MACHINE LEARNING MODEL

COMPARISON BASED ON SOME METRICS

Safa ORHAN

Computer Engineering Student

Istanbul Kultur University

Istanbul, Turkey

Eyüp USTA

Computer Engineering Student

Istanbul Kultur University

Istanbul, Turkey

I. A Deep-Learning-Driven Light-Weight Phishing Detection Sensor:

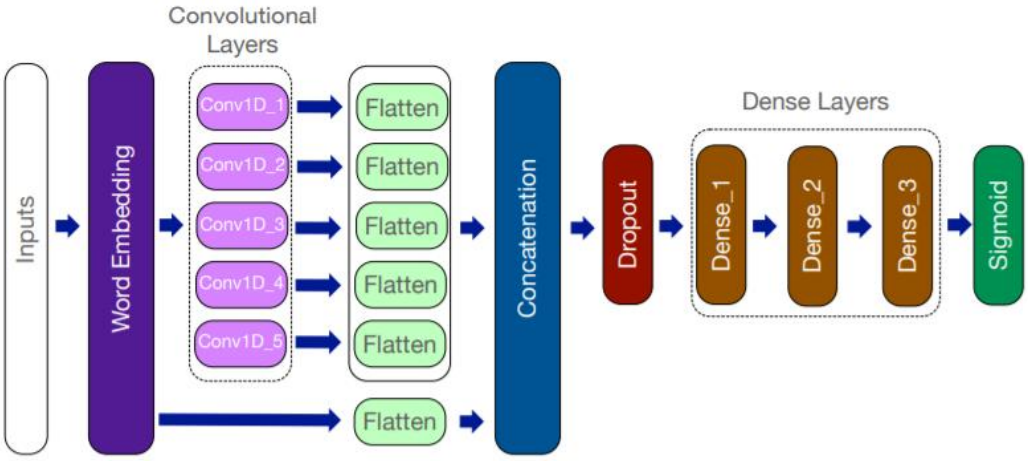

Table 1. Architecture configuration of the proposed DNN model.

	Output Dimension	
Word Embedding	32	
	Number of Filters	Kernel Size
Conv1D_1	256	2
Conv1D_2	256	3
Conv1D_3	256	4
Conv1D_4	256	5
Conv1D_5	256	10
	Dropout Rate	
Dropout	0.5	
	Number of Units	
Dense_1	128	
Dense_2	128	
Dense_3	128	

Table 2. The effect of the dense layers.

	Accuracy
Proposed	**86.630%**
1 Dense Layer	86.537%
2 Dense Layers	86.538%
3 Dense Layers	86.542%

Table 3. The effect of the convolutional layers.

	Accuracy
Proposed	**86.630%**
1 Convolutional Layer	85.401%
2 Convolutional Layers	85.832%
3 Convolutional Layers	86.169%
4 Convolutional Layers	86.439%

Table 4. The effect of the concatenation of the output from the embedding layer.

	Accuracy
Proposed	**86.630%**
No Concatenation	83.472%

II. Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning:

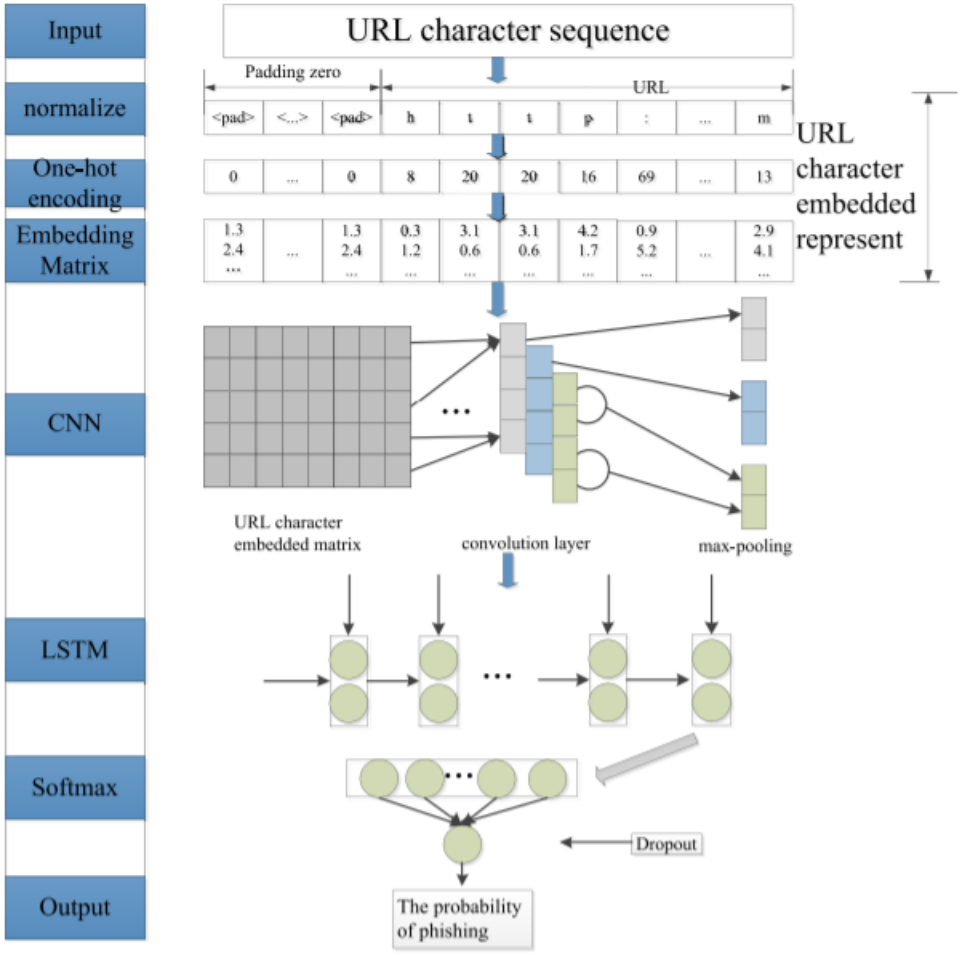

FIGURE 3. The CNN-LSTM algorithm.

Algorithm 1 The CNN-LSTM Algorithm

Input: The training set U_{train}, the testing set U_{test}, $u_i \in S_{train}$, $u_i' = S_{test}$.
Output: The probability of phishing P(u_i').

K = num of sliding-window, k = size of sliding step, p = size of pooling-window, β = threshold of the loss function $L(x, y)$. $\text{Weight}_{\text{embedding}} = V$, $V \in \mathbb{R}^{p \times m}$, $G \in \mathbb{R}^{m \times n}$.
$U = U_{train} \cup U_{test}$, $l = |U|$, $G = \emptyset$, $N = \lceil K/p \rceil$,
for i in l **do**
  $u_i \in U$, $e_i = URLF(u_i)$
  $g_i = Asc(e_i)$
  $G = G \cup g_i$
end for
$S = VG = (\overrightarrow{s_1}, \overrightarrow{s_2}, \ldots, \overrightarrow{s_n})$
for f in Q **do**
  for i in C **do**
    $h_i^f = \sigma(W_f \cdot (\overrightarrow{s_i}, \overrightarrow{s_{i+1}}, \ldots, \overrightarrow{s_{i+k-1}}) + b_f)$
  end for
end for
for f in Q **do**
  for k in N **do**
    $p_j^f = Max(h_{(j-1)p}^f, h_{(j-1)p+1}^f, \ldots, h_{jp-1}^f)$
  end for
end for
$H_p = (\overrightarrow{p^1}, \overrightarrow{p^2}, \ldots, \overrightarrow{p^j}, \ldots, \overrightarrow{p^s})^T$, $\overrightarrow{p^j} \in \mathbb{R}^{N \times 1}$
$H = (h_1, h_2, \ldots, h_N) = lstm(H_p)$
$C'' = \text{softmax}(h_N)$
while $L(C'', C) > \beta$
  $W = Train(u_i, C_i)$
end while
$P(d_i') = Test(u_i', W)$
return $P(u')$

Algorithm 2 The Multidimensional Feature Algorithm

Input: The URL set U, $U = \{u_1, u_2, \ldots, u_n\}$
Output: The probability of phishing P(U)

$N = |U|$, $H = \emptyset$
for i in N **do**
  $Fd = \emptyset$, $Fu = \emptyset$, $Fc = \emptyset$, $Ft = \emptyset$, $F = \emptyset$
  $Fd = \text{CNN-LSTM}(u_i)$
  $Fu = \text{extract\_url}(u_i)$
  $Fc = \text{extract\_code}(u_i)$
  $Ft = \text{extract\_text}(u_i)$
  $F = Fd \cup Fu \cup Fc \cup Ft$
  $H = H \cup F$
end for
$P(U) = XGBoost(H)$
return $P(U)$

Algorithm 3 The Dynamic Category Decision Algorithm

Input: The suspicious URL u_i
Output: The status of detection

$p_0 = \text{CNN-LSTM}(u_i)$
$p_1 = 1 - p_0$
$maxi = \max(p_0, p_1)$
$mini = \min(p_0, p_1)$
if $(maxi/mini > \alpha || !access(u_i))$ **then**
  if $(p_1 > p_0)$ **then**
    return '*phishing*'
  else return '*legitimate*'
end if
else return multidimensional_features (u_i)
end if

TABLE 3. Data distribution.

dataset	positive	negative	total
DATA1	22445	22390	44835
DATA2	999313	966631	1965944

TABLE 4. Comparison of different models on *DATA2*.

Models	Accuracy/%	FPR/%	FNR/%	Cost/%	Epoch/s
CNN	95.87	3.04	5.18	20.4	45
CNN-CNN	98.12	1.07	2.65	8.02	64
RNN	97.87	1.46	2.78	10.09	97
RNN-RNN	97.71	2.12	2.47	13.05	148
LSTM	98.42	1.18	1.97	7.87	256
LSTM-LSTM	98.57	1.05	1.79	7.06	578
CNN-RNN	98.44	1.07	2.04	7.38	72
CNN-LSTM	98.61	0.96	1.82	6.6	140

TABLE 5. Comparison of MFPD and other approaches.

Approaches	Precision/%	Recall/%	F1
J. Mao et.al [11]	94.22	95.82	0.94
CANTINA+ [19]	97.50	93.47	0.963
X. Zhang et.al [32]	98.60	98.80	0.987
MFPD	99.41	98.57	0.990

III. Deep Learning with Convolutional Neural Network and Long Short Term Memory for Phishing Detection:

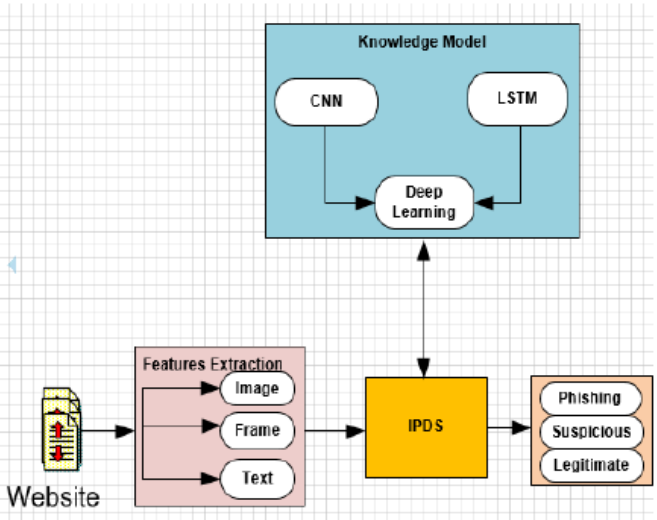

Fig. 3: Intelligent Phishing Detection System (IPDS) Structure

B. Model Details

The model was developed in Matlab version 9.5 using the deep learning toolbox. For the CNN architecture, there were three categories of data. The image (Fig. 4) is a sample of image which was loaded into the image data store and processed to extract the speeded-up robust features (SURF) from all images using the grid method to create a bag of features where the GridStep was set to [8 8] with BlockWidth of [32 64 96 128]. Then we used clustering to create a 1000-word visual vocabulary. For the LSTM architecture, the dataset was partitioned, and holdout cross-validation was set to 0.3 for training and validation. The URLs were tokenised to separate each URL into a series of separate words, all of which were set in lowercase. The tokenised data was then encoding to make it available for training, where the maximum length was set to 75, the hidden size was set to 180, and the embedding dimension was 100 with the fully connected network. The training options were set to *adam*; epoch = 100, gradient threshold = 1, learning rate = 0.01 and verbose = false. By doing this, we were able to tweak the network architecture layer that included the parameter mentioned above to achieve better training accuracy.

Table 1: Relative performance

Algorithm	Accuracy %	Recall %	Precision %	F-measure %
CNN	92.55	92.51	92.58	92.54
LSTM	92.79	92.78	92.81	92.80
IPDS	93.28	93.27	93.30	93.29

IV. HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis:

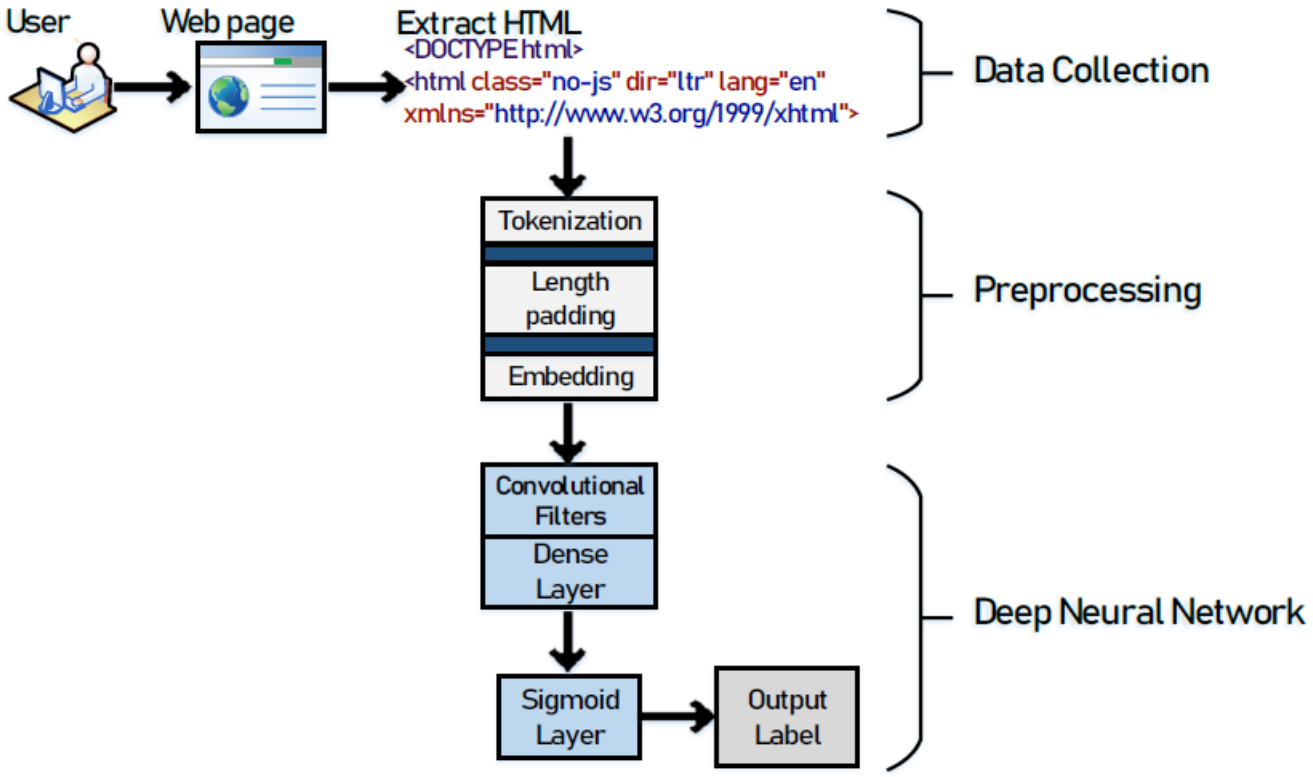

Fig. 2: A Schematic Overview of the Stages Involved in Our Proposed Model

TABLE I: HTML Documents Used in this Paper

Dataset	D1	D2
Date generated	11 - 18 Nov, 2018	10 -17 Jan, 2019
Legitimate Web Pages	23,000	24,000
Phishing Web pages	2,300	2,400
Total	25,300	26,400

TABLE II: HTMLPhish-Full Deep Neural Network

Layers	Values	Activation
Embedding	Dimension = 100	-
Convolution	Filter = 32, Filter Size = 8	ReLU
Max Pooling	Pool Size = 2	-
Dense1	No. of Neurons = 10, Dropout = 0.5	ReLU
Dense2	No. of Neurons = 1	Sigmoid
Total Number of Trainable Parameters	412,388,597	-

TABLE III: Result of HTMLPhish and Baseline Evaluations on the D1 dataset

Models	Accuracy	Precision	True Positive Rates	F-1 Score	AUC	Training time
HTMLPhish-Full	**0.98**	**0.97**	**0.98**	**0.97**	**0.93**	**6.75 mins**
HTMLPhish-Word	0.94	0.93	0.94	0.93	0.88	10 mins
HTMLPhish-Character	0.95	0.92	0.95	0.94	0.90	3.5 mins
[28]	0.97	0.96	0.97	0.96	0.93	5.25 mins
[20]	0.95	0.94	0.95	0.94	0.91	18 mins

TABLE IV: Result of HTMLPhish and Baseline Evaluations on the D2 dataset

Models	Accuracy	Precision	True Positive Rates	F-1 Score	AUC	Testing time
HTMLPhish-Full	**0.93**	**0.92**	**0.93**	**0.91**	**0.88**	**9 seconds**
HTMLPhish-Word	0.90	0.87	0.91	0.88	0.73	107 seconds
HTMLPhish-Character	0.91	0.89	0.91	0.89	0.77	7 seconds
[28]	0.91	0.84	0.91	0.87	0.73	15 seconds
[20]	0.90	0.90	0.92	0.90	0.78	112 seconds

V. Research on Website Phishing Detection Based on LSTM RNN:

The experiment uses the Python programming language. The LSTM model is implemented by a deep learning class such as Keras. It contains 5 LSTM layers with 128 nodes each.

The model uses a stochastic gradient descent (SGD) optimization method with an initial learning rate of one thousandth and a batch size of 128. The objective function of the least-squares fit is a quadratic polynomial function. The dataset used consisted of 2000 legitimate websites collected from Yahoo Directory (http://dir.yahoo.com/) and 2,000 phishing websites collected from Phishtank (http://www.phishtank.com/). Collected data sets carry label values, "legal" and "phishing". In this data set randomly selected 70% for training, 30% for the test. The training dataset is used to train the neural network and adjust the weight of the neurons in the network, while the test dataset remains unchanged and used to evaluate the performance of the neural network. After training, run the test data set on the optimized neural network.

Predict phishing websites using LSTM Recurrent neural network. The above ten features are taken as input, that is, the number of input layer nodes in the LSTM network is 10 and the number of output layer nodes is one. Training network to choose a strong adaptability of the three-layer LSTM network, incentive function is sigmoid function:

$$f(x)=\frac{1}{1+e^{-x}} \tag{11}$$

LSTM neural network for classifying phishing URLs based on LSTM units. Each input character is translated by a 128-dimension embedding. The translated URL is fed into a LSTM layer as a 150-step sequence. Finally, the classification is performed using an output sigmoid neuron. The learning rate of LSTM neural network is 0.1.

In order to better illustrate the accuracy of the algorithm in this paper, an ordinary CNN is used to test the experimental data set. By experimenting with the selected data set, the results show that LSTM network are better than normal CNN, and their prediction accuracy is higher than that of CNN.

TABLE I. EVALUATIONS OF LSTM RNN AND CNN

Method	Accuracy	Precision	Recall	FNR
CNN	0.9742	0.9648	0.9723	0.0591
LSTM	0.9914	0.9874	0.9891	0.0212