Classifying Phishing URLs Using Recurrent Neural Networks

Alejandro Correa Bahnsen†, Eduardo Contreras Bohorquez*, Sergio Villegas†,
Javier Vargas† and Fabio A. González*
†Easy Solutions Research
*MindLab Research Group, Universidad Nacional de Colombia, Bogotá
Email: acorrea@easysol.net, econtrerasb@unal.edu.co, svillegas@easysol.net, jvargas@easysol.net, fagonzalezo@unal.edu.co

***Abstract*—As the technical skills and costs associated with the deployment of phishing attacks decrease, we are witnessing an unprecedented level of scams that push the need for better methods to proactively detect phishing threats. In this work, we explored the use of URLs as input for machine learning models applied for phishing site prediction. In this way, we compared a feature-engineering approach followed by a random forest classifier against a novel method based on recurrent neural networks. We determined that the recurrent neural network approach provides an accuracy rate of 98.7% even without the need of manual feature creation, beating by 5% the random forest method. This means it is a scalable and fast-acting proactive detection system that does not require full content analysis.**

***Keywords*—Phishing detection; Cybercrime; Feature engineering; Recurrent neural networks; Long short term memory networks.**

I. Introduction

Phishing attacks are a growing problem worldwide. According to the Anti-Phishing Working Group (APWG), phishing websites increased by 250% from the last quarter of 2015 to the first quarter of 2016, targeting more than 400 brands each month [1]. This is the most the APWG has ever seen since they began tracking and reporting on phishing in 2004. Phishing, by definition, is the act of defrauding an online user by posing as a trustworthy institution or entity in order to obtain personal information [2]. The use of phishing by a criminal is centered around using social engineering schemes to steal personal and financial information. The attacks are designed to lead consumers to reveal financial data such as usernames and/or passwords in fraudulent websites posing as legitimate entities. [1], [3].

Nowadays, phishing attacks can be launched from anywhere in the world at insignificant costs by people with little to no technical skills [4]. Organizations trying to protect their users from these attacks are having a hard time dealing with the massive amount of emerging sites, which must be identified and labeled as malicious or harmless before users can access them.

There is no shortage of methods at the time of distributing attacks, and phishers enjoy a wide range of techniques for making a site appear legitimate while evading detection [4], [5]. However, at the end of the day, all of them rely on URLs that redirect their victims to the phishing trap. Impersonating legitimate URLs is the most common social engineering method a phisher can use to lure victims to their website. Therefore, a solid first step towards blocking fraud sites is to use the URLs themselves to screen possible phishing websites [6].

Being able to determine the maliciousness of a website by simply evaluating its URL provides a major strategic advantage. The number of victims can be reduced to nearly zero while minimizing operational efforts by avoiding massive use of more complex methods such content analysis [7].

For this work, we focused on using machine learning techniques for the classification of phishing sites using only their URLs. Specifically, we compared the combination of lexical and statistical analysis of URLs as input for a random forest (RF) classifier against a novel approach that employs recurrent neural networks, more particularly, a long/short term memory network (LSTM). RFs, with manually-created features, have been widely used for classification problems [8]. Moreover, this method has been successfully applied to identifying phishing URLs [9]. On the other hand, LSTM models are competent at detecting long patterns in sequences and have been applied to solve different text analysis problems [10]. The LSTM method does not require the manual extraction of features, since it directly learns a representation from the URL's sequence of characters [11]. Recently, they have been used to detect domain-generated algorithms [12], showing great promise for the infosec industry. To the best of our knowledge, this is the first time that the LSTM model has been applied to the detection of phishing URLs.

To evaluate both approaches, we took a corpus comprised of one million phishing URLs extracted from Phishtank[1] and one million harmless URLs from CommonCrawl[2]. The results show that despite using the URLs as sole input, the RF and the LSTM methods achieved an accuracy rate of 93.5% and 98.7%, respectively. Additionally, we compared both methods in terms of training and evaluation times, and the amount of data needed to converge.

The remaining of this paper is organized as follows: In Section II, we will provide a background on the problem, as well as any related work on phishing detection. Sections

[1]PhishTank (https://www.phishtank.com/)
[2]Common Crawl (http://commoncrawl.org/)

978-1-5386-2701-3/17/$31.00 © 2017 IEEE

III and IV will provide in-depth descriptions of both machine learning methods. Subsequently, in Section V, we will describe the data and methodology used for the experiments. Section VI presents the experimental results. Finally, we will provide the conclusions of the paper in Section VII.

II. Background

A. Phishing Detection

Phishing URL detection can be done via proactive or reactive means. On the reactive end, we find services such as Google Safe Browsing API[3]. This type of services expose a blacklist of malicious URLs to be queried. Blacklists are constructed by using different techniques, including manual reporting, honeypots, or by crawling the web in search of known phishing characteristics [13], [14]. For example, browsers make use of blacklists to block access upon reaching the URLs contained in them. One drawback of such reactive method is that in order for a phishing URL to be blocked, it must be previously included in the blacklist. This implies that web users remain at risk until the URL is submitted and the blacklist is updated. What is more, since the majority of phishing sites are active for less than a day [14], [15], their mission is complete by the time they are added to the blacklist.

Proactive methods mitigate this problem by analyzing the characteristics of a web page in real time in order to assess the potential risk of a web page. Risk assessment is done through a classification model [16]. Some of the machine learning methods that have been used to detect phishing include: support vector machines [17], streaming analytics [18], gradient boosting [6], [19], random forests [20], latent Dirichlet allocation [21], online incremental learning [22], and neural networks [23]. Several of these methods employ an array of website characteristics, which mean that in order to evaluate a site, first it has to be rendered before the algorithm can be used. This adds a significant amount of time to the evaluation process [24], [25]. Using URLs, instead of content analysis, reduces the evaluation time because only a limited portion of text is analyzed.

Lately, the application of machine learning techniques for URL classification has been gaining attention. Several studies proposing the use of classification algorithms to detect phishing URLs have come to the light in recent years [6], [20], [26]. These studies are mainly focused on creating features through expert knowledge and lexical analysis of the URL. Then, the phishing site's characteristic are used as quantitative input for the model. The model in turn learns to recognize patterns and associations the inputs must follow in order to label a site as legitimate or malicious.

B. Uniform Resource Locator Structure

The Uniform Resource Locator (URL), as specified in the RFC 1738 [27], is a string representation for a resource available on the Internet.

A URL is written as follows:

$$< scheme >:< scheme - specific - part >$$

The scheme specifies the resource's access mechanism or network location (e.g. http, ftp, mailto), while the rest of the URL may vary depending on the scheme selected. For the HTTP protocol, a possible syntactic construction for the next part can be:

$$// < host >:< port > / < URL - path >$$

It starts with the domain name or IP address of a network host. Then, there is the port number to connect to and the URL-path that provides details on how the resource can be accessed (e.g. http://host.com:80/page).

III. Classifying Phishing Using URL Lexical and Statistical Frequencies

In this section, we will describe our approach to combine the lexical and statistical analysis of a URL with a random forest classifier to classify phishing websites based on URL features. The process is summarized in Fig. 1. First, a series of important variables are extracted with a feature-engineering approach. Then, a classification algorithm is used to build the model.

A. Feature Engineering

The attackers' objective when crafting a phishing URL is to trick users into thinking it is a legitimate website. In this way, the cybercriminal hopes users will reveal their personal and financial information. In order to achieve this, the attackers follow certain tried-and-true patterns, which can be detected by an experienced eye. In collaboration with security analysts at Easy Solutions, Inc.[4], we identified a set of 14 features that can be used to create lexical and statistical analysis of URLs:

- *Domain exists in Alexa rank*[5]: If the domain exists among the top one million Alexa domains. The Alexa rank is a list of domains arranged by internet popularity. Most phishing sites are hosted in hacked legitimate sites or new domains. If the phishing is hosted in a hijacked website, it is unlikely the domain is part of the top Alexa domains, since top-ranked domains tend to have better security measures. If the phishing is hosted in a newly-registered domain, the domain will not appear in the Alexa rank.
- *Subdomain length*: This takes the subdomain's URL length. Phishing sites try to mimic the legitimate site's URL by using its domain as their sub domain. Real websites tend to have a short subdomain.
- *URL length*: This takes the URL's length. A long URL increases the odds of confusing the user.
- *Path length*: This takes the URL's path length. Phishing URLs tend to have a longer path than the legitimate ones.
- *URL Entropy*: Calculates URL entropy. The higher the entropy of a URL, the more complex it is. Since phishing URLs tend to have random text, we can attempt to find them by their entropy.

[3]https://safebrowsing.google.com/

[4]https://www.easysolutions.net

[5]http://www.alexa.com/

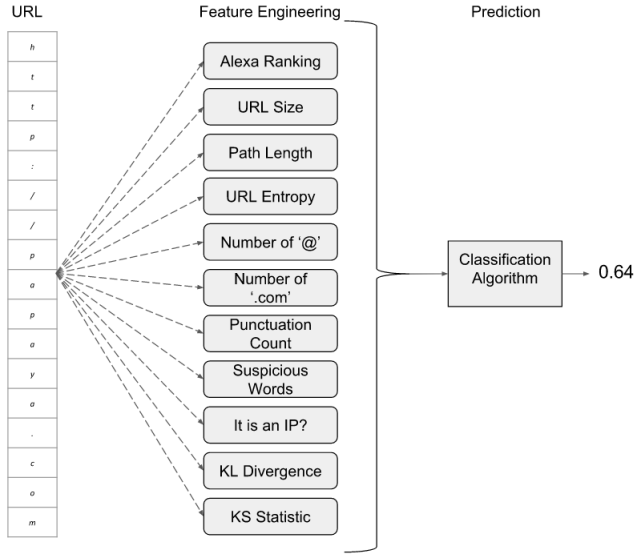


Fig. 1. Feature-engineering approach for classifying phishing URLs. First, a series of important variables are extracted with a feature-engineering approach. Then, a classification algorithm is used to build the model.

- *Length ratio*: Calculates the ratio between URL length and path length. In [20] they concluded that phishing URLs tend to have a higher ratio than legitimate URLs.
- *'@' and '-' count*: Counts @ and - characters in the URL. In accordance with [20] this feature is added. In URLs, everything to the left of @ gets ignored. In light of this, phishing URLs use it to deceive users. For example goodURL.com@phishURL.com.
- *Punctuation count*: The count of . ! # $ % & , . ; ' in the URL. In [20], they found that phishing URLs usually show a higher occurrence of punctuation count.
- *Other TLDs count*: The number of TLDs that appear in the URL's path. Phishing URLs try to impersonate legitimate URLs by using their domain and TLD in the path.
- *Is IP*: If the URL is an IP instead of a domain. It is a feature that has been used in the literature.
- *Suspicious words count*: The number of suspicious words in the URL. Suspicious words include 'confirm', 'account', 'secure', 'webscr', 'login', 'signin', 'submit', 'update', 'logon', 'secure', 'wp', 'cmd' and 'admin'. They were chosen manually by observing phishing URLs.
- *Euclidean distance*: The Euclidean distance between English characters in the URL. This feature, and the following two, attempt to measure how much the URL differs from common English.
- *Kolmogorov-Smirnov statistic*: Calculates the two-sample Kolmogorov-Smirnov statistic on character frequencies between the URL and English.
- *Kullback-Leibler divergence*: Calculate the Kullback-Leibler divergence on the character frequencies between the URL and English.

B. Classification Algorithm

Once the features are extracted, a binary classifier is trained using the presented URL features. We use a random forest (RF) [28] method to achieve this. An RF is a classification algorithm that relies on weaker models to build a stronger model with the average of the weaker model responses. The weaker models used are classification trees and each one of them recursively splits the data set based on feature values, and stops the current split when all input instances belong to the same class. We chose RF since it is widely used [8] and can be trained to run in parallel.

IV. Modeling Phishing URLs with Recurrent Neural Networks

In the previous section, we designed a set of features extracted from a URL and fed them into a classification model to predict whether a URL is a case of phishing. We now approach the problem in a different way. Instead of manually extracting the features, we directly learn a representation from the URL's character sequence.

Each character sequence exhibits correlations, that is, nearby characters in a URL are likely to be related to each other. These sequential patterns are important because they can be exploited to improve the performance of the predictors [11].

A neural network is a bio-inspired machine learning model that consists of a set of artificial neurons with connections between them. Recurrent Neural Networks (RNN) are a type of neural network that is able to model sequential patterns. The distinctive characteristic of RNNs is that they introduce the notion of time to the model, which in turn allows them to process sequential data one element at a time and learn their sequential dependencies [10].

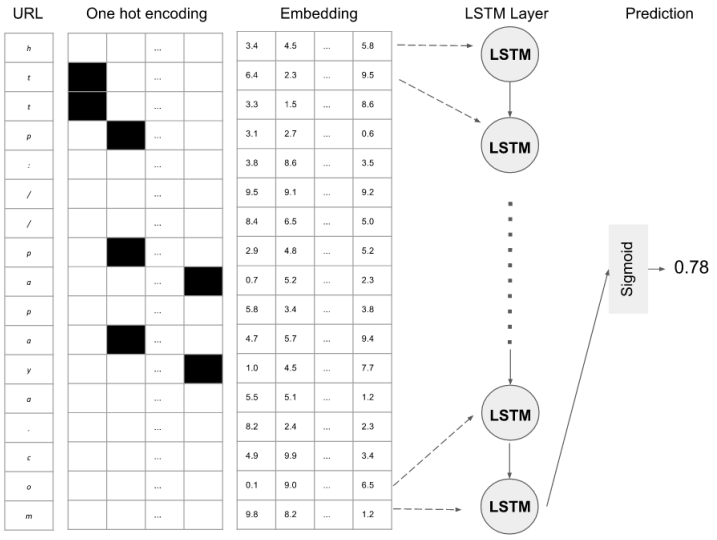


Fig. 2. Recurrent neural network for classifying phishing URLs based on LSTM units. Each input character is translated by an 128-dimension embedding. The translated URL is fed into a LSTM layer as a 150-step sequence. Finally, the classification is performed using an output sigmoid neuron.

One limitation of general RNNs is that they are unable to learn the correlation between elements more than 5 or 10 time steps apart [29]. A model that overcomes this problem is Long Short Term Memory (LSTM). This model can bridge elements separated by more than 1,000 time steps without loss of short time lag capabilities [30].

LSTM is an adaptation of RNN. Here, each neuron is replaced by a memory cell that, in addition to a conventional neuron representing an internal state, uses multiplicative units as gates to control the flow of information. A typical LSTM cell has an input gate that controls the input of information from the outside, a forget cell that controls whether to keep or forget the information in the internal state, and an output gate that allows or prevents the internal state to be seen from the outside.

In this work, we used LSTM units to build a model that receives as input a URL as character sequence and predicts whether or not the URL corresponds to a case of phishing. The architecture is illustrated in Fig. 2. 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 network is trained by backpropagation using a cross-entropy loss function and dropout in the last layer.

V. Experimental Setup

A. Data

To train both models, a dataset of real and phishing URLs was constructed. In total, 2 million URLs were used in the training process. Half of them legitimate and half of them phishing. The legitimate URLs came from Common Crawl, a corpus of web crawl data. The phishing URLs came from Phishtank, a website used as phishing URL deposit. In TABLE I, a sample of ten legitimate URLs and ten phishing URLs are shown. Note how similar the legitimate and malicious URLs can actually be. This is expected as the objective of an attacker

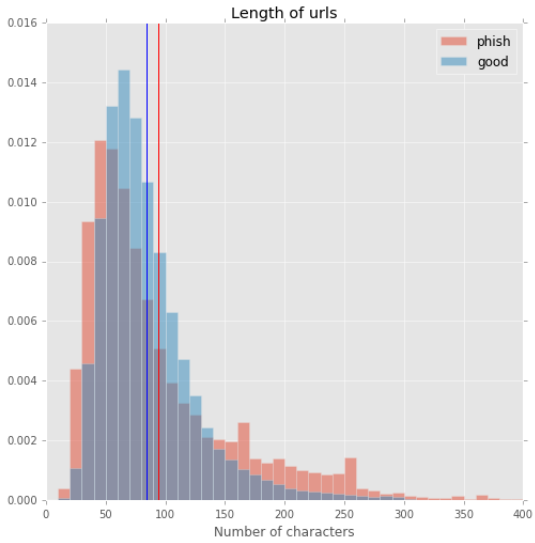


Fig. 3. URL length distribution. It is shown that the phishing and legitimate URLs have a very similar length distribution, confirming that they are quite similar and difficult to tell apart.

is to confuse web users by making the phishing site look as genuine as possible.

Moreover, in Fig. 3, a comparison of the URL length distribution is presented. It can be observed that the phishing pages tend to have slightly longer URLs (measured in number of characters).

B. Experiment Design

In order to evaluate the performance of the models, we used a 3-fold cross-validation strategy. This process consists of splitting data in 3 folds. Then train the data using two folds while the remaining one is used for model validation. This process is repeated 3 times, only using each fold for validation once. In the end, all the performance metrics on validation

TABLE I. SAMPLE OF THE RAW URL DATABASE. SHOWN IS THE SIMILARITY OF LEGITIMATE AND PHISHING URLS. THIS IS EXPECTED AS THE OBJECTIVE OF AN ATTACKER IS TO CONFUSE THE VICTIM BY MAKING THE PHISHING SITE LOOK AS HARMLESS AS POSSIBLE.

URL	Phish
http://www.cheatsguru.com/pc/the_sims_3_ambitions/requests/	False
http://www.sherdog.com/pictures/gallery/fighter/f_1349/137143/10/	False
http://www.mauipropertysearch.com/maui-meadows.php	False
https://www.sanfordhealth.org/HealthInformation/ChildrensHealth/Article/73980	False
http://strathprints.strath.ac.uk/18806/	False
http://www.grahamleader.com/ci_25029538/these-are-5-worst-super-bowl-halftime-shows	False
http://www.nwherald.com/2014/04/14/rizzo-homers-for-cubs-in-loss-to-cardinals/apxo9hf/	False
http://th.urbandictionary.com/define.php?term=politics&defid=1634182	False
http://www.carolinaguesthouse.co.uk/onlinebooking/?industrytype=1&startdate=2013-09-05&nights=2&windowsearch=0&location&productid=25d47a24-6b74-46...	False
http://www.lander.edu/Business-Administration/Human-Resources/new-employees/policies-procedures	False
http://msystemtech.ru/components/com_users/Italy/zz/Login.php?run=_login-submit&session=68bbd43c854147324d77872062349924&=68bbd43c854147324d778720...	True
http://moviesjingle.com/auto/163.com/index.php	True
http://any3.co.nz/wp-includes/Text/pp/5885d80a13c0db1f8e%26ee%3D111e61ae3eeb78bcbc5ec9fa804ee562/5885d80a13c0db1f8e%26ee%3D111e61ae3eeb78bcbc5ec9f...	True
http://paypal.com.update.account.toughbook.cl/8a30e847925afc5975161aeabe8930f1/?cmd=_home&dispatch=d09b78f5812945a73610edf3852f5ebed09b78f5812945a...	True
http://www.zeroaccidente.ro/cache/mod_login/home/37baa5e40016ab2b877fee2f0c921570/	True
http://mail.kungfuexperience.co.uk/user-verfication/216545649874az6548945648t754867t56/5959730380a7dbe17368373c106f5866	True
http://www.argo.nov.edu54.ru/plugins/system/applse3/54e9ce13d8baee95696633257b33b2b5/	True
http://rarosbun.rel7.com/	True
http://tech2solutions.com/home/wp-admin/includes/trulia/index.html	True
http://esxcc.com/js/index.htm?http://us.battle.net/login/en/?ref=http://ruuyqyrus.battle.net/d3/en/index&amp	True

folds are averaged. In this way, the variance is reduced and we can obtain a better estimate of the model's performance.

The performance evaluation is done using standard classification evaluation measures, as described below:

- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- Recall $= \frac{TP}{TP+FN}$
- Precision $= \frac{TP}{TP+FP}$
- F_1-Score $= 2\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$,

where TP and FN are the numbers of true and false negatives respectively. We define the phishing URLs as positive and the legitimate/ham ones as negative. Lastly, we also used the ROC curve to evaluate AUC statistic.

VI. RESULTS

In this section we present the experimental results. First, we evaluated the performance of the traditional feature engineering plus the classification-algorithm methodology presented in Section III. We created 14 features based on the URL's lexical and statistical analysis. Then, we trained a random forest classifier with 100 decision trees. We used the random forest implementation of the Scikit-Learn library [31].

Using the 2,000,000 URLs described above, we tested the accuracy of the model using a 3-fold cross validation strategy. The results are shown in TABLE II. The average accuracy of the model stands at 93.47%, with a recall of 93.28% and precision of 93.63%. We also evaluated the AUC of the model for each fold (ROC curves are shown in Fig. 4. In average, the models show an AUC statistic of 98.44%. Moreover, the models provide consistent results as the standard deviation of the accuracy folds is 0.01%, and 0.0008% for the AUC.

Furthermore, we analyzed which features were more important for performing classification in the random forest classifier. This is done by counting the number of times each feature was selected in the different decision trees inside the random forest. Feature importance is shown in Fig. 5. The most important feature for the algorithm is the number of suspicious words in the URL. This is not surprising since attackers will try to deceive users by employing suspicious words known by the victims. Also, it is observed that phishing URLs tend to have a higher length ratio between the length of the path and the hostname.

TABLE II. RESULTS RANDOM FOREST

Fold	AUC	Accuracy	Recall	Precision	F1-score
0	0.984499	0.934818	0.932642	0.936632	0.934633
1	0.984458	0.934782	0.932798	0.93663	0.93471
2	0.984489	0.934588	0.93302	0.935924	0.934469
Average	**0.984482**	**0.934729**	**0.93282**	**0.936395**	**0.934604**
Std dev	**1.8e-05**	**0.000101**	**0.000155**	**0.000333**	**0.0001**

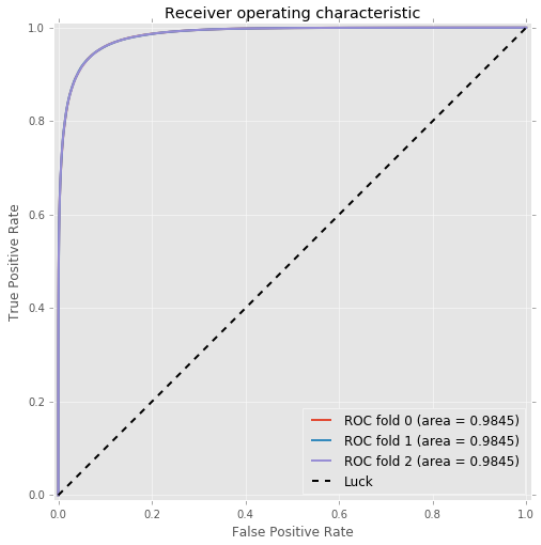


Fig. 4. ROC curve random forest classifier. In average, the models have an AUC statistic of 98.44% and a standard deviation of 0.0008%.

Afterwards, we trained the LSTM network as described in Section IV. In particular, we used the implementation of the Keras [32] with a Theano [33] backend. We defined the number of epochs to be 20, and for each fold, we used 90% of the data for training, and 10% for internal validation. In Fig. 6, the learning curve of the LSTM network is shown. It is observed that in just 20 epochs, the validation accuracy converges, increasing to over 98% from epoch 10 onward.

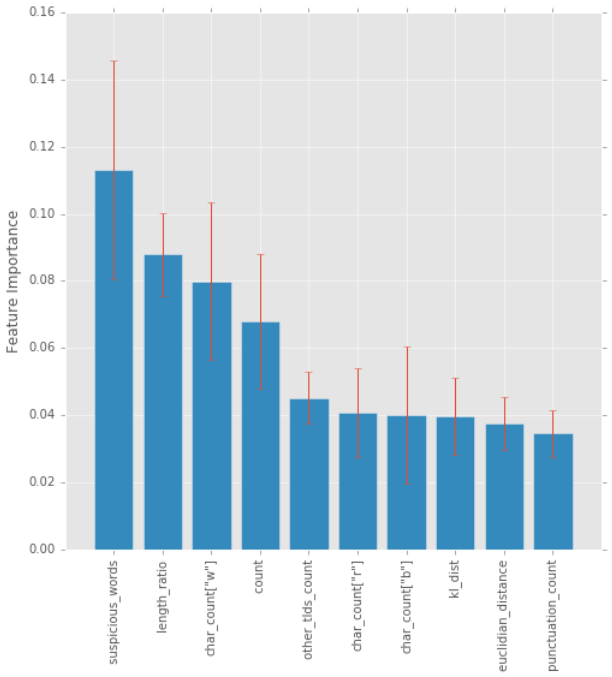


Fig. 5. Feature importance of the random forest classifier. The most important features are the number of suspicious words and the ratio of the path and hostname lengths.

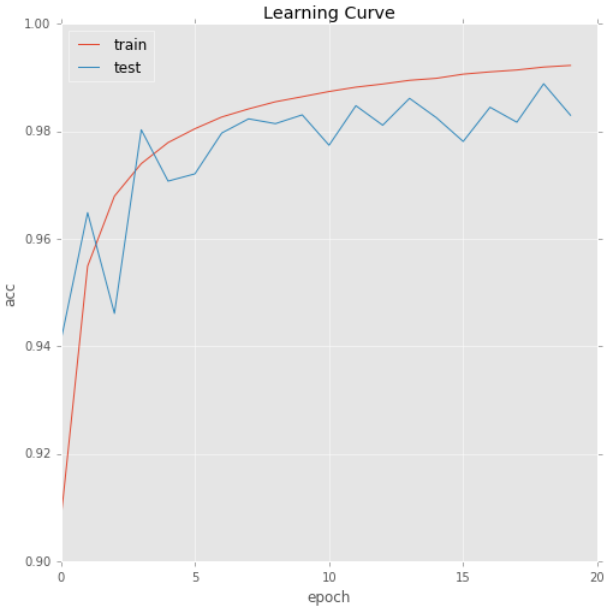


Fig. 6. Learning curve LSTM network. The learning curve shows that in just 20 epochs the validation accuracy converges, increasing to over 98% from epoch 10 onwards.

As shown in TABLE III, the LSTM model has an average accuracy of 98.76%, with a standard deviation of just 0.03%, suggesting stable performance across the different folds. Moreover, as shown in Fig. 7, the AUC of the model is very high, having an average of 99.91%.

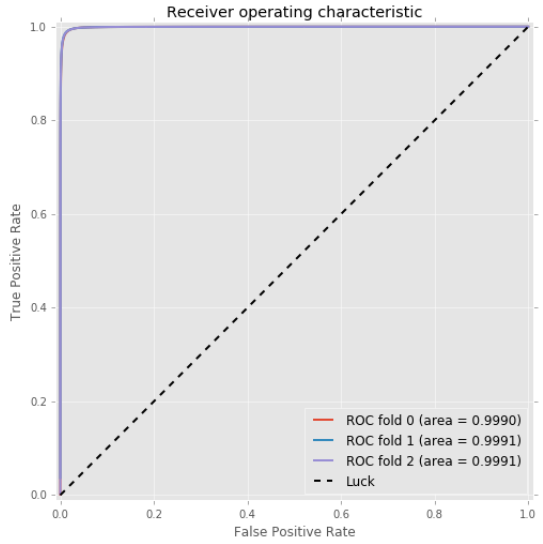


Fig. 7. ROC curve LSTM. The AUC of the model is very high, having an average of 99.91%.

TABLE III. RESULTS LSTM NETWORK

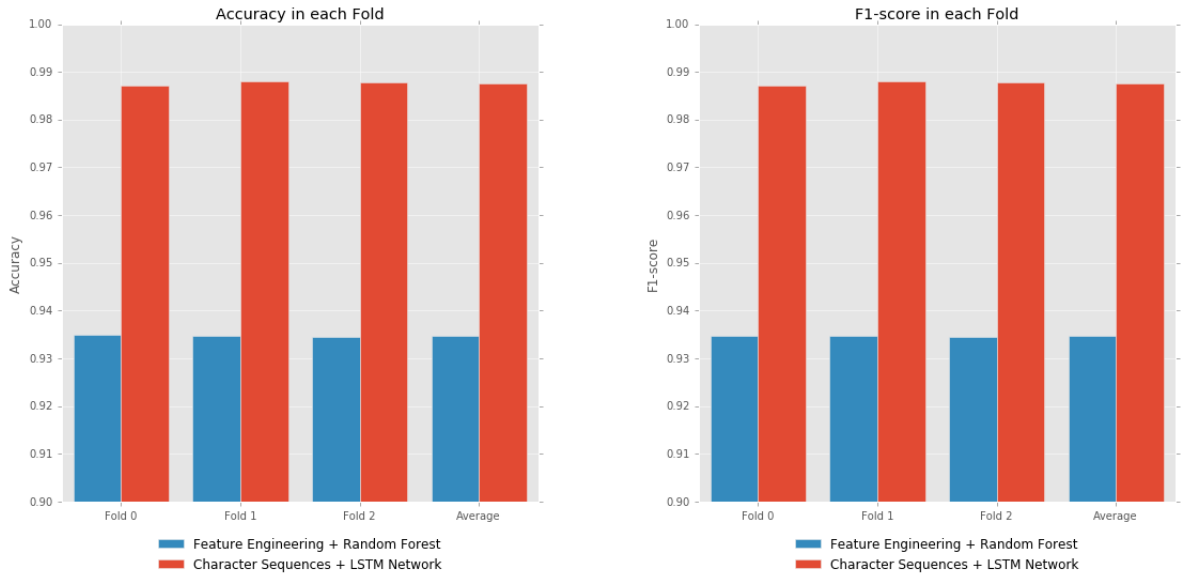
Fold	AUC	Accuracy	Recall	Precision	F1-score
0	0.999044	0.9871	0.991114	0.983203	0.987143
1	0.999106	0.987921	0.989549	0.986359	0.987952
2	0.999141	0.987844	0.98716	0.988506	0.987833
Average	**0.999097**	**0.987622**	**0.989274**	**0.986023**	**0.987642**
Std dev	**4e-05**	**0.00037**	**0.001626**	**0.002178**	**0.000357**

Comparison of the methods

First, we compared the accuracy and F1-Score of both methods. The comparisons are presented in Fig. 8a and Fig. 8b. For both measures, the difference between models is consistent across folds. In average, the LSTM network has an accuracy 5% higher than the feature-engineer model with RF. Similar results were found for the F1-Score.

We also compared the accuracy of the models using different numbers of URLs for training. For this experiment, we randomly selected 200,000 URLs and use them to test the different models. Then, we selected samples of sizes 1,000, 5,000, 10,000, 50,000, 100,000, 500,000 and 1,000,000, and trained both methods in each sample. The results are summarized in Fig. 9. The LSTM model consistently outperforms the RF model. Also, the LSTM network improves its own performance faster than the RF, as the number of training URLs increases.

Lastly, we compared the training and evaluation times for both models. The comparison took place on a Lenovo Y50 machine with 16 GB of memory, an Intel Core i7-4710HQ CPU @ 2.50GHz x8 processor, and a GeForce GTX 860M GPU. As can be seen in TABLE IV, the LSTM model requires significantly more time to train. Using the 2 million URLs, the RF is trained in an average time of less than 3 minutes. On the other hand, LSTM requires 238 minutes. It should be pointed out that once the models have been trained, the RF method is able to evaluate 942 URLs per second compared to the 281 URLs per second of the LSTM method. However, the memory requirements of the RF model are almost 500 times higher than those for LSTM. This is directy related to the complexity of storing the model parameters.



(a) Accuracy by Fold

(b) F1-Score by Fold

Fig. 8. It is observed that the LSTM method consistently outperforms the random forest in each fold. Moreover, the accuracy and F1-Score are stable between folds, which suggest a highly robust model.

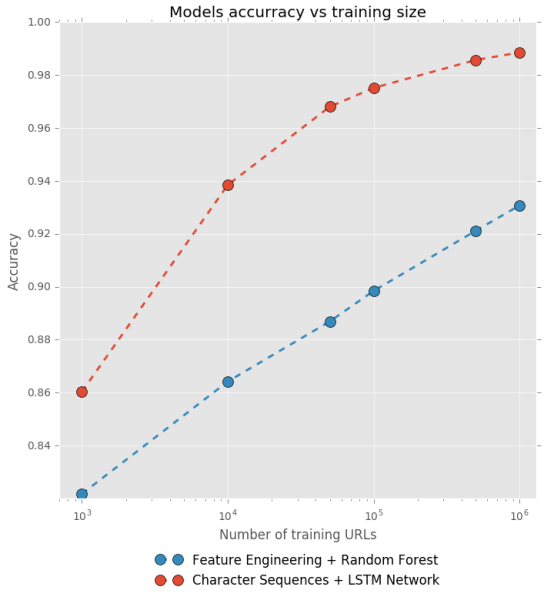


Fig. 9. Comparison of model accuracy vs number of training URLs. For this experiment, we randomly selected 200,000 URLs and use them to test the models. It is observed that the LSTM model consistently outperforms the RF model, improving its performance faster as the number of training URLs increases and arriving to a near perfect classification with less training cases.

TABLE IV. COMPARISON OF THE METHODS

Method	Training Time minutes	Evaluation Time URLs per second	Memory Consumption MB
RF	2.95 $\pm$ 0.11	942.12 $\pm$ 95.02	288.7
LSTM	238.7 $\pm$ 0.79	280.90 $\pm$ 64.48	0.581

VII. CONCLUSIONS AND DISCUSSION

We explored how well we can discern phishing URLs from legitimate URLs using two methodologies: feature-engineering with a lexical and statistical URL analysis with a random forest (RF) classifier, and a novel approach using a long/short term memory neural network (LSTM). The former has been widely used since the 1990s, the latter is a newer method within recurrent neural networks. In order to evaluate the approaches, we used a database comprised of one million legitimate URLs from the Common Crawl database and one million phishing URLs from Phishtank. Both models showed great statistical results. On one hand the RF had an F_1-Score of 0.93 and an accuracy of 93.5%, while the LSTM had F_1-Score of 0.98 and an accuracy of 98.7%.

With these results, we can conclude that discerning URLs by their patterns is a good predictor of phishing websites. The results yield that creating an URL-based proactive phishing detection system is a much more feasible approach than doing full-content analysis. In comparison, this system would exhibit faster responses, since full-content analysis is not required. RF and LSTM are able to evaluate URLs at a rate of 942 per second and 281 per second respectively. Nevertheless, a significant difference in the memory requirements of the models is noticeable. While RF takes 288.7 MB of memory, LSTM only employs 581 KB. This is crucial as there are memory-restricted applications, such as mobile apps. In this case. the RF model is unpractical and LSTM should be chosen.

In our analysis of the methodologies we found pros and cons for both methods. The LSTM model shows an overall higher prediction performance without the need of expert knowledge to create the features. The downside is that inner workings cannot be interpreted easily. Conversely, the RF model on average achieved a performance 5 percentage points lower than the LSTM model and needed expert knowledge to create the features. However, the RF model can be interpreted more easily due to input features and feature importance. In general terms, neural network models require far more training data, time and expertise to achieve satisfactory results than traditional models such as RF. We showed how an RF can

be trained in less than 3 minutes, while LSTM required 238 minutes. Additionally, we have to take into account parameters tuning in both models. In an RF model, we were required to tweak the number of trees and the depth, two static variables. In LSTM, we tweaked the network architecture, which includes number of epochs, size of inner layers, embedding size, and dropout parameters, among others.

REFERENCES

[1] APWG, "Phishing Activity Trends Report, 3rd Quarter 2016," Tech. Rep. December, 2016.

[2] S. Roopak and T. Thomas, "A Novel Phishing Page Detection Mechanism Using HTML Source Code Comparison and Cosine Similarity," in *2014 Fourth International Conference on Advances in Computing and Communications*, 2014, pp. 167–170.

[3] R. Dhamija, J. D. Tygar, and M. Hearst, "Why Phishing Works," in *SIGCHI Conference on Human Factors in Computing Systems*, 2006, pp. 581–590.

[4] J. Vargas, A. Correa Bahnsen, S. Villegas, and D. Ingevaldson, "Knowing your enemies: Leveraging data analysis to expose phishing patterns against a major US financial institution," in *2016 APWG Symposium on Electronic Crime Research (eCrime)*, 2016, pp. 52–61.

[5] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Learning to detect malicious urls," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 30:1–30:24, May 2011.

[6] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know Your Phish: Novel Techniques for Detecting Phishing Sites and Their Targets," in *International Conference on Distributed Computing Systems*, 2016, pp. 323–333.

[7] T. Thakur and R. Verma, *Catching Classical and Hijack-Based Phishing Attacks*. Cham: Springer International Publishing, 2014, pp. 318–337.

[8] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems ?" *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014.

[9] S. Marchal, R. State, and T. Engel, "PhishScore: Hacking Phishers Minds," in *CNSM*, 2014, pp. 46–54.

[10] Z. C. Lipton, "A Critical Review of Recurrent Neural Networks for Sequence Learning," *CoRR*, vol. abs/1506.0, pp. 1–38, 2015.

[11] T. Dietterich, "Machine learning for sequential data: A review," *Structural, syntactic, and statistical pattern recognition*, pp. 1–15, 2002.

[12] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting Domain Generation Algorithms with Long Short-Term Memory Networks," *http://arxiv.org/abs/1611.00791*, nov 2016.

[13] J. Zhang, P. Porras, and J. Ullrich, "Highly Predictive Blacklisting," in *17th USENIX Security Symposium*, 2008, pp. 107–122.

[14] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond Blacklists : Learning to Detect Malicious Web Sites from Suspicious URLs," *World Wide Web Internet And Web Information Systems*, pp. 1245–1253, 2009.

[15] C. Whittaker, B. Ryner, and M. Nazif, "Large-Scale Automatic Classification of Phishing Pages," *NDSS '10*, 2010.

[16] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit on - eCrime '07*, 2007, pp. 60–69.

[17] G. L'Huillier, A. Hevia, R. Weber, and S. Rios, "Latent semantic analysis and keyword extraction for phishing classification," in *International Conference on Intelligence and Security Informatics*, 2010, pp. 129–131.

[18] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.

[19] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know Your Phish: Novel Techniques for Detecting Phishing Sites and their Targets," oct 2015.

[20] R. Verma and K. Dyer, "On the Character of Phishing URLs: Accurate and Robust Statistical Learning Classifiers," in *ACM Conference on Data and Application Security and Privacy*, 2015, pp. 111–121.

[21] V. Ramanathan and H. Wechsler, "Phishing detection and impersonated entity discovery using Conditional Random Field and Latent Dirichlet Allocation," *Computers & Security*, vol. 34, pp. 123–139, 2013.

[22] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying Suspicious URLs : An Application of Large-Scale Online Learning," in *International Conference on Machine Learning*, Montreal, Canada, 2009, pp. 681–688.

[23] R. M. Mohammad, F. Thabatah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, vol. 25, no. 2, pp. 443–458, 2014.

[24] C. Ardi and J. Heidemann, "Poster: Lightweight content-based phishing detection," USC/Information Sciences Institute, Tech. Rep. ISI-TR-2015-698, May 2015.

[25] G. Wang, H. Liu, S. Becerra, K. Wang, S. Belongie, H. Shacham, and S. Savage, "Verilogo: Proactive phishing detection via logo recognition," UC San Diego, Tech. Rep. CS2011-0969, Aug. 2011.

[26] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL Names Say It All," in *INFOCOM, 2011 Proceedings IEEE*, 2011.

[27] T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform resource locators (url)," Tech. Rep., 1994.

[28] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[29] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM." *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[30] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1–32, 1997.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[32] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[33] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.