Employing Machine Learning Techniques for Detection and Classification of Phishing Emails

*Naghmeh Moradpoor *(1st auther)*
School of Computing (SoC)
Edinburgh Napier University (ENU)
Edinburgh, UK
*n.moradpoor@napier.ac.uk

Benjamin Clavie *(2nd auther)*
School of Computing
Edinburgh Napier University (ENU)
Edinburgh, UK
benjaminclavie@gmail.com

Bill Buchanan *(3rd auther)*
School of Computing (SoC)
Edinburgh Napier University (ENU)
Edinburgh, UK
b.buchanan@napier.ac.uk

***Abstract*—A phishing email is a legitimate-looking email which is designed to fool the recipient into believing that it is a genuine email, and either reveals sensitive information or downloads malicious software through clicking on malicious links contained in the body of the email. Given that phishing emails cost UK consumers £174m in 2015, this paper proposal is driven by a problem whose resolution will have a great impact on people's lives in the UK and in the world. In this paper, we proposed a Neural Network (NN)-based model for detections and classifications of phishing emails using publically available email datasets for both benign and phishing emails. The results of the experiments are presented in order to demonstrate the effectiveness of the model in terms of accuracy, true-positive rate, false-positive rate, network performance and error histogram.**

Keywords— Intrusion Detection and Classification, Phishing Emails, Spam Emails, Machine Learning, Artificial Intelligence, Neural Networks, Cybersecurity, Cyberattacks, Web Attacks

I. INTRODUCTION

The Raytheon kill chain model outlines that an attack goes through the stages of Reconnaissance; Weaponization; Delivery; Exploitation; Installation; Command & Control; and Actions on Objectives [17]. While many different types of malware exist, the usage of phishing emails is one of most popular methods of delivering and comprising hosts and user accounts. Normally this involves tricking the user with a valid looking email message which tricks them into entering their user credentials, or to open a document which contains a malicious element.

The phishing emails normally contain graphics, text or design elements that tricks the user into thinking that it is from a credible and trusted source. Overall there are very few elements of an email that can be truly trusted, especially without examining the SMTP header information, as the sender of the email can be easily changed, and email messages can be mocked up with the same design elements as the spoofed organization. The usage of secure emails with signatures has never really taken-off, and thus there are very few visual signs that an email may be a phishing email. So while protocols such as Pretty Good Privacy (PGP) provide the authentication of the sender of an email, the usage of digital certificates and key rings are often cumbersome.

Recent cases of phishing emails from the HMRC in the UK [18] has shown that the phishing emails are often well crafted and increasingly targeted, with the usage of the Cascading Style Sheet (CSS) and graphics from the spoofed organization, and where the user is tricked into clicking on a link which redirects to a malicious site.

One example of tricking the user is to inform them that their user details has changed on their corporate email account, and for them to log-in to review the changes. Once they click on an obfuscated link, they are re-directed to the malicious site, which gathers their details, and then redirects them back to the corporate site. As far as the user is concerned they had just put in the incorrect details, but have just given away their login credentials.

In this paper, we employed word embedding or vectorisation [4] and proposed a neural network-based model for detection and classification of phishing emails. Our model made from six components and uses six features and ten-fold cross validation for training, validation, and testing. The input features are extracted from two publically available email datasets for both benign and phishing emails.

The remainder of this paper is organised as follows. In Sections II, we review the related work for the phishing email detections and classifications. Our proposed neural network-based model for detection and classification of phishing emails along with the implementations are detailed in Sections III which is trailed by the captured results in section IV. This is followed by conclusions of the work in Section V, acknowledgment, and references.

II. RELATED WORK

In this section, existing work related to phishing email detection and classification techniques and methodologies are addressed as follows.

In [6], the authors proposed a method to detect and filter phishing emails by employing Stochastic Learning-Based Weak Estimators (SLWE) in real life environment. SLWE approach was studied and implemented based on Naive Bayes classification for filtering phishing emails that are unpredictable in nature. They used two different datasets: 1,200 real benign emails and 600 real phishing emails. To evaluate the effectiveness of their proposal, they compared their captured results from SLWE approach with Maximum Likelihood Estimator (MLE). MLE is a popular and widely used estimation scheme. Their results revealed that SLWE-based Naive Bayes approach outperforms the MLE scheme regarding accuracy. However, their proposed method suffers from an enormous number of features, which can affect system performance, and unlimited training, which can consume large amounts of storage.

In [7], the authors proposed a lexical URL analysis technique in order to enhance the classification accuracy for phishing emails. In their proposal, which is a continuous work to their previous publications [8-9], they constructed two feature sets of 47 features and 48 features. Then, they ran feature subsets on the two feature sets. The idea behind running feature subsets on two feature sets is to stop unnecessary features from increasing the time and space complexity of the classifier. This also stops the accuracy degradation for the classifier. They used the publicly available benign and phishing datasets in order to evaluate their proposal. This includes 4,150 benign emails and 4,116 phishing emails. Addressing their captured results, their proposed lexical URL analysis technique proved to be effective in enhancing the classification performance.

In [10], the authors proposed a framework called Phishing Evolving Neural Fuzzy Framework (PENFF) in order to detect and predict unknown "zero-day" phishing emails. PENFF is based on adopted Evolving Fuzzy Neural Network (EFuNN). Their proposal includes: Email Dataset, Pre-processing, Email Object Similarity, EFuNN, and PENFF. In their experiments, they used 2,000 real benign emails besides 2,000 real phishing emails. They also took into account sixteen features for phishing emails each represented in binary (0 or 1). Addressing their captured results, their proposed framework proved its ability to detect phishing emails and provide classification with low error rate.

In [11], the authors proposed a real-time hybrid Neuro-Fuzzy Scheme in order to detect phishing websites and protect the customers performing online transaction. Their model takes five inputs: "Legitimate site rules", "User-behavior profile", "PhishTank", "User-specific sites" and "Pop-Ups from emails" as well as 288 features. They also applied 2-fold cross-validation for training and testing. Addressing their captured results, their proposal can be effective in detecting phishing sites with a high accuracy in real-time. Their results also offered a better performance when they compared with the previously reported research.

In [12], the authors proposed a Neural Network-based framework to predict phishing websites. They used Anti-Phishing Working Group and PhishTank in order to extract phishing website features. They employed the extracted features in order to train and test their model. They also discovered that phishing websites lived only for 2.25 days before taken down. However, they have not presented any formal results therefore it is hard to review the effectiveness of their proposed model.

In [13], the authors proposed an intelligent model to detect phishing emails by employing a pre-processing phase. The pre-processing phase extracts a set of features by taking into account different email parts and then uses J48 algorithm for classification. They used 23 features and ten-fold cross validation for training, validation, and testing. Their primary focus was to enhance the email classification accuracy by using a pre-processing phase and determine the best algorithm that can be used. For this, they compared ten different classification algorithms where random forest achieved the highest accuracy of 98.87% when the pre-processing phase applied.

In [14], the authors proposed an online phishing detection toolbar for transactions. The toolbar runs continuously in the background of Internet Explorer web browser checking all websites users request against a dataset in a real-time manner. Their proposal is a feature-based online toolbar that uses six sets of inputs. They also combined a voice generating user warning interface with a text directives and color status to detect phishing websites and alert users from phishing attacks. They evaluated their online phishing detection toolbar by using 200 phishing websites, 200 suspicious websites, and 200 legitimate websites. Reflecting on their captured results, their proposed toolbar demonstrated 96% accuracy.

In this paper, we used word embedding or vectorisation and proposed a neural network-based model for phishing email detection and classification. Word embedding is a common name for a set of language modelling and feature learning techniques in Natural Language Processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers [4]. Our proposed model built from six elements and uses six features and ten-fold cross validation for training, validation, and testing. The input features are extracted from two publically available email datasets: "SpamAssassin" [1] for benign emails and "Phishcorpus" [2] for phishing emails. In our model, we conduct email purifications which is a compulsory phase before the vectorisation stage.

III. DESIGN AND IMPLEMENTATIONS

In this section, we explain the components of our proposed neural network-based model for detection and classification of phishing emails. This includes discussions on six distinct components of: "Emails", "Email Classifier", "Email Parser", "Email Sanitiser", "Email Vectoriser", and "Neural Network Model", Figure 1. We also explain how we developed each module and what tools and techniques we have used.

A. The "Emails" Component

As depicted in Figure 1, the "Emails" component includes all the real benign and real phishing emails used for our model. We used "SpamAssassin" dataset [1] for benign emails and "Phishcorpus" dataset [2] for phishing emails. The SpamAssassin dataset is a public mail corpus which includes a selection of mail messages and suitable for use in testing spam filtering system. This dataset includes both benign and spam emails.

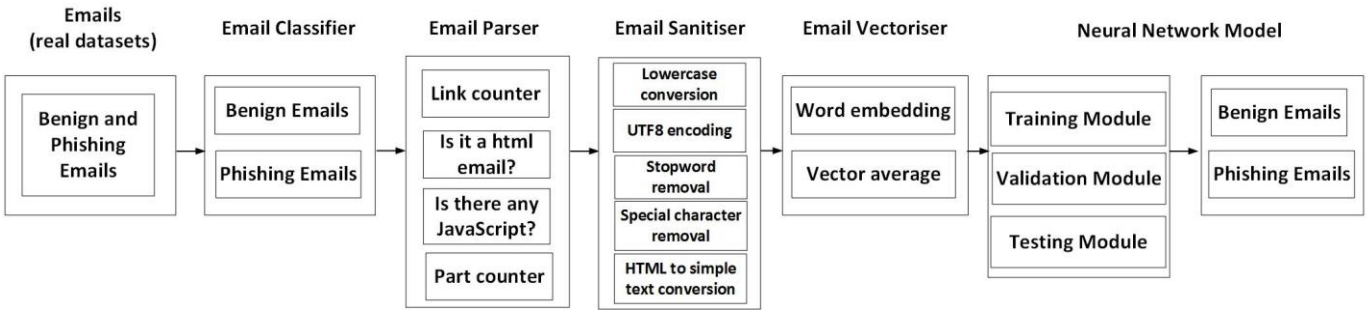

Figure 1. Components of the proposed model for detection and classification of phishing emails

In this paper, we only used the benign emails from this dataset which are identified by "_ham" suffix in [1]. This includes "_easy_ham", "_hard_ham", and "_easy_ham_2" directories. The "_easy_ham" directories include benign or non-spam messages which are quite easy to differentiate from spam messages since they do not contain any spam signatures such as HTML. The "_hard_ham" directories also include benign or non-spam messages but they are closer to typical spam messages in many respects such as: use of HTML, unusual HTML markup, colored text, and spam sounding phrases. And finally, we have "_easy_ham_2" directories which also include benign or non-spam messages but it is a more recent addition to the SpamAssassin dataset. Each directory has been compressed and the messages in SpamAssassin dataset have been named by a message number and their MD5 checksum value. There is also another directory in the path named as: "obsolete" which contains the older version of the messages but we did not use them in this paper. In this paper, we used total of 6,656 benign emails form SpamAssassin dataset.

Phishcorpus dataset is also a public mail corpus which includes a collection of phishing emails for several years with slight header modifications. They are located with ".mbox" suffix in [2]. Mbox stands for MailBox and is the most common format of storing email in hard drives. All the messages for each mailbox are stored as a single, long, text file in a string of concatenated e-mail messages, starting with the "From" header of the message. In this paper, we employed all the ".mbox" files from the Phishcorpus dataset [2]. This gives a total of 7,714 phishing emails for our implementation.

B. The "Email Classifier" Component

The "Email Classifier" component is accountable for classifying each email as either a benign email or a phishing email. Basically, this component deals with all the email datasets used in this paper. This includes the benign emails from Spamcorpus and the phishing emails form Phishcorpus datasets. We implemented a python scrip that scans through the both datasets and identifies type of each email as either a benign or a phishing where "0" represents benign class and "1" represents phishing class. For each email, the email type is then saved in a BOOLEAN variable. The emails classifier's functionality is mathematically defined as follows.

Let an email characteristic e_i is defined by a random variable E_i as follows:

$E_i =$

$$\begin{cases} 1, if\ discovered\ by\ the\ email\ signature\ detectors \\ 0, if\ not\ discovered\ by\ the\ email\ signature\ detector \end{cases}$$

Let *C* be a random variable indicating a given email class which can be either benign or phishing:

$C \epsilon$ *{benign, phishing}*

Each email (benign/phishing) is assigned with a vector defined by $e^{-} = (e_1, e_2, , \ldots, e_n)$ with e_i being the result of the *i-th* random variable E_i. This is also called features.

C. The "Email Parser" Component

The "Email Parser" component, Figure 1, is accountable for parsing a given email in order to find:

1) number of the web links;
2) if the email is an HTML email or a simple text;
3) if there is any JavaScript in the email; and
4) number of the email's parts (e.g. attachment, HTML text, plain text, and so on).

In our implementation, the email parsing procedure is done by coding a Python script. For each email in each dataset, the script pinpoints the number of the web links in the email body, whether it is an HTML email or a simple text, whether there is any JavaScript in the email's body or not, and number of the email's parts. Then, the number of the web links is saved in an INTEGER variable and if the email is an HTML email, the logical value of "1" will be assigned to a BOOLEAN variable. However, if the email is a simple text message, this BOOLEAN variable will be assigned with "0". Likewise, if the email contains any JavaScript, the logical value of "1" will be assigned to another BOOLEAN variable. And finally, the number of the email's parts is counted and saved in another INTEGER variable.

These INTEGER and BOOLEAN values are the four features that help our neural network-based model to distinguish between a benign email and a phishing email through the learning process.

D. The "Email Sanitiser" Component

The "Email Sanitiser" component, as shown in Figure 1, purifies a given email and makes it ready for vectorisation. This includes: uppercase to lowercase conversion, UTF8

encoding, stopword (e.g. 'to', 'the', 'a', and 'an') removal, special character (e.g. '£', '$', '*', and '&') removal, and HTML to simple text conversion.

In our implementation, the email sanitation is done by developing a Python script that probes and then purifies each and every single email that we used in this paper. This includes all the benign emails from Spamcorpus dataset and all the phishing emails from Phishcorpus dataset. After the sensitization, a given email will be converted into a list containing all the remaining words which is now ready for vectorisation.

The email purification is compulsory for vectorisation, which is also known as word embedding, where words or phrases from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary size [4]. Vectorisation process is detailed in the next section.

E. The "Email Vectoriser" Component

The "Email Vectoriser" component is accountable for two tasks: 1) word embedding/vectorisation and 2) vector average calculation. Word embedding/vectorisation is the collective name for a set of language modelling and feature learning techniques in Natural Language Processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers [4].

For task1, the "Email Vectoriser" component used Word2Vec [5] which is a group of related models that are used to produce word embedding. These models are two-layer neural networks that are trained to reconstruct linguistic contexts of words. In our implementation, Word2Vec takes the message, which is a list containing all the words that remain after the sanitisation, and produces a high dimensional space, with each unique word in the list being assigned a corresponding vector in the space. Once all the words have been vectorised, the "Email Vectoriser" component runs the task2 which sums the vectors for a message and calculates the average. The average will be saved in a LONG variable.

To do task1 and task2 successfully, we imported the Word2Vec module into our Python script and added extra codes in order to: 1) vectorise each message and 2) calculate the vector average, respectively. This is done for all the benign emails from Spamcorpus dataset and all the phishing emails from Phishcorpus dataset

Once we pass through all the five modules of: "Emails", "Email Classifier", "Email Parser", "Email Sanitiser" and "Email Vectoriser" for a given email, all the variables that indicate: the email type (benign, phishing), the number of web links in the email body, whether or not the email is a html email, whether or not there is JavaScript in the email, the number of the email's parts, and the vector average will be imported into our ".csv" dataset. The ".csv" dataset will be used for training, validating and also testing our neural network-based model. This is discussed in the next section.

F. The "Neural Network Model" Component

The "Neural Network Model" component, Figure 1, deals with the emails that have already been: classified into benign or phishing by the "Email Classifier", parsed by the "Email Parser", purified by the "Email Sanitiser", and finally vectorised by the "Email Vectoriser". All the emails then feed into our Neural Network (NN) model in the form of .csv dataset. For each email, our CSV dataset carries six features: vector average; the number of the web links in the email body; whether the email is an HTML email or not; whether there is JavaScript in the email or not; number of the email's parts; and email type (benign/phishing).

A given NN model includes x inputs and y outputs connected by direct arrows via n hidden layers or neurons. The arrows are single arrows which are pointed from left to right (x towards y), Figure 2 [15]. Each arrow can have different value. The values are called connection weights or simply weights. There are many algorithms that can help a neural network model to learn the weights. Generally, a neural network model starts with two sets of data: a set of random inputs and a set of desired outputs. On the first run, the NN model takes the inputs and generates a random set as outputs. Obviously as the weights are selected randomly in the first round, there will be a difference between the generated outputs and the desired outputs. The difference is called the network error [16]. When the NN model identifies the error, it tries to adjust the weights in order to generate outputs closer to the desired outputs. The process continues until the NN model produces outputs which have the smallest error when it compares with the desired outputs.

In our proposal, the neural network model has three modules: training, validation, and testing. For simplicity we combined validation and testing modules. They are defined as follows.

1) Training module

The Training module includes three components of: "Input Matrix", "Target Matrix", and "Fitness Network" as follows.

- "Input Matrix": this matrix contains all the benign emails from Spamcorpus dataset and all the phishing emails from Phishcorpus dataset that the NN model uses in training stage. These emails have been already: parsed by the "Email Parser", sanitised by the "Email Sanitiser", and vectorised by the "Email Vectoriser", Figure.1. In our implementation, this matrix is a logical *14,370 x 5* matrix which represents a matrix with *14,370* rows and *5* columns. *14,370* represents the total number of the emails in our implementation, which is *6,656* for benign

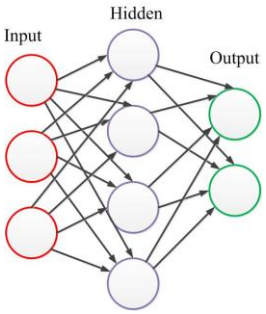

Figure 2. An artificial neural network model [15]

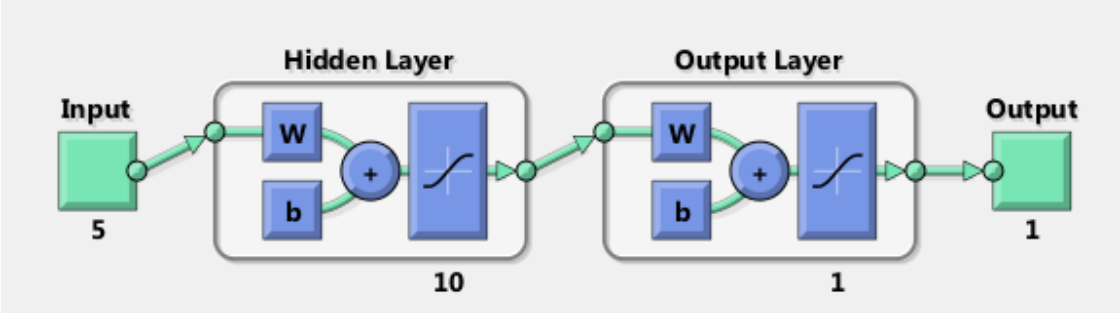

Figure 3. Network Architecture for the proposed Neural Network model

emails and *7,714* for phishing emails precisely. 5 represents the size of the assigned vectors to the emails which carries five features for each email: the number of links in the email body, whether or not the email is an HTML email, whether or not there is JavaScript in the email, the number of the email's parts, and the vector average.

- "Target Matrix": this matrix includes all the decisions (benign or phishing) for all the emails. These decisions are for each and every email stored in the "Input Matrix". In our implementation, this matrix is a logical *14,370 x 1* matrix where *14,370* represent the total number of the emails while *1* represents the size of the assigned decision vector to each email which either carries 0 (benign) or 1(phishing) as a value.
- "Fitness Network": this is the NN model with *n* layers with *x* inputs and *y* outputs where the data from 'Input' and 'Target' matrixes are used for training, validation, and testing, respectively. In our implementation, our NN model has 10 hidden nodes or 10 layers/neurons where 70% of the data from 'Input' and 'Target' matrices are used for training, 15% for validation, and 15% for testing

2) Validation and Testing modules

The Validation and Testing modules of the NN model includes two components of "Sample Matrix" and "Output Matrix" as follows.

- "Sample Matrix": this matrix contains sample data from the "Input Matrix". The trained NN model uses the data in the "Sample Matrix" as inputs during the testing phase. In our implementation, this matrix is a logical n x 5 matrix contains n sample data from the "Input Matrix".
- "Output Matrix": this matrix contains output data for the data in the "Sample Matrix". The trained NN model predicts the output values for the "Sample Matrix" and stores them in the "Output Matrix". In our implementation, this matrix is a logical n x 1 matrix contains output data for the emails represented in "Sample Matrix". The trained NN model predicts the output value, in terms of an email being benign or phishing, for each email in the "Sample Matrix". These predictions will be stored in the "Output Matrix" and will be used to evaluate the performance of the neural network.

We used 70% of the entire dataset, which includes all the benign emails from Spamcorpus dataset and all the phishingemails from Phishcorpus dataset, for training, 15% for validation and 15% for testing. We used MATLAB [3] to develop, train, validate and then test our neural network model. Our developed NN model has 10 hidden layers, 5 input features, 1 output layer, and 1 output features, Figure 3. The captured results are discussed in the next section.

IV. RESULTS

The two datasets, which includes *14,370* emails and *14,370* decisions (benign/phishing), are used in order to train, validate, and test our neural network-based model for detection and classification of phishing emails. We used 70%, 15%, and 15% of this data for training, validation, and testing, all respectively. As it was discussed in the previous section, we implemented our proposal in MATLAB [3]. Our NN model has 10 hidden layers, 5 input features, 1 output layer, and one output features (Figure 3). The results are captured, represented, and analyzed in in terms of: Confusion Matrix; Receiver Operating Characteristic (ROC); Network performance; and Error Histogram, as follows.

The Confusion Matrices for all three phases of training, validation, and testing are illustrated in figures 4 to 6, each respectively. We also illustrate the overall Confusion Matrix for all three phases in Figure 7. Addressing our proposal, we have two output classes and two target classes: Class 0 which represents benign emails and Class 1 which represents phishing emails. For each class, the number of the correct responses is presented in a green square and the number of the incorrect responses is presented in red square. The grey square represents the percentages of the accuracies (upper numbers) and inaccuracies (bottom numbers) for output and target classes. The blue square displays the overall percentages of the accuracies (upper numbers) and inaccuracies (bottom numbers) for each phases of training, validation, and testing.

For instance in Figure 7, which shows the overall Confusion Matrix for all three phases, our model was successful to classify 6,237 benign emails and 7,015 phishing

emails correctly. By taking into account the initial 6,656 benign emails and the initial 7,714 phishing emails, this classification gives us 89.9% and 94.4% accuracies and 10.1% and 5.6% inaccuracies for benign and phishing email classifications, both respectively. Therefore, the overall accuracies and inaccuracies come to 92.2% and 7.8%, respectively.

The Receiver Operating Characteristic (ROC) curves for three phases of: training; validation; and testing, are shown in Figures 8 to 10, respectively. We also took the overall ROC curve in Figure 11 where all three phases are combined. The ROC curve is a plot of the true-positive rate or sensitivity against the false-positive rate or specificity. In our implementations, the true-positive rate is the percentages of the benign emails correctly classified as benign and the percentages of the phishing emails correctly classified as phishing. Additionally, the false-positive rate is the percentages of the benign emails incorrectly classified as phishing emails and the percentages of the phishing emails incorrectly identified as benign emails.

A perfect neural network would show points in the upper-left corner, with 100% sensitivity (i.e. predicting all benign emails as benign and all phishing emails as phishing) and 100% specificity (i.e. not predicting any benign email as phishing and not predicting any phishing email as benign). Addressing Figure 8 to Figure 11, our neural network performs well.

The network performance for all three phases of training, validation, and testing is depicted in Figure 12. This is measured in terms of mean squared error and is illustrated in log scale. The mean squared error is the difference between output values and target values. This is also called a network error. Thus the lower values are better, and zero means there is no error in the network. Addressing Figure 12, our model reached its best performance at almost 39 milliseconds from the start of the simulation for all three phases of training, validation, and testing.

For additional verification of the network performance, we captured the error histogram where the blue bars represent training data, the green bars represent validation data, and the red bars represent testing data. Addressing Figure 13, we realized that the highest error falls between -0.15 and 0.14 points with the maximum error at 0.04 point for the entire scenario.

V. CONCLUSION AND FUTURE WORK

In this paper, we investigated the performance of our proposed neural network-based model for detection and classification of phishing emails. For our model, we used real benign emails from "SpamAssassin" dataset and real phishing emails from "Phishcorpus" dataset. The datasets include emails with different level of difficulties. For instance, in "SapmAssassin" dataset, some of the benign emails are fairly easy to separate from phishing emails as they don't have any phishing email signatures. Conversely, other benign emails are closer to typical phishing emails in many aspects such as having: HTML markup, colored text, and phishing sounding

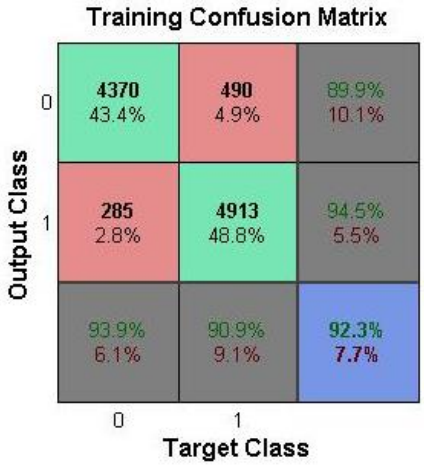

Figure 4. Confusion matrix for training phase

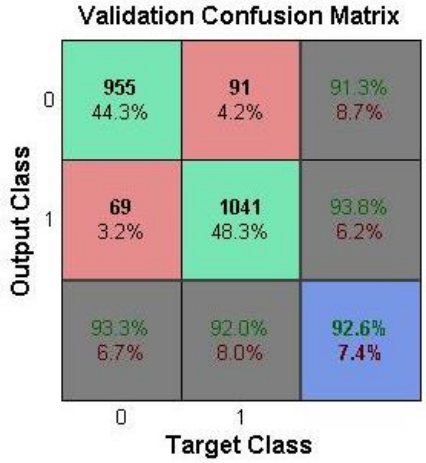

Figure 5. Confusion matrix for validation phase

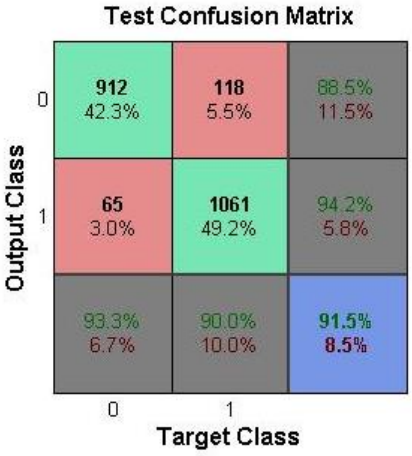

Figure 6. Confusion matrix for testing phase

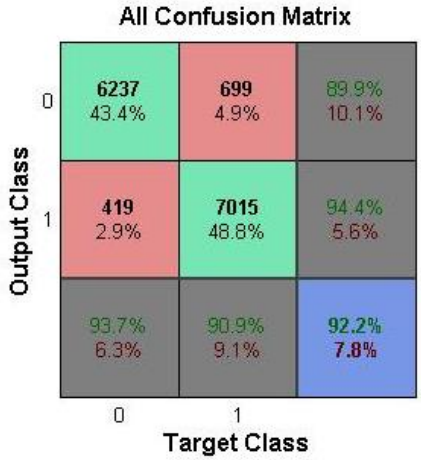

Figure 7. Confusion matrix for three phases

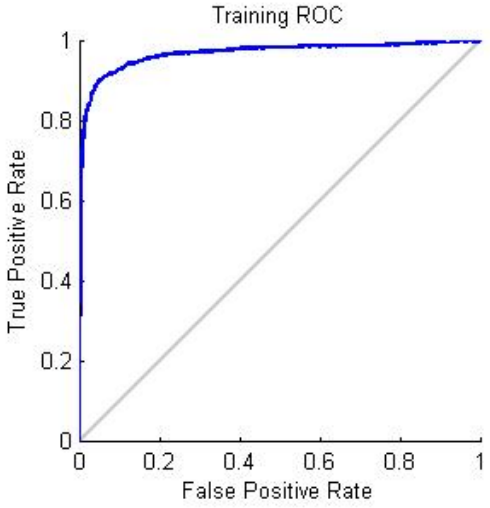

Figure 8. ROC for training phase

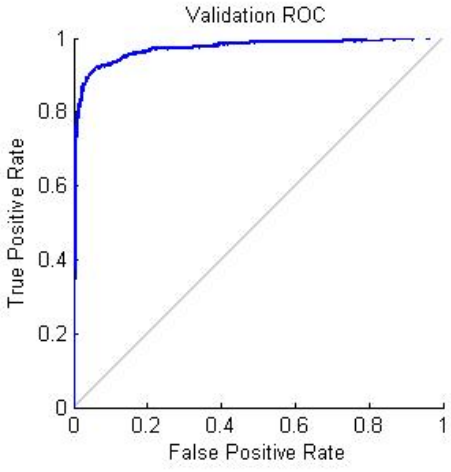

Figure 9. ROC for validation phases

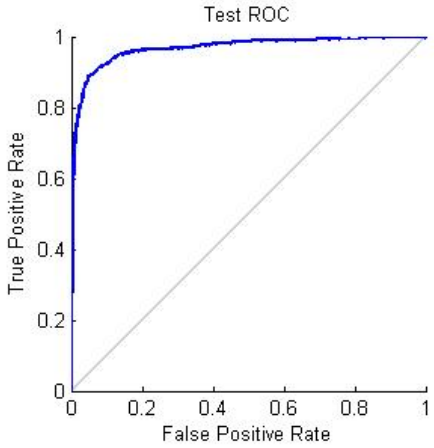

Figure 10. ROC for testing phase

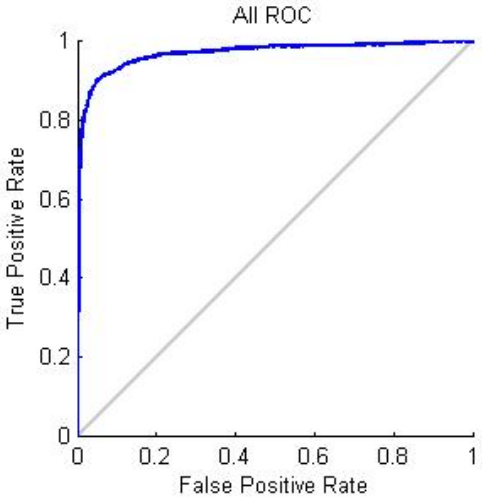

Figure 11. ROC for all three phases

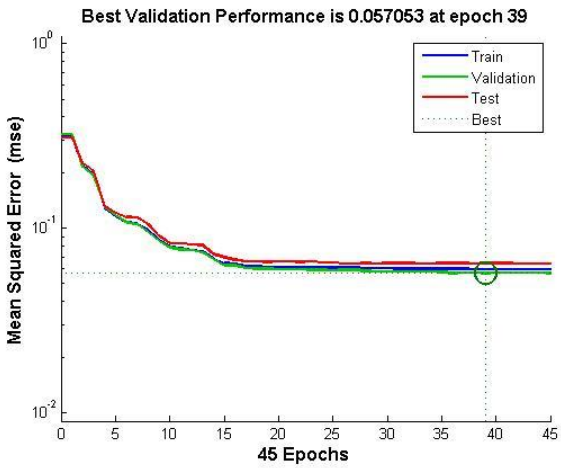

Figure 12. Network performance

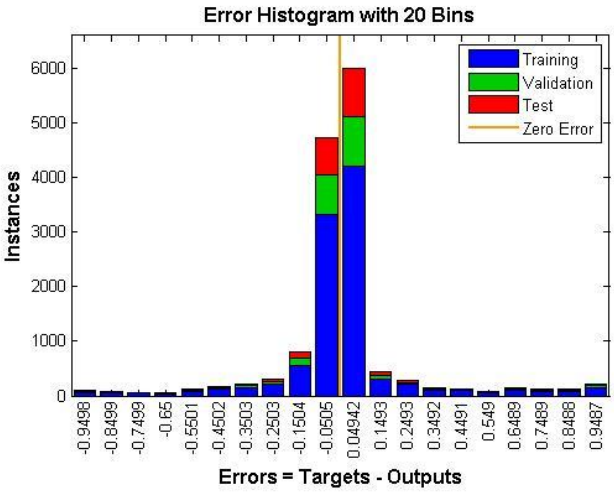

Figure 13. Error Histogram

phrases. Therefore, there was a great chance for our model to falsely detect the benign emails as phishing and vice versa. This could also lead to wrong classifications. We used Python and MATLAB to implement our model and measure the effectiveness of our proposal in terms of accuracy, true-positive rate, false positive-rate, network performance, and error histogram. The results were captured, represented, and analysed in three groups of: confusion matrix, ROC, and network performance. Addressing the captured results, our proposed neural network-based model for detection and classification of the phishing emails presented a satisfactory performance in terms of accuracy, true-positive rate, false-positive rate, and network performance.

In future work, we plan to on further developing the word embedding techniques. We intend on using alternative methods provided by the current research on word embedding in order to attempt to vectorise entire documents at once rather than averaging the vectors of the words contained in an email.

We have noticed during the writing of this paper that there currently is a lack of recent, relatively large phishing and benign e-mail datasets available publicly. Therefore, we also intend on gathering e-mails in order to create such a dataset. This would both allow us to better train our vector model on additional data as well as ensure that it is able to distinguish modern phishing emails from benign emails.

ACKNOWLEDGMENT

The authors would wish to acknowledge the support of the Edinburgh Napier University and The Cyber Academy for funding this work.

REFERENCES

[1] Spamassassin dataset; Retrieved July 14, 2016, from: https://spamassassin.apache.org/publiccorpus

[2] Phishcorpus dataset; Retrieved July 14, 2016, from https://monkey.org/~jose/phishing

[3] MathWorks, T. (1994). MathWorks – makers of MATLAB and Simulink - MathWorks United Kingdom. Retrieved July 14, 2016, from http://www.mathworks.co.uk

[4] Word embedding or vectorisation; Retrieved July 14, 2016, from: https://en.wikipedia.org/wiki/Word_embedding

[5] Word2Vec; Retrieved July 14, 2016, from: http://deeplearning4j.org/word2vec

[6] Zhan, J.Thomas, L., "Phishing detection using stochastic learning-based weak estimators," in Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on, pp. 55-59, 2011.

[7] Khonji, M.Iraqi, Y.Jones, A., "Enhancing Phishing E-Mail Classifiers: A Lexical URL Analysis Approach," International Journal for Information Security Research (IJISR), vol. 2,no.1/2, 2012

[8] Khonji, M., Iraqi, Y., and Jones, A., "Lexical url analysis for discriminating phishing and legitimate websites", In Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS '11, pages 109–115, New York, NY, USA, 2011. ACM.

[9] M. Khonji, A. Jones, and Y. Iraqi, "A study of feature subset evaluators and feature subset searching methods for phishing classification," in Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, ser. CEAS '11. New York, NY, USA: ACM, 2011, pp. 135–144.

[10] Ammar Almomani, Tat-Chee Wan, Altyeb Altaher, "Evolving Fuzzy Neural Network for Phishing Emails Detection," Journal of Computer Science, vol. 8,no.7, pp. 1099-1107, 2012.

[11] Barraclough, P.A., Hossain, M.A., Tahir, M.A., Sexton, G. and Aslam, N., 2013. Intelligent phishing detection and protection scheme for online transactions. Expert Systems with Applications, 40(11), pp.4697-4706. Vancouver

[12] Martin, A., Anutthamaa, N. B., Sathyavathy, M., Marie Francois, M. S., & Venkatesan, 2011, "A framework for predicting phishing websites using neural networks", International Journal of Computer Science Issues (IJCSI), 2(8).

[13] Smadi, S., Aslam, N., Zhang, L., Alasem, R. and Hossain, M.A., 2015, December. Detection of phishing emails using data mining algorithms. In 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA) (pp. 1-8). IEEE. Vancouver.

[14] Barraclough, P. A., Graham Sexton, and Nauman Aslam. "Online phishing detection toolbar for transactions." *Science and Information Conference (SAI), 2015*. IEEE, 2015.

[15] Moradpoor, N. (2015, September). SQL-IDS: evaluation of SQLi attack detection and classification based on machine learning techniques. InProceedings of the 8th International Conference on Security of Information and Networks (pp. 258-266). ACM.

[16] Moradpoor, N. (2015). A Pattern Recognition Neural Network Model for Detection and Classification of SQL Injection Attacks. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 9(6), 1355-1365.

[17] Raytheon Kill Chain Model; Retrieved July 14, 2016, from: http://cyber.lockheedmartin.com/solutions/cyber-kill-chain

[18] Avoid and report internet scams and phishing; Retrieved July 14, 2016, from: https://www.gov.uk/report-suspicious-emails-websites-phishing