Vilnius University
Institute of Mathematics and Informatics
L I T H U A N I A

INFORMATICS ENGINEERING (07 T)

MACHINE LEARNING BASED OPEN SOURCE INTELLIGENCE INFORMATION EXTRACTION AND ANALYSIS METHODS

Paulius Vaitkevičius

October 2020

Technical Report DMSTI-DS-T007-20-08

Vilnius University Institute of Data Science and Digital Technologies,
Akademijos str. 4, Vilnius LT-08663, Lithuania
www.mii.lt

Contents

1 Foreword

Research results presented in this technical report *are directly related* to the research aim and object of the doctoral studies and future dissertation.

Research results presented in this technical report covers objective No 2 of the doctoral studies and future dissertation: replicating the results of state-of-the-art algorithms, is presented.

Research aim and object of the doctoral studies and future dissertation are introduced further in this section.

RESEARCH OBJECT, AIM, AND OBJECTIVES OF THE DOCTORAL STUDIES

Research object:

1. Machine Learning and Deep Machine Learning algorithms for phishing websites detection.
2. Adversarial Machine Learning algorithms.

Research aim: The research aim is to develop a new method for effective and reliable phishing websites detection, based on Deep Neural Networks and Adversarial Machine Learning algorithms.

Research objectives:

1. Performing literature review, analyzing *state-of-the-art* algorithms for phishing website detection.
2. Replicating the results of *state-of-the-art* algorithms.
3. Proposing new and more effective method for phishing website detection.
4. Creating datasets for new experiments.
5. Conducting experimental research comparing the proposed method with state-of-the-art algorithms.

2 Introduction

Phishing is a cybercrime built on social engineering which employs technical trickery and masking as a trustworthy entity to steal sensitive user data, such as passwords, credit card details, digital identity data from unsuspecting users. Phishers usually lure victims into entering a fraudulent website by impersonating legitimate URL and sending it to the victim by email or SMS along with a threatening message, like notice of account termination or illegal transaction [14]. Phishing attacks are still very successful nowadays, despite many existing anti-phishing solutions. The Anti-Phishing Working Group (APWG) reported as many as 2,172 unique phishing websites detected per day in 2019 with a 71.0% increase during the last six years of monitoring [2]. Global losses from phishing activities exceeded 3.5 billion USD in 2019, with a 29.6% annual increase and total global losses of 10.2 billion USD in the last five years, according to the FBI's Internet Crime Complaint Center [11]. This success is determined by the fact that phishers are professional adversaries who: (i) have the financial motivation, (ii) exploit computer illiteracy of ordinary Internet users [1], and (iii) manage to learn from their previous experience and improve their future attacks.

The scientific community has put much effort into solving the phishing websites' URL detection problem during the last decade, seeing the ability to determine the maliciousness of a website by evaluating its URL as a significant advantage. The number of victims can be reduced, minimizing operational efforts by avoiding extensive use of more complex methods such as content analysis of a website [3]. Many proposed methods attempted to solve phishing websites detection as a supervised machine learning problem on different phishing datasets with predefined features [5,22]. These methods work well on predefined dataset design but are sensitive to changing environment: phishers can learn the most relevant URL features these methods employ and adapt their behavior to avoid being detected [16]. In search of methods resilient to the changing environment, the scientific community started to explore deep learning techniques with automatic feature detection to identify complicated behaviors with new patterns of attack. Deep learning algorithms developed rapidly in recent years and found many applications for problem solving in different areas. Therefore, the hope of creating a resilient phishing websites' URL detection method lies with these algorithms [3,16].

To our best knowledge, no previous research involved methods, employing the ensembles constructed of RNN and CNN algorithms. The results of the literature review and the well-known fact that ensembles of classification algorithms usually produce better results than single algorithms, motivate us to go one step further and propose a new method, based on the ensembles of RNNs, which would improve classification accuracy.

In this paper, we present the results of our experiment and answer these research questions:

1. Do methods, composed of RNN ensembles, show significantly better classification results in comparison with single RNN methods?

2. Do methods, composed of RNN ensembles, show significantly better results on raw URL dataset in comparison with classical classification algorithms on the same dataset with manually designed features? To answer this question, we compare achieved results with the results of classical classification algorithms on the same dataset from the authors' previous paper [22].

The rest of the paper is organized as follows: In Section 3 we give a review of related works. In Section 4 we describe our research methodology. In Section 5 we report our

Table 1: Accuracy of classical classification methods with predefined features.

Year	**Authors**	**Classifier**	**Phishing URLs**	**Legitimate URLs**	**Accuracy**
2017	Marchal et al. [15]	Gradient Boosting	100,000	1,000	99.90%
2010	Whittaker et al. [26]	Logistic Regression	16,967	1,499,109	99.90%
2011	Xiang et al. [27]	Bayesian Network	8,118	4,780	99.60%
2018	Cui et al. [7]	C4.5	24,520	138,925	99.78%
2013	Zhao et al. [30]	Perceptron	990,000	10,000	99.49%

experiment results. We conclude the paper in Section 6.

3 Related Works

The first methods for phishing websites' URL detection were created more than a decade ago and included blacklisting techniques and heuristic approaches. There still are a few initiatives to use a centralized phishing websites' URLs blacklisting solutions (e.g., PhishTank[2], Google Safe Browsing API [3]). Although these methods were proven unavailing because phishing websites have a very short lifespan (usually not more than a day) and it takes time to detect, report, confirm, and publish a malicious URL in a blacklisting database [24].

Later, as an improvement on blacklisting methods, the heuristic methods were implemented where the signatures of frequent attacks were identified and blacklisted for the future use of Intrusion Detection Systems, giving these methods better generalization capabilities and the ability to detect threats in previously unseen URLs [19]. Although heuristic methods superseded simple blacklisting methods, they could not generalize to all types of new threats [24].

More recent methods for phishing websites' URL detection were based on the application of classical supervised machine learning algorithms on phishing datasets with predefined features. Best performing methods are enlisted in the table 1. These methods scored above 99.49% and were implemented using different types of classifiers: neural networks, regression, decision trees, ensembles, and probabilistic. Although these methods work well on predefined dataset design but they are sensitive to changing environment: phishers can learn the most relevant URL features these methods employ and adapt their behavior to avoid being detected [16]. It should be stated that methods enlisted in the table 1 measure accuracy and use highly unbalanced datasets, therefore, evaluating performance by accuracy does not reveal how these methods would perform on more balanced datasets.

[2]`https://www.phishtank.com/`

[3]`https://developers.google.com/safe-browsing/`

Table 2: Deep learning based classification methods accuracy.

Year	Authors	Classifier	Phishing URLs	Legitimate URLs	Accuracy
2017	Saxe and Berlin [18]	CNN	9,533,939	9,533,939	99.30%
2018	Vazhayil et al. [23]	CNN-LSTM	58,050	58,050	98.90%
2018	Vazhayil et al. [23]	CNN	58,050	58,050	98.70%
2017	Bahnsen et al. [3]	LSTM	1,000,000	1,000,000	98.70 %
2019	Yang et al. [28]	CNN-LSTM	1,021,758	989,021	98.50%
2019	Zhao et al. [29]	GRU	240,000	150,000	98.50%

Most recent approaches use deep learning techniques, including Recurrent neural networks like Long Short-Term Memory networks (LSTM) or Gated Recurrent Unit (GRU). The results of the best methods on balanced datasets are enlisted in table 2. Authors of these papers built their work on the premise that deep learning methods can automatically learn the feature representation from URL's character sequence, without using any predefined features. Additionally, Vazhayil et al. [23] have shown, that adding CNN layers to the LSTM improves classification accuracy.

From the literature review, we can ascertain that blacklisting, heuristic, and classical classification methods on datasets with predefined features are prone to learn a specific dataset design and are not capable of safeguarding internet users from "zero-hour" attacks. We can also see the potential of deep learning-based methods with automatic feature detection to achieve accuracies close to classical classification methods while having the quality of being resilient to the changing environment and cope with "zero-hour" attacks. Furthermore, we can see that the scientific community focuses on RNN and CNN based methods for the qualities like having internal memory, the ability to find patterns in raw data, and automatically learning the essential features from the data.

To our best knowledge, no previous research involved methods, employing the ensembles constructed of RNN and CNN algorithms. The results of the literature review and the well-known fact that ensembles of classification algorithms usually produce better results than single algorithms, motivate us to go one step further and propose a new method, based on the ensembles of RNNs, which would improve classification accuracy.

4 Research Methodology

In this section, we describe our research methodology. We start by defining algorithms used in the experiment and reasons for the algorithm selections (Subsection 4.1), later we describe the dataset (Subsection 4.2), metrics and methods (Subsection 4.3) used in the experiment. We finish this section by explaining how all previous sections are composed to set up the experimental design for our research (Subsection 4.4). The results of our experiment, designed according to this experimental design, are presented in Section 5.

4.1 Algorithms Used in the Experiments

In this subsection, we present the algorithms we used and explain in detail the configuration of methods we composed and tested in our experiment.

We have chosen to employ RNN and CNN algorithms for the construction of our method in order to utilize RNN's properties of having a memory and CNN's properties of recognizing data patterns. Combined properties of these algorithms allow our method to automatically learn the feature representation from URL's character sequence, without using any manually predefined features, thus increasing our method's flexibility, as well as resilience to the changing environment and "zero-hour" attacks.

Figure 1 provides a formal description of the architecture of our RNN-based methods as well as CNN-RNN hybrid methods [23, 28]. Each method consists of three main parts: (i) input and embedding layers, (ii) different RNN (and in some methods additional CNN) layers, and (iii) a dense layer for making a final prediction. We will describe all parts in more detail in the following subsections.

4.1.1 Character Embedding.

Instead of extracting URL features manually, we aim to learn a representation directly from the sequence of URL characters, building our methods on the premise that RNNs can learn essential features and sequential dependencies of the data automatically [3, 18, 28]. First, we encode each URL character in its ASCII code and set the URL size to 256 characters. If the URL is shorter than 256 characters, we pad it with zeroes in the front, and if it is longer, we cut off characters from the beginning of the URL. Later, character encoded URL vectors are provided as inputs to the embedding layer, which optimizes character vectors to better reflect their semantic meaning, by being optimized jointly with the rest of the model during the learning process [25]. The embedding layer parameters are described in table 3.

4.1.2 Recurrent Neural Network Layers.

In our experiment, we use four different recurrent neural networks:

- Long Short-Term Memory network (LSTM), which is a well-known implementation of RNN with a designated memory, widely used by the scientific community in various fields. LSTM overcomes a well known long-term dependency problem of RNNs by implementing a specific memory called "cell state", regulated by structures called gates [10].

- Long Short-Term Memory network with Peepholes (LSTM-P), which is a modification of the LSTM by adding so-called "peephole connections", allowing gate layers to look at the cell state [8].

- Gated Recurrent Units (GRU), which is a noticeable modification of the LSTM, combining the forget and input gates into a single "update gate", resulting in more sim-

Table 3: Hyper-parameters of the models.

Method	**Hyper-parameters**
LSTM	Embedding output dimension: 32; Embedding regularizer: L2 penalty; Output size: 32; Optimizer: Adam (learning rate: 0.01, epsilon: 1e-07); Dropout: 0.5; Penalty: 0.001; Epochs: 40;
LSTM-P	Embedding output dimension: 32; Embedding regularizer: L2 penalty; Output size: 32; Optimizer: Adam (learning rate: 0.01, epsilon: 1e-07); Dropout: 0.5; Penalty: 0.001; Epochs: 64;
GRU	Embedding output dimension: 32; Embedding regularizer: L2 penalty; Output size: 32; Optimizer: Adam (learning rate: 0.01, epsilon: 1e-06); Dropout: 0.5; Penalty: 0.001; Epochs: 40; Loss function: binary cross-entropy;
Simple RNN	Embedding output dimension: 32; Embedding regularizer: L2 penalty; Output size: 32; Optimizer: Adam (learning rate: 0.001, epsilon: 1e-08); Dropout: 0.5; Penalty: 0.001; Epochs: 16;
CNN-LSTM	Embedding output dimension: 64; Embedding regularizer: L2 penalty; Output size: 64; Optimizer: Adam (learning rate: 0.05, epsilon: 1e-06); Dropout: 0.5; GRU layer dropout: 0.35; Penalty: 0.001; Epochs: 40; Kernel size: 3; Pooling size: 4; Convolutional layer activation function: ReLU;
CNN-GRU	Embedding output dimension: 64; Embedding regularizer: L2 penalty; Output size: 64; Optimizer: Adam (learning rate: 0.005, epsilon: 1e-07); Dropout: 0.5; GRU layer dropout: 0.25; Penalty: 0.01; Epochs: 40; Kernel size: 5; Pooling size: 8; Convolutional layer activation function: ReLU;

pler model [6].

- Simple RNN cell[4] based neural network with no explicit memory implementation, which is known for not learning "long-term dependencies" very well due to input noise and vanishing gradient problems [4].

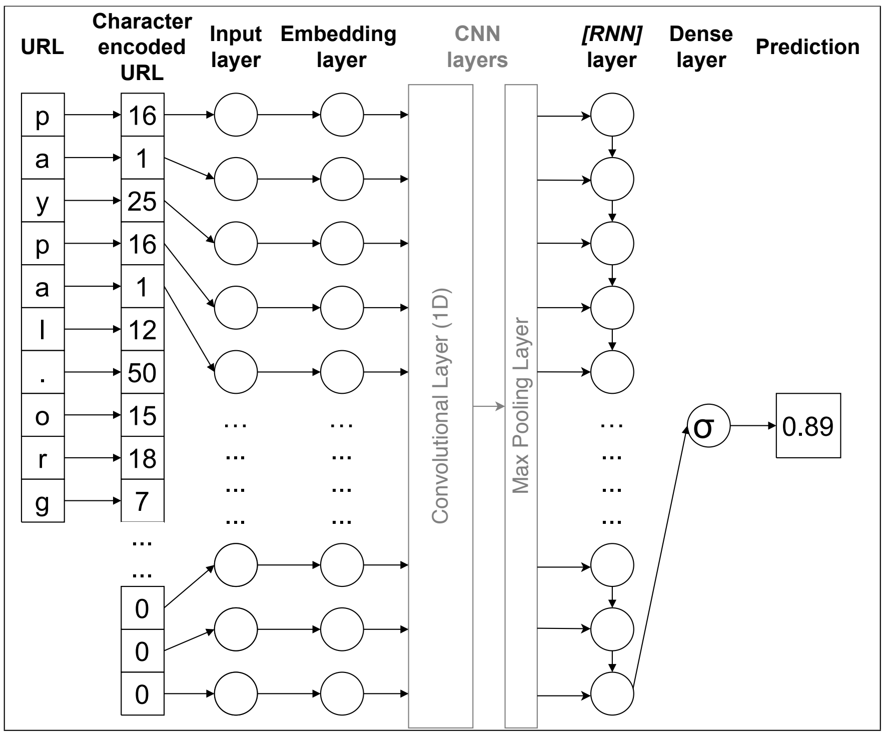
Additionally, we use CNN and RNN hybrids, such as CNN-LSTM and CNN-GRU methods [23]. These hybrid networks benefit from CNN properties to find patterns in data, prior to feeding the data to RNN cells, as depicted in figure 1. We add one 1D convolutional layer and one max-pooling layer between the embedding layer and the RNN layer. We use a dropout regularization layer[5] before the last dense layer to prevent overfitting [21].

Additionally, the Naïve-Bayes probabilistic classifier [13] was trained directly on character encoded strings and used in the ensembles as well.

We use binary cross-entropy loss function, batch size of 200 and dropout rate of 0.5 for all models in our experiment. Other dissimilar hyper-parameters used in our models are provided in table 3.

[4]As described in https://www.tensorflow.org/api_docs/python/tf/keras/layers/SimpleRNNCell
[5]https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

Figure 1: Configuration of our methods



4.1.3 Prediction

is made by the sigmoid function [9] in the last dense layer of the network, giving the real value between 0 and 1, which corresponds to the prediction probability to our classes ('0' for legitimate URL, '1' for phishing URL).

4.1.4 Stacking Ensemble

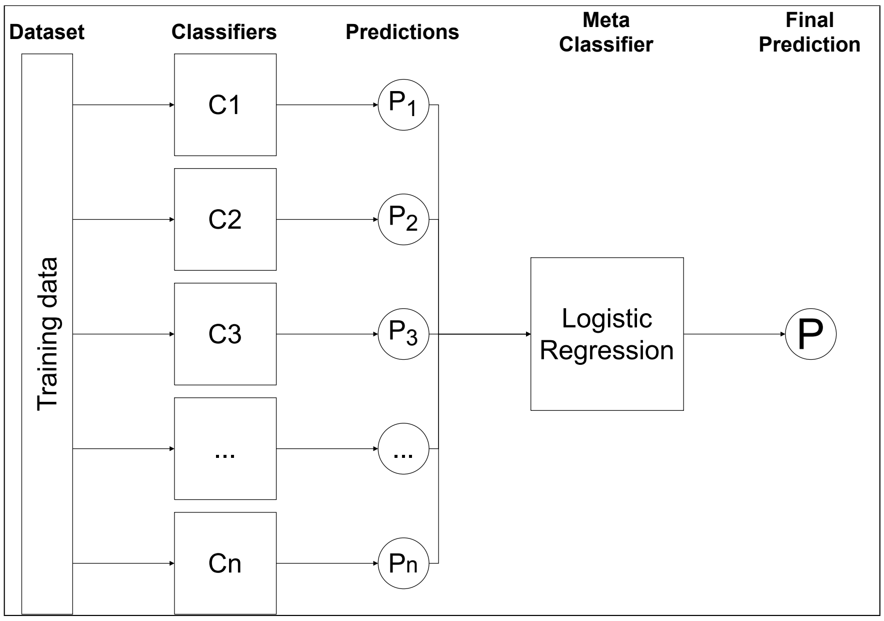
is used in our experiment to combine predictions of previously trained methods into a unified predictor. As described in the figure 2, a new algorithm called "meta classifier" (in our case based on Logistic regression [12]) learns how to combine the predictions from multiple methods and produce a final prediction. The meta-classifier is trained based on predicted probabilities instead of class labels. We used MLxtend 0.17.0 library to build our stacking ensemble methods[6].

4.2 Dataset

In our experiment, we used a dataset from Mendeley Data portal[7], published by Choon Lin Tan (Universiti Malaysia Sarawak) in March 2018. This balanced dataset contains 5,000 phishing and 5,000 legitimate websites URL samples. Additionally, a total of 48 features were extracted from these websites by the authors [5]. We have chosen this dataset for our experiment so that we could compare the performance of the proposed method

[6] http://rasbt.github.io/mlxtend/
[7] https://data.mendeley.com/datasets/h3cgnj8hft/1

Figure 2: Configuration of Stacking Ensemble methods



with the performance of classical classification algorithms from our previous paper [22], where the same dataset was used.

4.3 Measures and Methods

4.3.1 Classification Accuracy

in our experiment is the rate of phishing and legitimate websites which are identified correctly with respect to all the websites, defined as follows:

$$ACCURACY = \frac{TP + TN}{TP + FP + TN + FN}, \tag{1}$$

where

- TP - number of websites, correctly detected as phishing (True Positive),
- TN - number of websites, correctly detected as benign (True Negative),
- FP - number of legitimate websites, incorrectly detected as phishing (False Positive),
- FN - number of phishing websites, incorrectly detected as legitimate (False Negative).

We chose classification accuracy as our classification quality quantification metric because: (i) most other researchers use classification accuracy to define results of their experiments (see Section 3), therefore the comparability of research results is homogeneous throughout our work; (ii) in our experiment, we used a dataset with equal class distributions (there is no disparity between the number of positive and negative labels) therefore we do not have the majority and minority classes; (iii) we used cross-validation function with stratification option, which generates test sets that contain the same distribution of classes, or as close as possible. In these circumstances, classification accuracy is a useful non-bias measure.

4.3.2 Welch's T-Test

in our experiment was used to determine whether the means of classification accuracy results produced by any two classifiers have a statistically significant difference. We used *scipy.stats* package for Python to perform a T-test.

4.3.3 Shapiro–Wilk Test

was used to check whether samples came from a normally distributed population [20]. We used *scipy.stats* package for Python to perform a Shapiro–Wilk test.

4.4 Experimental Design

In this subsection, we present the experimental design we employed to perform the experiment. The objective is to train all the classifiers from Section 4.1 on our chosen dataset from Section 4.2 for their best possible classification accuracy described in Section 4.3.1, and to compare classification results. The experiment was divided into two parts: (i) training the classifiers, (ii) comparing the classification results.

We train the classifiers by taking these steps:

1. Configure the classifier in Python's 3.7.5 environment using Tensorflow-GPU 2.0.0 library[8]; we use a computer with Nvidia GeForce RTX 2080 Ti graphical card, Intel Core i7-4770 3.40GHz processor, and 16 Gb of RAM to perform our experiment.

2. Perform the exhaustive grid search to find the best fitting hyper-parameters, using the Scikit Learn 0.22.0 library[9] [17].

3. Train and test the classifier using a cross-validation function with 10 stratified folds from the Scikit Learn 0.22.0 library. Repeat this step 3 times with different seed to get 30 classification accuracy measures.

4. Perform a Wilk-Shapiro test, as described in Section 4.3, to check if the accuracy scores are normally distributed. If not, take action to normalize the values (e.g., repeating the training and testing step).

5. Save the results for further actions.

We compare classification results by taking these steps:

[8]https://www.tensorflow.org

[9]https://scikit-learn.org/

1. Using Welch's T-test, described in Section 4.3.2, check every possible pair of classifiers if their mean classification accuracies have statistically significant differences.
2. Arrange all classifiers by their mean classification accuracy in descending order.
3. Evaluate groups of methods that mean classification accuracies have no statistically significant differences.
4. Plot all results in the box plot.
5. Compare the results with our other experiment with classical classification algorithms on the same dataset with manually extracted features [22].

For each algorithm, we perform an experiment with all the steps described above.

5 Experimental Results

In this section, we present the results of our experiment, conducted following the research design described in Section 4.4. We present all RNN-based methods and five best RNN ensemble-based methods of our experiment in the figure 3. In this box-plot, the yellow dotted lines group the algorithms which mean accuracy results (assessed with Welch's T-test, as described in section 4.3.2) have no statistically significant differences with each other.

From this diagram, we can see that:

- the results of ensembles 1, 2, and 3 have no statistically significant difference, but are significantly better than all other models;
- the results of ensembles 4, 5, and CNN-LSTM have no statistically significant difference, but are significantly better than GRU, CNN-GRU, LSTM, LSTM-P, and Simple RNN methods;
- the results of GRU, CNN-GRU, LSTM, and LSTM-P have no statistically significant difference but are significantly better than the Simple RNN method.
- Simple RNN method demonstrated the worst result of RNN-based methods. This result is consistent with the theory, provided in section 4.1, stating that simple RNN with no specifically implemented memory can not learn "long-term dependencies" and is inferior in comparison with LSTM or GRU.

In table 4, we compare the results of this experiment (in the highlighted font) with our previous experiment, were we applied classical classification algorithms on the same dataset with manually extracted features [22].

Welch's T-test has shown that Ensembles 1 - 3 give the results of the same significance as Gradient Tree Boosting and AdaBoost, applied on the same dataset only with manually selected features. We can see from the results that single RNN methods score 1 - 2 %

Figure 3: Classification results

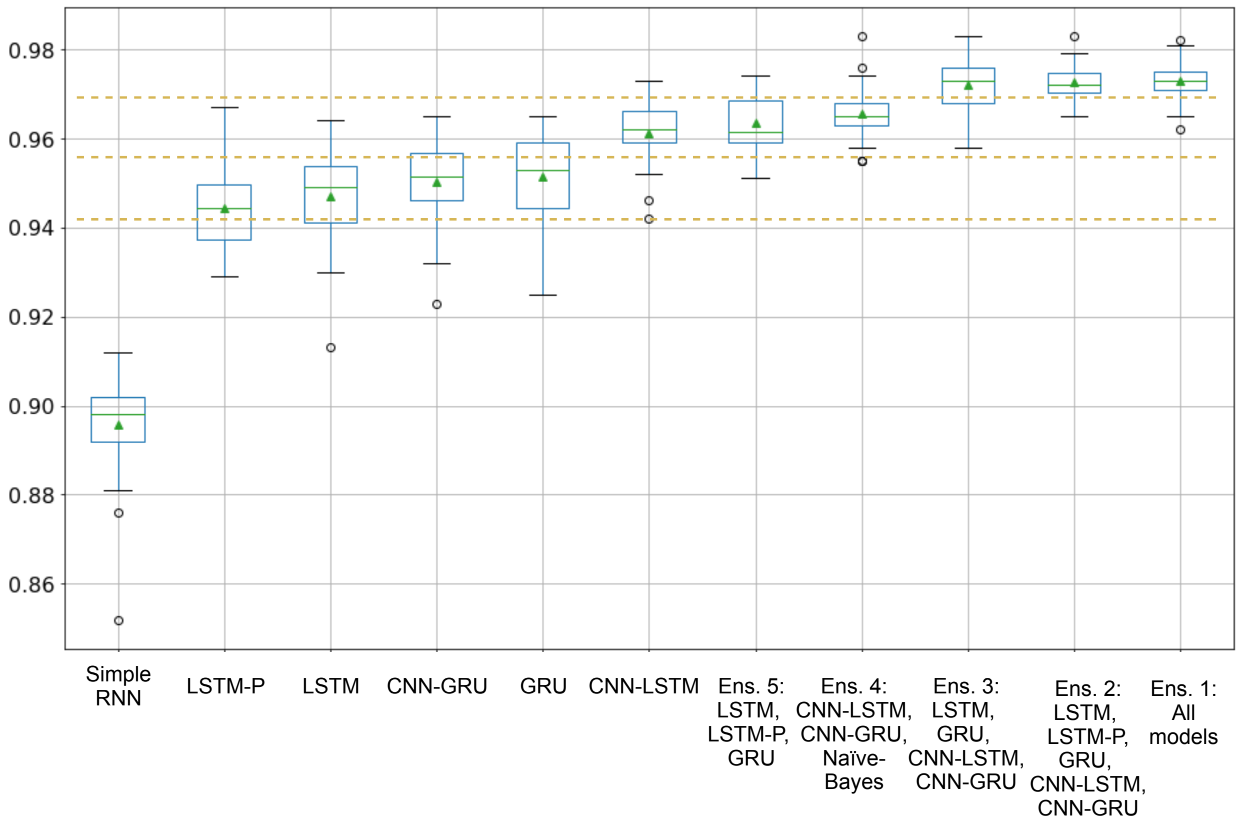


Table 4: Classification results

Method	**Accuracy**
Gradient Tree Boosting	0.9742
Ensemble 1 (LSTM, LSTM-P, GRU, CNN-LSTM, CNN-GRU, Naïve-Bayes)	**0.9730**
AdaBoost	0.9728
Ensemble 2 (LSTM, LSTM-P, GRU, CNN-LSTM, CNN-GRU)	**0.9725**
Ensemble 3 (LSTM, GRU, CNN-LSTM, CNN-GRU)	**0.9721**
Random Forest	0.9715
Multilayer Perceptron	0.9671
CNN-LSTM	**0.9612**
Classification and Regression Trees	0.9574
Ensemble 4 (CNN-LSTM, CNN-GRU, Naïve-Bayes)	**0.9657**
Ensemble 5 (LSTM, LSTM-P, GRU)	**0.9634**
Support Vector Machine	0.9549
GRU	**0.9515**
CNN-GRU	**0.9503**
LSTM	**0.9471**
LSTM-P	**0.9443**
Naïve-Bayes	0.9177
SimpleRNN	**0.8958**
Naïve-Bayes (this experiment)	**0.6056**

lower accuracies than the best ensemble-based methods. We also observe that Ensembles 1 - 3, which scored the best accuracy results, have a higher number of RNN methods, and this confirms theoretical results that the higher number of independent classifiers improve ensemble classification accuracy. We can also see from our results that RNN without memory (SimpleRNN cell) performs worse of all RNNs. The conclusions of our experiment results are provided in Section 6.

6 Conclusions

In this paper, we have proposed a new ensemble-based method for phishing websites' URL detection. We have presented the results of our experiment, where we have compared our method with other different types of methods, based on RNNs, CNNs, and different hybrid methods, consisting of these algorithms. From our research, we draw the following conclusions, which are valid on the dataset we have used:

1. Our proposed method, employing RNN-based ensembles, outperform single RNN methods by at least 0.02 difference in classification accuracy, which is statistically significant.

2. Our proposed method performs as well as Gradient Tree Boosting and AdaBoost with human extracted features on the same dataset. The accuracies between our

method, Gradient Tree Boosting, and AdaBoost ensembles have no statistically significant difference, according to Welch's T-test.

3. Adding the CNN layer to the method increases classification accuracy by 0.01, and this difference is statistically significant.

4. For phishing websites' URL classification problem, RNNs with explicit memory implementation, like LSTM and GRU, outperform classic RNNs without memory by 0.06 difference in classification accuracy, which is statistically significant.

5. RNNs outperform simple probabilistic classifiers like Naïve-Bayes on URL characters sequence by 0.355 increase of accuracy. Additionally, Naïve-Bayes performs significantly better on manually extracted features rather than string character sequences, with a 0.31 difference of classification accuracy.

References

[1] Adebowale, M., Lwin, K., Sánchez, E., Hossain, M.: Intelligent web-phishing detection and protection scheme using integrated features of Images, frames and text. Expert Systems with Applications **115**, 300–313 (jan 2019). https://doi.org/10.1016/J.ESWA.2018.07.067, https://www.sciencedirect.com/science/article/pii/S0957417418304925?via{%}3Dihub

[2] Anti-Phishing Working Group, I.: Phishing Activity Trends Reports (2019), https://apwg.org/resources/apwg-reports/

[3] Bahnsen, A.C., Bohorquez, E.C., Villegas, S., Vargas, J., Gonzalez, F.A.: Classifying phishing URLs using recurrent neural networks. In: 2017 APWG Symposium on Electronic Crime Research (eCrime). pp. 1–8 (2017). https://doi.org/10.1109/ECRIME.2017.7945048, http://ieeexplore.ieee.org/document/7945048/

[4] Bengio, Y., Simard, P., Frasconi, P.: Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Transactions on Neural Networks **5**(2), 157–166 (1994). https://doi.org/10.1109/72.279181

[5] Chiew, K.L., Tan, C.L., Wong, K., Yong, K.S., Tiong, W.K.: A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Information Sciences **484**, 153–166 (may 2019). https://doi.org/10.1016/j.ins.2019.01.064, https://www.sciencedirect.com/science/article/pii/S0020025519300763?via{%}3Dihubhttps://linkinghub.elsevier.com/retrieve/pii/S0020025519300763

[6] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078v3 (2014)

[7] Cui, B., He, S., Yao, X., Shi, P., Yao, X., He, S., Cui, B.: Malicious URL detection with feature extraction based on machine learning. International Journal of High Performance Computing and Networking **12**(2), 166 (2018). https://doi.org/10.1504/ijhpcn.2018.10015545, http://www.inderscience.com/link.php?id=94367

[8] Gers, F.A., Urgen Schmidhuber, J.J., Cummins, F.: Learning to Forget: Continual Prediction with LSTM. In: Proc. ICANN'99 Int. Conf. on Arti?cial Neural Network. vol. 2, pp. 850–855. IDSIA (1999), http://www.idsia.ch/http://www.idsia.ch/

[9] Han, J., Moraga, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 930, pp. 195–201. Springer Verlag (1995). https://doi.org/10.1007/3-540-59497-3_175

[10] Hochreiter, S., Urgen Schmidhuber, J.J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (1997), http://www7.informatik.tu-muenchen.de/{~}hochreithttp://www.idsia.ch/{~}juergen

[11] Internet Crime Complaint Center: 2018 Internet Crime Report. Tech. rep., Internet Crime Complaint Center at the Federal Bureau of Investigation of United States of America (2019), https://www.ic3.gov/media/annualreport/2018{_}IC3Report.pdf

[12] Kleinbaum, D.G., Klein, M.: Introduction to Logistic Regression. In: Logistic Regression, pp. 1–39. Springer, New York, NY (2010). https://doi.org/10.1007/978-1-4419-1742-3_1, http://link.springer.com/10.1007/978-1-4419-1742-3{_}1

[13] Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In: ECML 1998: Machine Learning: ECML-98, pp. 4–15. Springer, Berlin, Heidelberg (1998). https://doi.org/10.1007/BFb0026666, http://link.springer.com/10.1007/BFb0026666

[14] Lin Tan, C., Leng Chiew, K., Wong, K.S., Nah Sze, S., Tan, C.L., Chiew, K.L., Wong, K.S., Sze, S.N.: PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder. Decision Support Systems **88**, 18–27 (2016). https://doi.org/10.1016/j.dss.2016.05.005, http://dx.doi.org/10.1016/j.dss.2016.05.005

[15] Marchal, S., Armano, G., Grondahl, T., Saari, K., Singh, N., Asokan, N.: Off-the-hook: An efficient and usable client-side phishing prevention application. IEEE Transactions on Computers **66**(10), 1717–1733 (2017). https://doi.org/10.1109/TC.2017.2703808

[16] Opara, C., Wei, B., Chen, Y.: HTMLPhish: Enabling Accurate Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis. arXiv:1909.01135 (aug 2019), www.phishtank.comhttp://arxiv.org/abs/1909.01135

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12**(Oct), 2825–2830 (2011), http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.htmlhttps://scikit-learn.org/stable/

[18] Saxe, J., Berlin, K.: eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint arXiv:1702.08568 (feb 2017), http://arxiv.org/abs/1702.08568

[19] Seifert, C., Welch, I., Komisarczuk, P.: Identification of Malicious Web Pages with Static Heuristics. In: 2008 Australasian Telecommunication Networks and Applications Conference. pp. 91–96. IEEE (dec 2008). https://doi.org/10.1109/ATNAC.2008.4783302, http://ieeexplore.ieee.org/document/4783302/

[20] Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). Biometrika **52**(3/4), 591–611 (1965)

[21] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (jan 2014)

[22] Vaitkevicius, P., Marcinkevicius, V.: Comparison of Classification Algorithms for Detection of Phishing Websites. Informatica **31**(1), 143–160 (mar 2020). https://doi.org/10.15388/20-infor404

[23] Vazhayil, A., Vinayakumar, R., Soman, K.: Comparative Study of the Detection of Malicious URLs Using Shallow and Deep Networks. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–6. IEEE (jul 2018). https://doi.org/10.1109/ICCCNT.2018.8494159, https://ieeexplore.ieee.org/document/8494159/

[24] Verma, R., Das, A.: What's in a URL. In: Proceedings of the 3rd ACM on International Workshop on Security And PrivacyAnalytics - IWSPA '17. pp. 55–63. ACM Press, New York, New York, USA (2017). https://doi.org/10.1145/3041008.3041016, http://dl.acm.org/citation.cfm?doid=3041008.3041016

[25] Wei, B., Hamad, R.A., Yang, L., He, X., Wang, H., Gao, B., Woo, W.L.: A Deep-Learning-Driven Light-Weight Phishing Detection Sensor. Sensors **19**(19), 4258 (sep 2019). https://doi.org/10.3390/s19194258

[26] Whittaker, C., Ryner, B., Nazif, M.: Large-Scale Automatic Classification of Phishing Pages. The 17th Annual Network and Distributed System Security Symposium (NDSS '10) (2010). https://doi.org/10.1109/TDSC.2013.3, http://www.isoc.org/isoc/conferences/ndss/10/pdf/08.pdf{%}5Cnhttp://research.google.com/pubs/pub35580.html

[27] Xiang, G., Hong, J., Rose, C.P., Cranor, L.: CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites. ACM Transactions on Information and System Security **14**(2), 1–28 (sep 2011). https://doi.org/10.1145/2019599.2019606, http://dl.acm.org/citation.cfm?doid=2019599.2019606https://www.ml.cmu.edu/research/dap-papers/dap-guang-xiang.pdf

[28] Yang, P., Zhao, G., Zeng, P.: Phishing website detection based on multidimensional features driven by deep learning. IEEE Access **7**, 15196–15209 (2019). https://doi.org/10.1109/ACCESS.2019.2892066

[29] Zhao, J., Wang, N., Ma, Q., Cheng, Z.: Classifying Malicious URLs Using Gated Recurrent Neural Networks. In: International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 385–394. Springer (jul 2019). https://doi.org/10.1007/978-3-319-93554-6_36, http://link.springer.com/10.1007/978-3-319-93554-6{_}36

[30] Zhao, P., Hoi, S.C.: Cost-sensitive online active learning with application to malicious URL detection. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13. p. 919. ACM Press, New York, New York, USA (2013). https://doi.org/10.1145/2487575.2487647, http://dl.acm.org/citation.cfm?doid=2487575.2487647