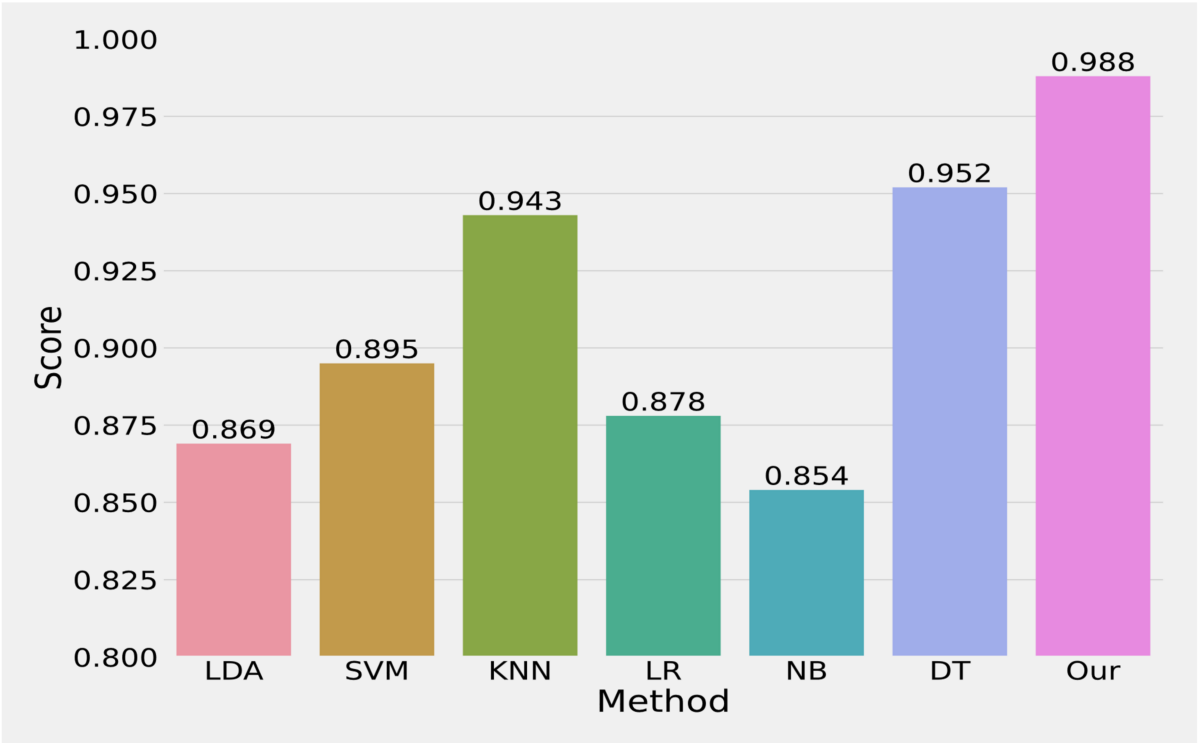

Figure 3. Comparison with baseline approaches on the recall metric.

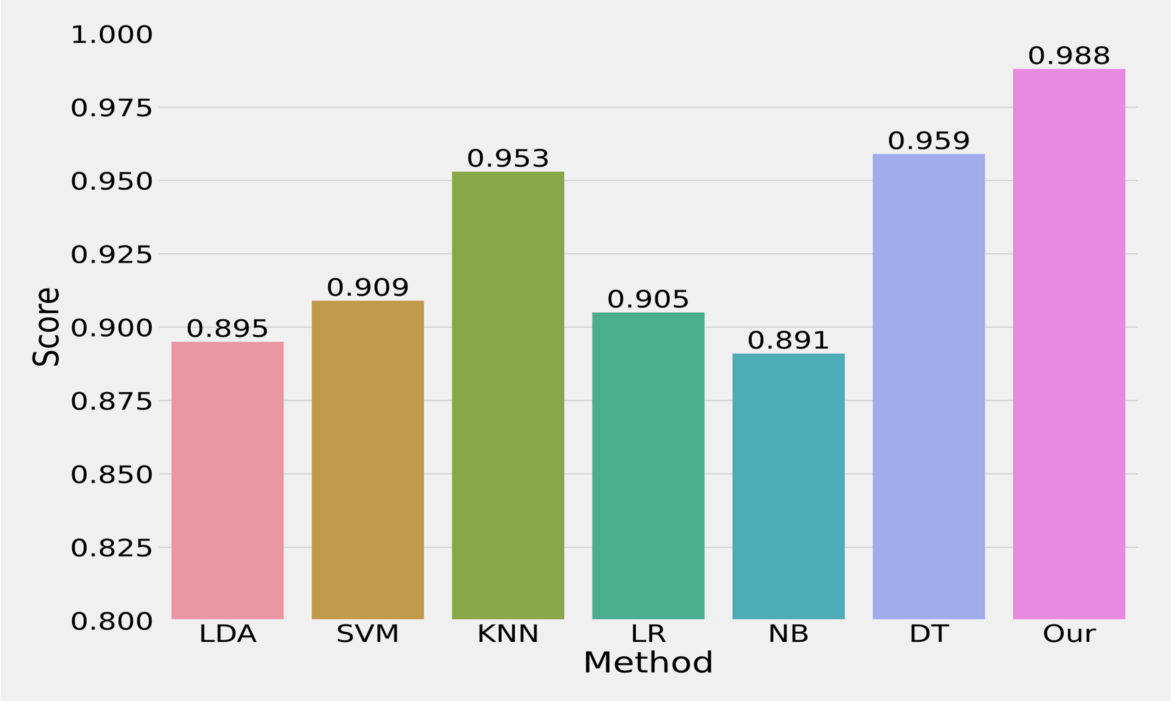

Figure 4. Comparison with baseline approaches on the F1-Score metric.

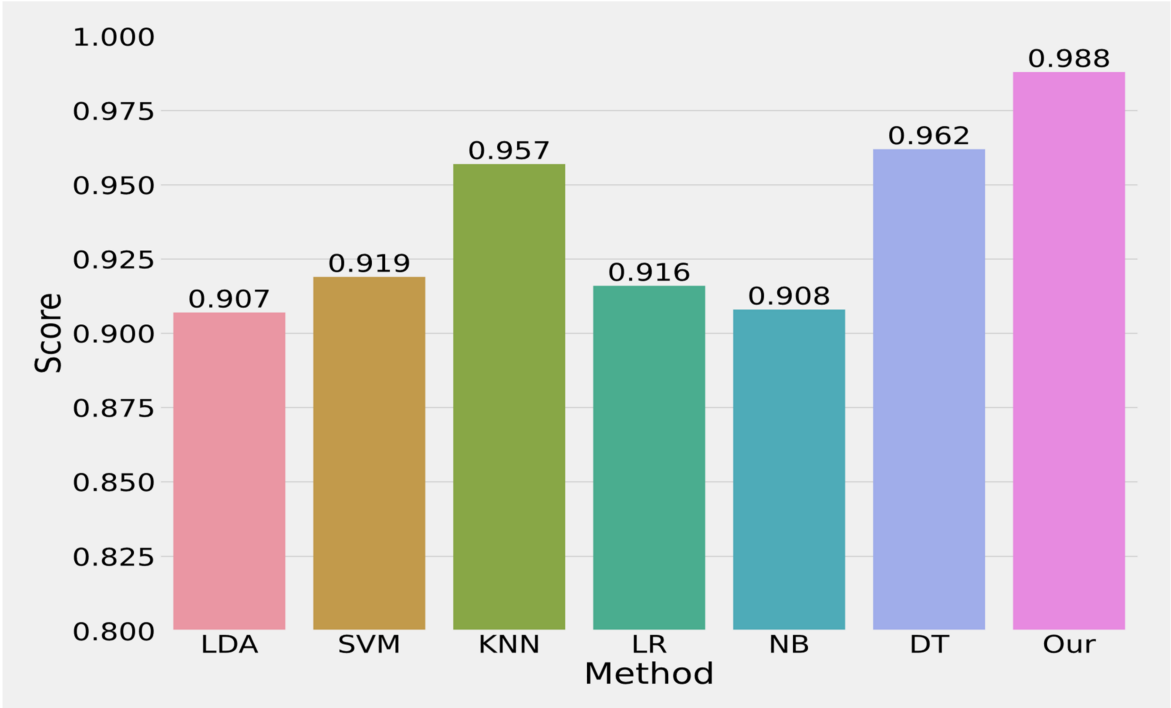

Figure 5. Comparison with baseline approaches on the accuracy metric.

RQ.2 How effective is our unified learning approach in comparison with individual features learning?

Numerical or textual features each provide separate useful information to distinguish whether the domain name is malicious or not. To investigate the numerical or textual features' impact on performance individually, we remove the numerical or textual fea-

tures from the input and its respective modules in the model, and perform comparative experiments while keeping other conditions unchanged.

We evaluate performance through the accuracy, recall, precision, and F1-Score, and Figure 6 shows the experiment results. From Figure 6 it can be observed that, after removing the numerical or textual features, the performance on all the metrics decreased. In terms of accuracy, the decrease is 0.159% and 0.194% using numerical and textual features, respectively. In terms of precision, the decrease is 0.023% and 0.248% using numerical and textual features, respectively. In terms of recall, the decrease is 0.307% and 0.067% using numerical and textual features, respectively. In terms of F1-Score, the decrease is 0.19% and 0.169% using numerical and textual features, respectively. We notice that, after removing numerical or textual features, although the performance has declined, this decline does not reach 0%. The reason for this might be that numerical or textual features only account for a small fraction of the total features used to accurately detect malicious domain name. All of these results indicate that the unified learning approach is able to fully exploit its ability to capture local numerical and textual features and extract deep relationships between them. This shows that it is useful to feed numerical features along with textual features to the model for the accurate detection of malicious domain names.

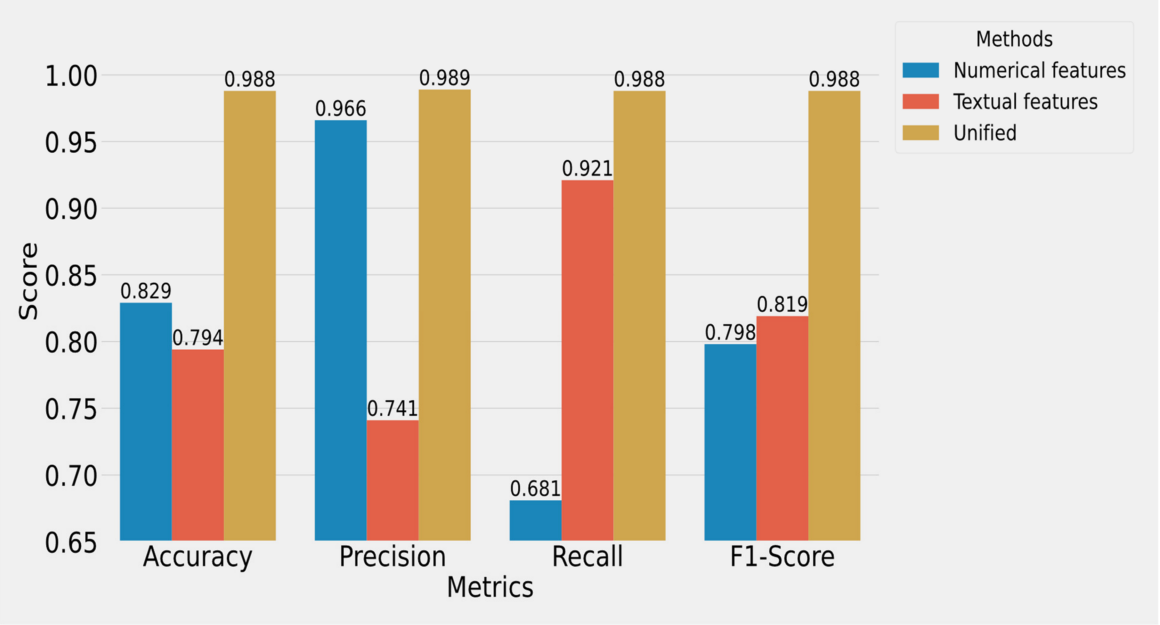

Figure 6. Individual and unified features comparison.

6. Conclusions

In this article, we proposed a novel approach that utilizes numerical and textual features of a domain name for malicious domain name detection. These domain name pairs are fed into a deep learning model that captures the semantic relationship between numerical and textual features. Then, these association features are used to classify whether a domain name pair is a malicious domain or not. We investigated the performance of our approach using a public malicious domain name dataset. We evaluated our approach's performance against the state-of-the-art machine learning-based approaches for malicious domain detection, and found our approach performs better for malicious domain name detection. We made a comparative study by removing either numerical or textual features, and performed experiments without changing other conditions. We found that it is more effective to feed both numerical and textual features to the deep learning model. This indicates that our proposed numerical and textual features pair representation is effective. In the future, we will investigate how our approach performs on industrial projects.

Author Contributions: Conceptualization, A.A.W.; Funding acquisition, Q.L.; Investigation, A.A.W.; Methodology, A.A.W.; Project administration, Q.L.; Software, A.A.W.; Supervision, Q.L.; Validation, S.M. and A.M.M.; Visualization, A.H. and M.B.; Writing—original draft, A.A.W.; Writing—review and editing, Z.Z. and D.K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by "Research on the Key Technology of Endogenous Security Switches" (2020YFB1804604) of the National Key R&D Program; "New Network Equipment Based on Independent Programmable Chips" (2020YFB1804600); the 2020 Industrial Internet Innovation and Development Project from Ministry of Industry and Information Technology of China; the Fundamental Research Fund for the Central Universities (30918012204, 30920041112); the 2019 Industrial Internet Innovation and Development Project from Ministry of Industry and Information Technology of China; Jiangsu Province Modern Education Technology Research Project (84365); National Vocational Education Teacher Enterprise Practice Base "Integration of Industry and Education" Special Project (Study on Evaluation Standard of Artificial Intelligence Vocational Skilled Level); Scientific research project of Nanjing Vocational University of Industry Technology (2020SKYJ03).

Data Availability Statement: A publicly available dataset was used in this study, which can be found at https://doi.org/10.17632/623sshkdrz.5 (accessed on 17 November 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, A.X. *Firewall Design and Analysis*; World Scientific: Singapore, 2010. [CrossRef]
2. Marques, C.; Malta, S.; Magalhães, J. DNS Firewall Based on Machine Learning. *Future Internet* **2021**, *13*, 309. [CrossRef]
3. Zhang, J.; Porras, P.; Ullrich, J. Highly predictive blacklisting. In Proceedings of the 17th Conference on Security Symposium, San Jose, CA, USA, 28 July–1 August 2018; USENIX Association: Berkeley, CA, USA, 2008; pp. 107–122.
4. Prakash, P.; Kumar, M.; Kompella, R.R.; Gupta, M. PhishNet: Predictive Blacklisting to Detect Phishing Attacks. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–5. [CrossRef]
5. Akiyama, M.; Yagi, T.; Itoh, M. Searching Structural Neighborhood of Malicious URLs to Improve Blacklisting. In Proceedings of the 2011 IEEE/IPSJ International Symposium on Applications and the Internet, Munich, Germany, 18–21 July 2011; pp. 1–10. [CrossRef]
6. Fukushima, Y.; Hori, Y.; Sakurai, K. Proactive Blacklisting for Malicious Web Sites by Reputation Evaluation Based on Domain and IP Address Registration. In Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, Changsha, China, 16–18 November 2011; pp. 352–361. [CrossRef]
7. Sun, B.; Akiyama, M.; Yagi, T.; Hatada, M.; Mori, T. Automating URL Blacklist Generation with Similarity Search Approach. *IEICE Trans. Inf. Syst.* **2016**, *E99.D*, 873–882. [CrossRef]
8. Saxe, J.; Berlin, K. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. *arXiv* **2017**, arXiv:1702.08568.
9. Yang, W.; Zuo, W.; Cui, B. Detecting Malicious URLs via a Keyword-Based Convolutional Gated-Recurrent-Unit Neural Network. *IEEE Access* **2019**, *7*, 29891–29900. [CrossRef]
10. Luo, C.; Su, S.; Sun, Y.; Tan, Q.; Han, M.; Tian, Z. A Convolution-Based System for Malicious URLs Detection. *Comput. Mater. Contin.* **2020**, *62*, 399–411. [CrossRef]
11. Mondal, D.K.; Singh, B.C.; Hu, H.; Biswas, S.; Alom, Z.; Azim, M.A. SeizeMaliciousURL: A novel learning approach to detect malicious URLs. *J. Inf. Secur. Appl.* **2021**, *62*, 102967. [CrossRef]
12. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
13. sklearn.preprocessing.LabelEncoder. Scikit-Learn. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (accessed on 16 December 2022).
14. sklearn.preprocessing.MinMaxScaler. Scikit-Learn. Available online: https://scikit-learn/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (accessed on 26 February 2022).
15. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
16. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; Omnipress: Madison, WI, USA, 2010; pp. 807–814.
17. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; Bengio, Y., LeCun, Y., Eds.; 2015. Available online: http://arxiv.org/abs/1412.6980 (accessed on 20 November 2022).
18. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
19. Marques, C.; Malta, S.; Magalhães, J.P. DNS dataset for malicious domains detection. *Data Brief* **2021**, *38*, 107342. [CrossRef] [PubMed]

20. Wayback Machine. 2022. Available online: https://web.archive.org/web/20220615132544/http://datajobstest.com/data-science-repo/LDA-Primer-[Balakrishnama-and-Ganapathiraju].pdf (accessed on 27 March 2023).
21. Lu, J.; Plataniotis, K.N.; Venetsanopoulos, A.N. Face recognition using LDA-based algorithms. *IEEE Trans. Neural Netw.* **2003**, *14*, 195–200. [CrossRef] [PubMed]
22. Fu, R.; Tian, Y.; Shi, P.; Bao, T. Automatic Detection of Epileptic Seizures in EEG Using Sparse CSP and Fisher Linear Discrimination Analysis Algorithm. *J. Med. Syst.* **2020**, *44*, 43. [CrossRef] [PubMed]
23. Elnasir, S.; Shamsuddin, S.M. Palm vein recognition based on 2D-discrete wavelet transform and linear discrimination analysis. *Int. J. Adv. Soft Comput. Appl.* **2014**, *6*, 43–59.
24. 1.4. Support Vector Machines. Scikit-Learn. Available online: https://scikit-learn/stable/modules/svm.html (accessed on 26 February 2022).
25. kNN Definition | DeepAI. 2022. Available online: https://web.archive.org/web/20220701054511/https://deepai.org/machine-learning-glossary-and-terms/kNN (accessed on 27 March 2023).
26. Hassanat, A.B.; Abbadi, M.A.; Altarawneh, G.A.; Alhasanat, A.A. Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach. *arXiv* **2014**, arXiv:1409.0919014.
27. Sklearn.Neighbors.KNeighborsClassifier—Scikit-Learn 1.2.2 Documentation. 2023. Available online: https://web.archive.org/web/20230315064604/https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (accessed on 27 March 2023).
28. Advantages and Disadvantages of Linear Regression. 2023. Available online: https://web.archive.org/web/20230111220233/https://iq.opengenus.org/advantages-and-disadvantages-of-linear-regression/ (accessed on 27 March 2023).
29. 1.9. Naive Bayes—Scikit-Learn 1.2.1 Documentation. 2023. Available online: https://web.archive.org/web/20230307185232/https://scikit-learn.org/stable/modules/naive_bayes.html (accessed on 27 March 2023).
30. 1.10. Decision Trees—Scikit-Learn 1.2.2 Documentation. 2023. Available online: https://web.archive.org/web/20230320174546/https://scikit-learn.org/stable/modules/tree.html (accessed on 27 March 2023).
31. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.