Email Phishing Detection using Deep Learning

Aatrayee Bhattacharjee
{a56bhatt}@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Introduction

In today's world, most of us rely on online resources for majority of our activities, such as shopping, watching movies, banking and education. Our online presence makes us susceptible to malicious activities from attackers. Phishing is one such malicious activity where attackers send fraudulent emails or messages to trick people into sending out their sensitive information. One example of a phishing mail is where people are given false warnings about their account credentials expiring if the link in the email is not clicked to enter the person's login details. To look more genuine, these attackers sometimes pose as big companies with subtle change in the email domain or as a friend or family member. They add an emotional incentive to the email such as, limited time to perform the task specified in the email. People who are not aware of phishing emails tend to click on the URL given in these emails and are taken to malicious websites. They are prompted to fill in their login credentials and this information is used by the attackers for malpractices such as identity theft or monetary gains.

Phishing attacks are extremely prominent, especially in the corporate world. Big corporations conduct mandatory annual training to help their employees identify phishing attacks. Some conduct social engineering tests as well, to check if the employees are aware of signs of phishing. Failing the test might limit the employees mailbox features. Some common features in phishing emails are: incorrect spellings, poor grammar, uncommon phrases, suspicious URLs. Though training employees on detecting phishing attacks is important, due to excessive emails received by them all, people may not notice the phishing email and fall prey to it. Attackers are also upgrading the content of phishing emails to make it look more genuine. One solution to this problem is to have automatic phishing email detectors in mailboxes that would prevent such malicious attacks.

According to the Anti-Phishing Working Group's report for 2020, the first quarter had 165,772 phishing sites detected, second quarter had 146,994 phishing sites detected and the third quarter had 571764 phishing sites detected (APWG 2021). This shows that the number of attacks drastically increased during the third quarter compared to quarter one and two. There was an increase in COVID-19 related phishing emails such as health packages. Many of the phishing sites were SSL protected. This makes us aware of the speed with which phishing attacks are increasing and we must come up with an efficient solution to prevent further attacks.

There are many implementations of using machine learning techniques or neutral networks to detect phishing emails. Many studies make the use of deep learning networks like Long Short-Term Memory and it provides efficient results (Chen, Zhang, and Su 2018). However, the studies take place using different databases and we cannot make a proper evaluation on which technique is more efficient. The research question of this project is: Will neural networks like Long Short-Term Memory (LSTM) perform better than Support Vector Machines in detecting phishing emails using the same database and similar features?

Emails tend to have numerous features such as sender ID, sender domain, subject, body, etc. In this study, we will compare the performance of Long Short-Term Memory with the performance of CS-SVM using a combination of important features to efficiently detect phishing emails. The performance metrics used to compare the results are: accuracy, precision and recall. The dataset to be used is approximately 3000 phishing URLs from PhishTank (OpenDNS 2016) and Apache SpamAssassin (Mason 2005) for spam and non-spam emails. The study will take place using Python and its existing libraries such as scikit-learn, keras and tensorflow.

Related Work

Various techniques have been used in the past to detect phishing emails. Before moving to approaches that use machine learning to detect phishing emails, let us discuss few other approaches:

White-list: The white-list contains a list of websites that the user regularly visits. The approach works well for zero day phishing attacks and has a low false positive rate. A zero day phishing attack is an attack that harms any component of software the developer is not aware of and must be fixed immediately to prevent further harm. The issue with this approach is, if a user visits a new website that is not regularly used, the white-list considers it as a phishing URL. Due to this, white-lists have a high false negative rate. Another disadvantage of white-list is that it has to be manually updated and maintained. (Li, Helenius, and Berki 2012)

Black-list: The black-list approach is a popular and simple approach to detect phishing websites. Most widely used web browsers such as Internet Edge and Google Chrome use a black-list to prevent users from opening malicious websites. In the black-list approach, a URL is compared with a black-list database that is already predefined. Blacklist tend to have low false positive rates. The disadvantage of black-list is that they do not work well on zero day phishing attacks as the black-list is already predefined and it is not possible for it to contain newly made URLs. (Sheng et al. 2009)

Network-Based approach interferes TCP or UDP to block malicious IP addresses but this approach is expensive and time consuming. (Gupta et al. 2017)

Content based approaches are most effective as they have a high accuracy rate, but they require large amount of training data to be efficient. One study that uses Cuckoo Search-SVM outperforms regular SVM as Cuckoo Search helps pick optimal parameters for Radial Basis Function instead of the default SVM parameters. This method has a higher True Positive Rate, False Negative Rate, 99.52% accuracy and uses 23 features to train the model on a dataset. (Niu et al. 2017)

A phishing detection system based on LSTM Recurrent Neural Network has an accuracy of 99.17% compared its this baseline CNN with 97% accuracy. This system takes into consideration the URLs in the email instead of considering both the URL and text, which is taken in the previous study. (Chen, Zhang, and Su 2018)

Systems that use dynamic evolving neural networks along on reinforcement learning use 55 features including both the text and URL which is appropriately selected based on the dataset used. Using reinforcement learning (mean square error) the number of nodes of the input, output and hidden layers of the neural network are selected according to the dataset and the neural network is trained efficiently. The system has 98.63%, 99.07%, and 98.19% of accuracy, True Positive Rate and True Negative Rate respectively.(Smadi, Aslam, and Zhang 2018)

In the study by (Moradpoor, Clavie, and Buchanan 2017), a custom neural network model is implemented. Sufficient results are obtained using this model.

(Fang et al. 2019) used recurrent convolutional neural networks with the attention mechanism and had an accuracy of 99.48% and a false positive rate of 0.043%. It used both the email header and the email body at both character and word levels and used an unbalanced dataset to mimic real work conditions.

(Paliath, Qbeitah, and Aldwairi 2020) compared five machine learning techniques (Support Vector Machines, Naive Bayes, Rough Set Theory, Random Forest and Random Tree) and a simple neural network. In this study, the neural network outperformed the other five machine learning algorithms with an accuracy of 99.76%, The performance of SVM was close with an accuracy of 99.75% and the other four machine learning techniques had an approximate accuracy of 98% each.

Clustering is also used to detect phishing emails. The algorithm puts vectors together, if the vectors are less than a certain threshold. Using this method it was discovered that 90% of the sites detected were replicas of older phishing sites. The method has a low False Positive Rate at 0.08%. The author mentioned that though this method works well, since attackers are improving their malicious acts rapidly, this method might not work well when it is finally deployed. (Cui et al. 2017)

(Feng and Yue 2020) created four Recurrent Neural Networks without any manual feature selection, using only lexical features of URLs and attained an accuracy greater than 99%. It also helped create visualization techniques that resulted in the appearance of previously unseen features which can be used with traditional machine learning techniques like Random Forest algorithm to detect phishing.

(Abdelnabi, Krombholz, and Fritz 2020) used triplet Convolutional Neural Network to help facilitate visual phishing detection technique. It used a database of 9363 screenshots of 155 most trusted websites. It had a similarity metric between two websites even though they did not have same content. This approach is different from the other approaches that rely on text and URL data.

Methodology

In this section, Long Short-Term Memory (LSTM) and Support Vector Machine (SVM) algorithms will be described along with details about the datasets used in this study.

1. Long Short-Term Memory

Long Short-Term Memory (LSTM) is a modified version of the Recurrent Neural Network (RNN) to help solve the vanishing gradient problem of RNNs during back-propagation. The vanishing gradient problem occurs when the gradient becomes very small in some iterations. Since the updated gradient is very small, the starting layers of the RNN do not learn. Hence, RNN does not remember long-term sequences i.e., it has a short-term memory problem. LSTM is a solution to this problem. They have gates that help in the regulation of information. The gates learn through various data passed into them and they can either keep or discard the data, based on their importance. Hence, it only passes relevant data.

The cell state and gates are integral parts of the LSTM model. The cell state can be considered the memory of the network. It carries relevant information during sequence processing. As the model trains, information to the cell state is added or removed via the gates. Gates decide on the information that should be allowed on the cell state and they learn what information is to be considered relevant or irrelevant. The gates contain sigmoid activation functions that bring values between 0 and 1. If a number is multiplied by 0, it gets forgotten and if a number is multiplied by 1, it remains the same. Using this, the network learns which data should be kept or forgotten. The equation of the sigmoid function is given in 1.

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

where x is the input to the sigmoid function.

A $\tanh$ activation function exists in the network to help in the regulation of the flow of values throughout the network and to help ensure the values are kept between -1 and 1 to avoid an explosion of values. The equation of the $\tanh$ function is given in 2.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

where x is the input to the $\tanh$ function.

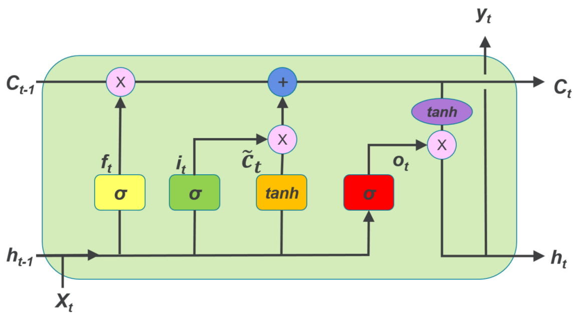

Figure 1: LSTM Cell
(Divyanshu Thakur, Medium 2021)

Three different gates help in the regulation of the flow of information in the Long Short Term Memory Cell called the forget gate, input gate, and output gate.

1) Forget gate: The forget gate decides which information should be remembered or forgotten. The previous hidden state and current input information are passed through the sigmoid function and if the value is closer to 0, the information is forgotten. If the value is closer to 1, the information is kept. The equation for the forget gate is shown in 3.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

where x_t is the input of the current layer, h_{t-1} is the output of from the previous layer (at time $t-1$), f_t is the forget gate, W_f is the weight of the forget gate neuron represented as a weight matrix and b_f is the bias of the forget gate.

2) Input gate: The input gate is used to update the cell state. Similar to the forget gate, the previous hidden state, and the current input information is passed through the sigmoid function and the value that is closer to 0 is considered not important, values closer to 1 are considered important. The previous hidden state and the current input information are passed through the $\tanh$ function as well to keep the values between -1 and 1 to help in the regulation of the network. The outputs from the sigmoid and $\tanh$ functions are multiplied. The output from the sigmoid helps decide which output from the $\tanh$ function is relevant to keep. The equation for the input gate is shown in 4.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4}$$

where i_t is the input gate, W_i is the weight of the input gate neuron represented as a weight matrix and b_i is the bias of the input gate.

Cell State: The information from the forget gate and input gate helps in the calculation of cell state. The cell state is point-wise multiplied with the forget vector, if the values are closer to 0 they are removed. This output is point-wise added to the output of the input gate and this output is the value to which the cell state updates, hence forming a new cell state. The equations for this are given in 5 and 6.

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5}$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{6}$$

where $\hat{C}_t$ is candidate for cell state at time t, C_t is cell state at time t, b_C is the bias of the cell state.

3) Output gate: The output gate decides what the next hidden state should be. Hidden state has information of previous outputs and is used for predictions. The previous hidden state and current input information is passed through the sigmoid function and then the new cell state is passed through the $\tanh$ function. The outputs of the sigmoid and $\tanh$ functions helps decide on which information the hidden state carries. This new hidden state and new cell state is carried over for the next update. The equation for the output gate is shown in 7 and next update value is shown in 8.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t * \tanh(C_t) \tag{8}$$

where o_t is the output gate, W_o is the weight of the output gate neuron represented as a weight matrix, b_o is the bias of the output gate.

We use LSTM to detect phishing emails because the LSTM model extracts valid features from the dataset, while other machine learning algorithms find it difficult to do so. LSTM can capture data requiring long-term dependencies and timing of data. It can work well with a large amount of data due to its strong learning ability. The prediction method for this model is also proven to be effective and it can easily solve problems that other traditional models find difficult. (Chen, Zhang, and Su 2018)

2. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm. It is widely used for classification problems. Our problem in this study is a binary classification problem. SVM creates a decision boundary called a hyperplane, that segregates a n-dimensional space into classes. N is the number of features in our dataset. There could be multiple hyperplanes chosen, but the objective is to select the hyperplane with maximum margin. A margin is the shortest distance between the observations and threshold. This process of maximization of margin helps in further classification of data points with more confidence.

Data points that are closest to the hyperplanes are called support vectors. They affect the orientation and position of the hyperplanes. Support vectors are used in the maximization of the margin of the classifier.

The output of a linear function is considered. The range of output we get is between -1 and 1. This range acts as a margin. If the output is greater than 1, it can be considered for one class, for example, phishing positive. If the output is lesser than -1, it can be considered for another class, for example, phishing negative.

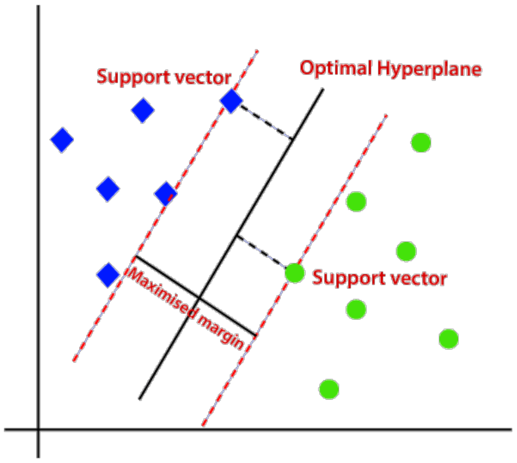

Figure 2: Functions of SVM (JavaTPoint 2021)

To maximize the margin between the hyperplane and data points, a loss function called hinge loss is used. The equation for hinge loss is given in 9.

$$c(x, y, f(x)) = \begin{cases} 0 & \text{if } y * f(x) \geq 1 \\ (1 - y * f(x)) & \text{else } c(\cdot) = (1 - y * f(x))_+ \end{cases} \tag{9}$$

where $c()$ is the loss function, x is the input, y is actual output, $f(x)$ computes the predicted output.

If the predicted and actual outputs are the same, the cost is 0. If they are different, the loss value is calculated. A cost function is added as a regularization parameter λ to balance the values of the loss and maximum margin. The equation of loss function with regularization parameter is given in 10.

$$min_w \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+ \tag{10}$$

where w is the weight value, i is the i^{th} index of the dataset.

The gradient is found by taking partial derivatives with respect to the weights. The gradient is used to update the weights in 11 and 12.

$$\frac{\partial}{\partial w_k} \lambda \|w\|^2 = 2\lambda w_k \tag{11}$$

$$\frac{\partial}{\partial w_k} (1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0 & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik} & \text{else} \end{cases} \tag{12}$$

When the model gives a correct prediction for the class in the dataset, only the gradient of the regularization parameter is updated, shown in 13.

$$w = w - \alpha \cdot (2\lambda w) \tag{13}$$

where α is the Lagrange multiplier.

When the model gives a wrong prediction for the class in the dataset, the gradient of the regularization parameter, as well as the loss function, is updated, shown in 14.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w) \tag{14}$$

SVM can be used for both linearly separable data (data that can be classified into two classes using one straight line) and non-linearly separable data (data that cannot be classified using a single straight line).

In our study, we want to compare the results of a deep learning model to a machine learning model. Compared to other machine learning algorithms, SVM is more efficient in classification problem and has been used in phishing detection in the past (Niu et al. 2017). Hence, we select SVM as an appropriate algorithm to compare the efficiency of the algorithms.

3. Dataset

Most studies consider only the email content, i.e., the email header or email body, while other studies consider only the URL. In this study, we will use both the email content as well as the URLs. The datasets we will be using for this study are:

1. SpamAssassin, a collection of spam and ham emails. Ham emails are emails that are not spam. They are authentic emails. (Mason 2005)
2. Phishcorpus, a collection of phishing emails. (Jose Nazario 2019)
3. Website phishing dataset, a collection of phishing and legitimate URLs. (Neda Abdelhamid 2016)

From the email datasets, we will be extracting 3000 emails of both phishing and ham category randomly. Spam emails are not considered in this study because they are usually advertisements, contain more content than a regular email, use unusual HTML markup, and may use coloured text.

All three datasets are public datasets. SpamAssassin and Phishcorpus are a collection of emails over several years. SpamAssassin has emails collected from 2004-2005, while Phishcorpus has emails collected from 2015-2019.

The Website phishing dataset has 1353 instances, 702 of which are phishing URLs from Phishtank (OpenDNS 2016), a website where users can report and track suspicious phishing websites, 584 are legitimate websites from Yahoo and 103 suspicious URLs. This data was collected by the creator using a web script in PHP (Abdelhamid, Ayesh, and Thabtah 2014). These datasets are widely used in studies for phishing detection. They contain sufficient information to differentiate between phishing and legitimate emails, hence are appropriate to be used in this study.

Since SpamAssassin and Phishcorpus consist of only emails, each email will be parsed to extract features such as: Email header, Sender Domain, Number of weblinks, contains JavaScript, Number of Spelling errors, Phishing indicator words. More details about this dataset is given in Table 1.

Phishing emails usually have spelling errors and improper grammatical structure, hence the words in the emails will be checked for any unusual spellings. They also contain certain words that cause a sense of urgency to the reader, for example, words like “password expired”, “bank account details”, “respond in 24 hours”, etc. These kinds of words will be

Feature	Data Type	Description
Email header	String	Header extracted from email
Sender Domain	String	Email Domain of the sender
Number of weblinks	Integer	Number of weblinks in the email
Contains JavaScript	Boolean	Indication of JavaScript present in email
Number of Spelling errors	Integer	Count of misspelled words
Phishing indicator words	List[String]	List of phishing indicator words in email

Table 1: Feature description of email dataset

added after finding the most common words from phishing emails.

The Website phishing dataset consists of 10 features which are:

1. **Server Form Handler (SFH):** In a legitimate online form, the information is processed by a server in the same domain, but in phishing websites, the server is either blank or is processed in a different domain.
2. **Using Pop-up window:** Legitimate sites do not ask users to fill details in pop-up windows, while phishing sites might ask users to fill details in pop-up windows.
3. **SSL Final:** Phishing websites might use a fake HTTPs protocol which should be verified using credible sources.
4. **Request URL:** Legitimate websites have audio and graphics from the same server while phishing websites might have these objects from servers in different domains.
5. **URL of anchor:** In phishing websites, the link within the webpage might redirect a user to another domain.
6. **Website traffic:** Phishing websites have very low website traffic because they are usually newly made and have a short life, while authentic websites usually have high website traffic. This can be checked on the Alexa database.
7. **URL length:** Attackers hide suspicious parts of a URL to make the URL look more authentic. In this dataset, if the URL length is greater than 54, it is considered a phishing URL.
8. **Age of domain:** Since phishing websites are newly made and have a short life, websites existing from less than a year are considered as phishing websites in this dataset.
9. **IP address:** Attackers usually hide IP addresses in the URL. This is usually used when attackers try to obtain personal information. A URL with an IP address can be considered a phishing URL.
10. **Result:** The phishing URL is labelled -1 and, the authentic URL is labelled 1.

All the features are labelled -1,0 and 1 which corresponds to Phishing URL, Suspicious URL, and Legitimate URL respectively. The website phishing dataset has been widely used in many studies where the URL is considered and has an appropriate size and feature vector. It can be used along with the email dataset by converting the URLs in the emails according to the features. The limitation of this dataset is that it does not contain the actual URLs used to extract the features. If the actual URLs were mentioned, it would improve the accuracy of text-based phishing email detection.

Dataset pre-processing

Email dataset:

1. Tokenization: In this step, we split all sentences in the email body into words.
2. Lower casing and Stop words removal: The tokenised words are all lowercased because “Email” and “email” are two different words according to a machine. All stop words are removed from the tokenised data because stop words like “a”, “the”, “of” do not convey any special meaning to our dataset.
3. Lemmatization: The word is transformed into the base form of the word. For example, “charging” is changed to “charge”. This helps reduce the word vector space.

Website phishing dataset:

1. Removing suspicious URL: The features with result having suspicious URLs are removed because in this study we only consider phishing and legitimate URLs.
2. Adding to email dataset: The phishing URLs will be concatenated to the phishing email part of the dataset and the legitimate URLs will be concatenated to the ham emails, both in random order. This is to create one dataset with important features to be trained using the models specified.

Results

Experimental Design

As mentioned in the previous section, the datasets need to have features extracted and need to undergo pre-processing. This will be implemented using Python built-in libraries. Natural Language Toolkit (NLTK) and Word2vec Python libraries will be used to extract words and create the feature vector from the SpamAssassin and Phishcorpus datasets. With the help of Dataframe and Pandas libraries, the datasets can be easily accessed and modified according to the pre-processing steps. On completion of the previous steps, we will have one dataset with email dataset features and URL dataset features combined. All features containing words will be tokenized using one-hot encoding to make the training process easy and efficient. Python libraries like Scikit-learn, Keras, and TensorFlow will be used to implement the SVM and the LSTM model.

LSTM: The LSTM model will consist of 5 layers and 128 nodes. The number of input layer nodes will be set to 16 as the number of features in our dataset is 16. There will be one output layer node that will give us the predicted label using the sigmoid function. The optimization method used in this algorithm is the stochastic gradient descent using an initial learning rate of 1000. The batch size will be set to 128. The learning rate of this network will be considered as 0.1. These parameters are based on the study by (Chen, Zhang, and Su 2018) which had an accuracy of 99.1%.

SVM: The training data will be input into the SVM model with a learning rate of 0.0001 and the regularization parameter λ will set to 1/epochs, following the algorithm mentioned in the previous section.

The dataset will be split into 60% training data, 10% validation data, and 30% testing data randomly. The dataset is split up randomly so that we can have an unbiased evaluation of the performance of the prediction.

Based on the output, each email can be classified into four categories.

1. True Positive (TP): The output is correctly classified as phishing.
2. True Negative (TN): The output is correctly classified as legitimate.
3. False Positive (FP): The output is incorrectly classified as phishing.
4. False Negative (FN): The output is incorrectly classified as legitimate.

Using these categories, we conduct performance evaluation using standard measures like Accuracy, Recall, and Precision. They are calculated using 15, 16 and 17 respectively.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{15}$$

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

$$Precision = \frac{TP}{FP + TP} \tag{17}$$

The reason to use these metrics is that Accuracy might not always be the best metric to select the best model in a binary classification problem. Recall determines the Actual Positives captured by the model. For example, if a phishing email is labelled as legitimate, the model will fail its purpose. Precision determines when the cost of FP is high. For example, if a legitimate email (FP) is labelled as a phishing email, a user might lose an important email. Hence, it is important to have these performance metrics to thoroughly examine the models.

Experimental Results

The two datasets (SpamAssassin and Phishcorpus) were both pre-processed. Since SpamAssassin and Phishcorpus were stored in different formats and file types, the extraction process of the datasets was different. Different kinds of LSTM models were tested to find the appropriate model for our dataset. The LSTM model is built using the Python library Keras, having 5 layers as mentioned before, and uses 100 epochs to train data. The LSTM model has a higher training time compared to a simple Convolutional Neural Network model using our dataset. The basic structure of the LSTM model is shown in Figure 3.

The LSTM model was tested by changing some of its parameters to check what parameters are suitable in this case. Three kinds of optimizers were used, Adam, RMSprop, and Adagrad, out of which Adam had the best performance with an accuracy of 95.84% and Adagrad had the worst performance with an accuracy of 77.25% as can be seen in Table 2 and Figure 4.

Few features were dropped during training as having many attributes led to the overfitting of data.

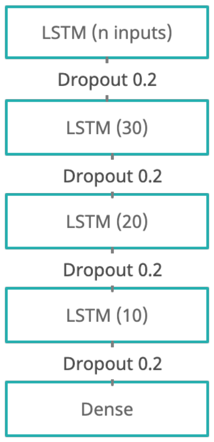

Figure 3: LSTM Structure

Optimizer	Accuracy
Adam	95.84%
RMSprop	91.09%
Adagrad	77.25%

Table 2: Optimizer Accuracy

If the number of drop out layers were removed, the accuracy was 95.70%. The number of epochs also had an effect on the accuracy, where 50 number of epochs had an accuracy of 94.71% and 10 number of epochs had an accuracy of 90.05%. In this case, more the number of training epochs, higher the accuracy.

The SVM model had an accuracy of 93.89%, with precision and recall as 94% and 92% for phishing websites and precision and recall of 94% and 95% respectively for non-phishing websites.

Metrics	Phishing Websites	Non-Phishing Websites
Accuracy	93.89%	93.89%
Precision	94%	94%
Recall	92%	95%

Table 3: Performance of SVM

The LSTM model had an accuracy of 95.84%, with precision and recall as 95% and 97% for phishing websites and precision and recall of 97% and 94% respectively for non-phishing websites. This model had Adam as optimizer. Compared to prior work that had an accuracy of 99.1% (Chen, Zhang, and Su 2018), the LSTM model used for our custom dataset had a smaller accuracy with a difference of approximately 3%. The training loss and accuracy of the LSTM model is shown in Figure 5.

The lesson learned during this experimentation is to use a more powerful computer to train the data faster. The dataset features should be carefully selected as they can affect the performance to a great extent. The number of epochs while

Metrics	Phishing Websites	Non-Phishing Websites
Accuracy	95.84%	95.84%
Precision	95%	97%
Recall	97%	94%

Table 4: Performance of LSTM

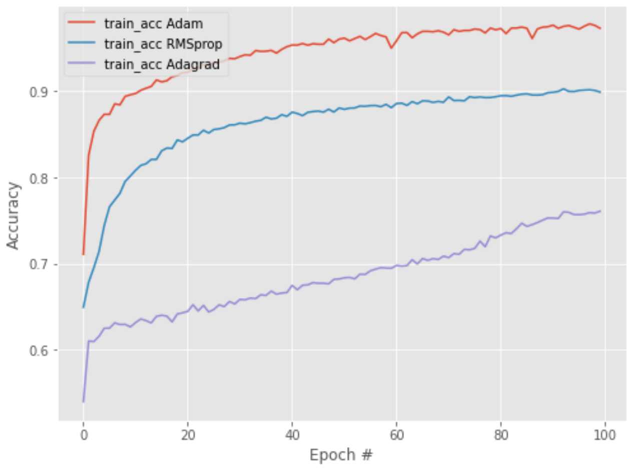

Figure 4: Accuracy of different optimizers

training a dataset should not be very less and Adam is a better optimizer compared to Adagrad and RMSprop.

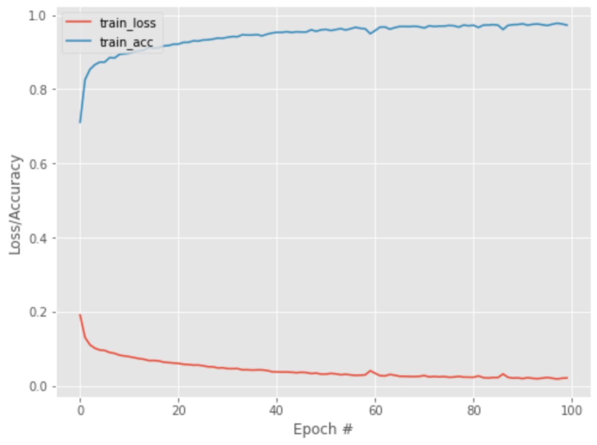

Figure 5: Training Accuracy and loss of LSTM model

References

Abdelhamid, N.; Ayesh, A.; and Thabtah, F. 2014. Phishing detection based associative classification data mining. *Expert Systems with Applications* 41(13):5948–5959.

Abdelnabi, S.; Krombholz, K.; and Fritz, M. 2020. Visualphishnet: Zero-day phishing website detection by visual similarity. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 1681–1698.

APWG. 2021. Apwg phishing trends reports. *Anti-Phishing Working Group, https://apwg. org/trendsreports*.

Chen, W.; Zhang, W.; and Su, Y. 2018. Phishing detection research based on lstm recurrent neural network. In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, 638–645. Springer.

Cui, Q.; Jourdan, G.-V.; Bochmann, G. V.; Couturier, R.; and Onut, I.-V. 2017. Tracking phishing attacks over time. In *Proceedings of the 26th International Conference on World Wide Web*, 667–676.

Divyanshu Thakur, Medium. 2021. Lstm and its equations. [Online; accessed February 20, 2021].

Fang, Y.; Zhang, C.; Huang, C.; Liu, L.; and Yang, Y. 2019. Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism. *IEEE Access* 7:56329–56340.

Feng, T., and Yue, C. 2020. Visualizing and interpreting rnn models in url-based phishing detection. In *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*, 13–24.

Gupta, B. B.; Tewari, A.; Jain, A. K.; and Agrawal, D. P. 2017. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications* 28(12):3629–3654.

JavaTPoint. 2021. Support vector machine algorithm. [Online; accessed February 20, 2021].

Jose Nazario. 2019. Phishcorpus. [Online; accessed February 20, 2021].

Li, L.; Helenius, M.; and Berki, E. 2012. A usability test of whitelist and blacklist-based anti-phishing application. In *Proceeding of the 16th International Academic MindTrek Conference*, 195–202.

Mason, J. 2005. The apache spamassassin public corpus. *URL: http://spamassassin. apache. org/publiccorpus*.

Moradpoor, N.; Clavie, B.; and Buchanan, B. 2017. Employing machine learning techniques for detection and classification of phishing emails. In *2017 Computing Conference*, 149–156. IEEE.

Neda Abdelhamid. 2016. Website phishing dataset. [Online; accessed February 20, 2021].

Niu, W.; Zhang, X.; Yang, G.; Ma, Z.; and Zhuo, Z. 2017. Phishing emails detection using cs-svm. In *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, 1054–1059. IEEE.

OpenDNS, L. 2016. Phishtank. *URL: https://www. phishtank. com/index. php*.

Paliath, S.; Qbeitah, M. A.; and Aldwairi, M. 2020. Phishout: Effective phishing detection using selected features. In *2020 27th International Conference on Telecommunications (ICT)*, 1–5. IEEE.

Sheng, S.; Wardman, B.; Warner, G.; Cranor, L.; Hong, J.; and Zhang, C. 2009. An empirical analysis of phishing blacklists.

Smadi, S.; Aslam, N.; and Zhang, L. 2018. Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. *Decision Support Systems* 107:88–102.