Intelligent Learning Architecture with Hybrid Features for Phishing Detection

Yu-Hung Chen
Department of Electrical Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan
D10907005@mail.ntust.edu.tw

Jiann-Liang Chen
Department of Electrical Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan
lchen1215@gmail.com

***Abstract*—** This study proposes a novel machine learning architecture that uses deep learning technology to extract features from the structure of a web page and construct a model for phishing detection. Hackers can commit crimes through a variety of Internet technologies. In recent years, phishing incidents have become more frequent, and the rapid development of information technology has enabled hackers to develop more advanced phishing attacks. Furthermore, the release of phishing toolkits, which are collections of software tools, make it easier for people with minimal technical skills to launch their own phishing attacks. Therefore, more attention must be paid to the prevention of such attacks. Protection from phishing websites has various aspects, including user training, public awareness, technical security measures and others. In this research, we further improve the phishing detection on phishing kits. This research proposes to use the combination HTML structural feature with the features proposed by AI@ntiPhish1.0 to train the phishing detection model. Relevant experimental results demonstrate that the combination of AI@ntiPhish1.0 features with extracted HTML structural features is more effective on detecting the phishing kits, increasing the accuracy thereof from 82% to 87.2%.

Keywords—Phishing attack, Machine Learning, Deep Learning, Cybersecurity, Data Imbalance

I. INTRODUCTION

With the rapid development of communication technology, an increase in the digital footprint of enterprises and individuals has enlarged the potential attack surface: all connecting devices may become attack targets. New cloud services and IoT applications to make everyday life more convenient are gradually emerging. They include those associated with online payments and online shopping, in relation to which data verification is required to ensure that a purchaser is a legitimate user. Personal Identifiable Information (PII), including account information, passwords, and other sensitive information, is sent over a network and authenticated with a database. User information must be transmitted over the Internet, allowing bad actors to commit financial crimes and cyber-attacks. Phishing is common away to obtain personal information and steal identity certification using email or social media [1].

Phishing is a crime that uses social engineering and information technology to steal personal information or financial account credentials. Most phishing websites attempt to imitate normal websites and are used for criminal activities. A phishing webpage looks like an official page of a bank, credit card company, or reputable public/private institution. Phishing attacks usually send message that contains malicious links that redirect recipients to fake websites, which request information such as account information, passwords, for example. Phishing attacks may even target a specific company or organization, leading to information leakage from the company or organization [2]. The attackers can carry out even more serious internal attacks using the leaked credential data.

A phishing kit is a collection of software tools that helps people with little or no technical skill to launch a phishing exploit. It typically includes website development software with a simple, low-code/no-code graphical user interface (GUI), and comes complete with email templates, graphics and sample scripts that can be used to generate convincing imitations of legitimate correspondence [3]. For an additional price, some kits may also include lists of e-mail addresses, telephone numbers and software for automating the malware distribution process.

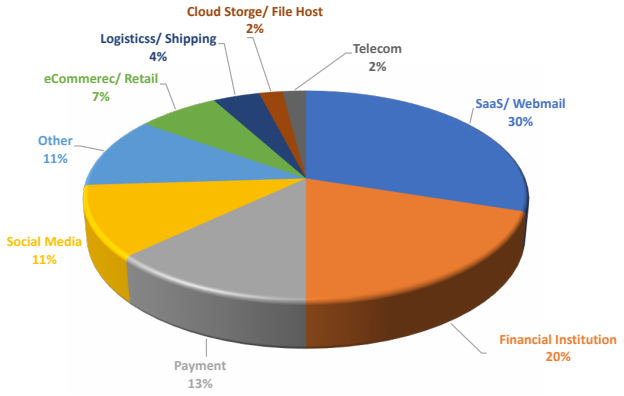

Fig. 1. Most-targeted Industries, 2020

According to an analysis by APWG in 2021, phishing attacks reached their peak in October 2020. In that month, 225,304 new phishing sites were launched, breaking the record. SaaS and Webmail providers were the most targeted industries of phishing in 2020, accounting for 30% of all attacks [4], as shown in Fig. 1.

Phishers sometimes deploy encryption technology to deceive users into believing that phishing sites are legal and safe. APWG contributor PhishLabs found that 84% of phishing sites used SSL encryption in the fourth quarter of 2020, as shown in Fig. 2. The frequency of encryption deployment on phishing sites exceeds that on ordinary websites, as only 66.8% of websites currently use SSL encryption.

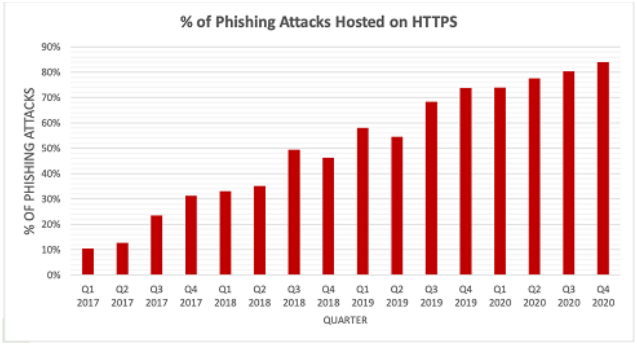

Fig. 2. Distribution of Phishing Attacks hosted on "Https"

In recent years, many studies have developed various phishing detection methods, which can be roughly divided into three categories: blacklist, visual similarity [5-6] and heuristic methods [7-10]. Heuristic methods include deep learning methods, which have emerged in recent years.

URL blacklisting is widely used to detect phishing attacks. The blacklisting method mostly involves comparing a URL to links that have been previously reported as phishing links. Blacklisting detects phishing sites based on a database of approved and unapproved URLs. Therefore, an up-to-date blacklist of phishing sites must be maintained.

Many studies have proposed detection that is based on image analysis. These methods involve taking screenshots of the website to be analyzed. Rendering the image of the page takes a long time and the image database must be constantly updated to ensure the accuracy of legal images.

Emerging artificial intelligence (AI) endows machines with intelligence and the ability to behave like humans. Many recent studies have used AI technology to train highly accurate and stable phishing detection models. AI-based algorithms learn the difference between phishing sites and legitimate sites without the disadvantages of blacklisting, and detect phishing more quickly than image processing or similarity analysis. This study does further research on more efficient detect and defend on phishing kit based on the previously developed AI@ntiPhish1.0 framework [10]. Since phishing kits have similar html structure, this study proposes to extract structural feature of phishing kits using a deep learning framework, and then combine this feature with features proposed by AI@ntiPhish1.0 to train a phishing detection model with high accuracy and stability.

II. Related Work

Phishing has been an ongoing problem since the emergence of the Internet. Hackers try to steal personal information and improperly use the obtained data. People must be very careful to avoid being tricked by phishing attacks, and require effective strategies to respond to such incidents if they are tricked. In recent years, many methods for detecting phishing have been proposed; they include such as blacklisting, image analysis, and others.

Blacklisting is a traditional phishing detection technique, which compares URLs with phishing links that have been reported by victims. This method relies on feedback from victims who have been deceived. A reported URL must be manually analyzed to determine whether it should be added it to the database. The disadvantage of this method is its failure to detect phishing attacks that have not been reported, so the blacklist database may be regarded as insufficient or even defective. Blacklist methods are already embedded in popular browsers, such as Google Chrome, Firefox, IE, and others.

Image-based phishing detection has been frequently used in recent years. This method involves capturing screenshots of web pages to be identified and extracts image features through certain feature extraction techniques. Traditional image processing methods include SIFT, SURF, HOG, etc. Deep learning methods that have been developed in recent years include CNN, LSTM, etc. After extracting the image using the above feature extraction method, the similarity is calculated with all the images in the image database through the distance formula. Ultimately, the tested URL is compared with the URL of the most similar legitimate website.

Haruta et al. proposed a novel visual similarity-based phishing detection scheme that uses hue information with an auto-updating database [5]. They combined whitelisting with an image signature to identify legitimate websites and phishing websites. Under this mechanism, the author prepared a lot of phishing website images, calculated the color ratios of these images, and filtered each image for analysis against a whitelist. After the preprocessing, the author compared the similarity of the suspicious images with the existing image database. If the image database contained similar web pages, then the web page was identified as a phishing web page. Since convolutional neural networks (CNN) have been very effective in image classification tasks overthe past ten years, they are often used in image classification tasks. However, training a deep learning model requires a large dataset and adjusting and optimizing the weight of the model to make the model more stable and effective takes a long time. Accordingly, Phoka et al. proposed the use of transfer learning technique to enhance the effectiveness and accuracy of image processing for phishing detection [6]. They used a CNN model, trained on the ILSVRC-2012-CLS dataset, as a pre-training model, to which they added enhanced phishing images for retraining. Finally, the output model is used for phishing detection and verification.

Heuristic analysis is to extract the features of legitimate websites and phishing websites and use artificial intelligence algorithms to quantify, train and learn the extracted features, which enhance the phishing recognition ability of the detection model.

Ghimire et al. used various algorithms, including Decision Tree (DT), Support Vector Machine (SVM) to train and evaluate different models [7]. To solve the problem of data

imbalance, the model combines under-sampling and over-sampling techniques. Experiment results revealed that the over-sampling method can bring effective help to the allogeneic of unbalanced data. Kumar et al. [8] proposed a URL-based lightweight machine learning model for phishing detection. Their method extracts 24 features from the URL and is trained using various algorithms, including Random Forest (RF), Fully Connected Neural Network (FNN), and others. The accuracy of the detection model that was thus trained on the Kaggle dataset reached 95.07%, while the accuracies of that trained on the UNB dataset for multi-class and binary-class classification were 97.75% and 99.72%, respectively. Since most people are directed to phishing websites from emails, some people apply Natural Language Processing (NLP) to detect phishing emails. Gualberto et al. [9] proposed a multi-level structure method for analyzing the content of emails using NLP. Their method extracts text from the body of an email and then extracts the corresponding text vector using Term Frequency–Inverse Document Frequency (TF-IDF). The extracted vectors are then preprocessed using two different feature extraction methods. The first applies Mutual Information and Chi-square processing to the features. The second extracts feature by using PCA and LSA. Finally, the extracted features are input to different AI algorithms for training; these include K-Nearest Neighbor (KNN), Extreme Gradient Boosting (XGBoost), and others. Based on the results of their analysis, they proposed that the LSA of Method 2 combined with the XGBoost algorithm can yield an F1-score of 100%, as can the use of Chi-square with Random Forest. AI@ntiPhish1.0 [10] proposed a novel machine learning architecture and various learning algorithms to build anti-phishing services to avoid phishing attacks. In the AI@ntiPhish1.0 framework, extracted feature data are used mostly in the first stage of, which involves feature evaluation and analysis by applying statistical and mathematical models. In the second stage, the extracted features are trained and evaluated through traditional machine learning algorithms, and their effectiveness and accuracy are analyzed. The AI@ntiPhish1.0 framework also evaluates the ensemble learning concept by combining multiple classifiers, such as Adaboost, bagging, and voting. The author claimed that among their considered algorithms, the XGBoost model performed best. In addition, AI@ntiPhish1.0 raises the problem of data imbalance in the training process and generates data by applying the over-sampling SMOTE algorithm in response to this problem. It then trains model using these generated data. Experimental results reveal that the proposed learning architecture with the SMOTE method improved the model coverage and performed best with respect to accuracy, precision, and recall. In recent years, the generation of representations through learning and the application of representation vectors to classification tasks or clustering tasks have aroused great interest and seen considerable progress [11-12]. Alon et al. [11] proposed a deep learning method for learning the feature vector of code, and evaluated the extensibility and generalizability of this vector. The architecture of the method is called code2vec. In an experiment, Jave's code was converted into an Abstract Syntax Tree (AST) and its syntax path was used as a representation. However, the declaration of parameters in the code frequently exhibits high variability, and training directly makes the matrix sparser. To solve this problem, Alon et al. first normalized the variables before training and then learn relevant feature vectors through deep learning from these representations. According to the relevant experimental results, in addition to effectively representing the similarity between functions, the resulting feature vector effectively represents the encrypted functions without being affected by the encryption.

Although heuristic analysis can increase the ability of recognizing phishing websites, the accuracy of the models will be influenced by the amount of data and the characteristics of the features of those sites. When the number of phishing sites collected is larger, the range that the model can learn is wider, which can have better recognition ability. In contrast, if the training data are insufficient, learning enough to determine whether it is a phishing site is difficult, resulting in a higher error rate. When a feature disappears from a phishing site or a phishing page is embedded into a certain path in a legitimate website, the accuracy of the detection model is affected. In recent years, the emergence of phishing kits has helped people with almost no technical skills to launch phishing attacks. Similar html templates can be set up on websites with different characteristics. The structure of html changes rapidly. In order to cache up the rapid change of html structure, it needs more generalized method to represent the html with sufficient feature efficiently. Embedding method transform the input data into a fixed dimension while keeping the most important feature. The feature extracted by the embedding method can also make the model more generalized. To represents the structure of HTML and extract more features from it, a learning framework that can represent its structure effectively is proposed. The extracted structural feature can be combined with AI@ntiPhish1.0 features effectively to improve the detection of, and defense against, such phishing kits. Comparing with model without the embedding feature, model trained with embedding feature has better results.

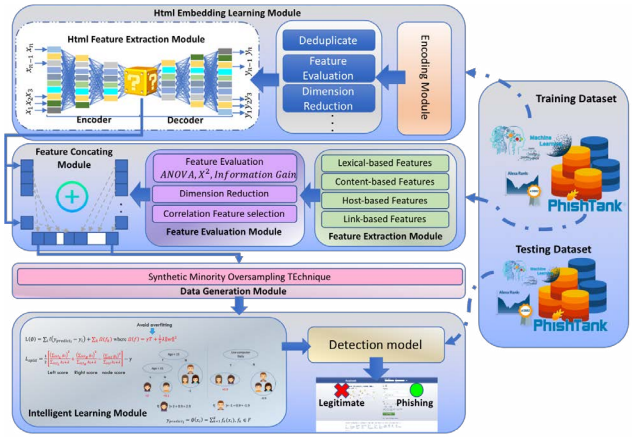

Fig. 3. Learning Architecture

III. Proposed Architecture

This research concerns the intelligent learning architecture of AI@ntiPhish1.0 and combines the feature vector with those

generated by html embedding learning architecture for phishing detection. Fig. 3 displays the architecture.

A. Data Collection

In this study, many phishing and legal websites were obtained from various sources, such as PhishTank, Alexa, VirusTotal, and others. Then, the collected data were sent to the HTML Embedding Learning Module and the Feature Extraction Module. The Feature Extraction Module uses the 28 features that were proposed in AI@ntiPhish1.0, as shown in Table I. The HTML Embedding Learning Module uses an autoencoder-decoder framework to learn features that can effectively represent the HTML structure, and requires a large amount of the HTML structures for training. A total of 10,000,000 HTML structural data were used to train the HTML embedding model. The training data that are used to develop the phishing detection model are associated with 25,000 phishing sites and 100,000 normal sites. These training data were collected from 2020/12/01 to 2020/12/31. With respect to the testing data, 1000 data of phishing and normal sites were collected respectively from 2021/01/01~2021/01/10.

B. Feature Extraction

The two sources of the training features were the html embedding learning module and the feature extraction module, as shown in Fig. 1. After collection, the training data were sent to the html embedding learning module and the feature extraction module that were used by AI@ntiPhish1.0. The features that were used by AI@ntiPhish1.0 included a lexical-based feature, a content-based feature, a host-based feature, and a link-based feature. The features that were most relevant were extracted using a mathematical model and statistical methods in the feature evaluation module. According to the analysis results of AI@ntiPhish1.0, the most effective feature information for training the phishing detection model were finally extracted. Table I presents the corresponding features. To increase the accuracy of detection of phishing kits by the model, the html embedding learning framework, which uses deep learning to learn the structure of html and extracts the feature vector that can be used to interpret the html structure through the final well-trained model, is proposed. Finally, the concatenation of html embedding features and the feature that was extracted by the AI@ntiPhish1.0 architecture is used to train the phishing detection model.

Since the html structure can interpret the hierarchical relationship between html tags via using the DOM (Document Object Model), the html was converted into a DOM in the html embedding learning module. The same hierarchical relationship of the html tags was captured and concatenated into a sequence, as shown in Fig. 4. During training of the embedding learning module, the sequence was extracted from html in order, and then preprocessed by deduplication, feature evaluation, dimensional reduction, and other procedures. Since the html tags were hierarchically related to each other, the autoencoder-decoder architecture in the html feature extraction module used the sequence-to-sequence architecture with a Long-Short Term Memory Network (LSTM) for training. LSTM is an extension of the recurrent neural network, which can maintain information in memory for longer time period. The extracted html sequence has order-dependency between html tags in the sequence, so we choose to use the LSTM architecture.

TABLE I. PHISHING FEATURES

Symbol	*Feature*	*Symbol*	*Feature*
F1	is_http_connection	F15	null_a_tag
F2	is_ip_address	F16	script_block_rate
F3	dots	F17	style_block_rate
F4	is_special_words	F18	get_title_feature
F5	url_linkin_num	F19	is_login_form
F6	url_traffic_rank	F20	is_with_whois
F7	get_kbytes	F21	get_time
F8	is_frame	F22	is_redirect
F9	is_meta_redirect	F23	ipv4_numbers
F10	is_meta_base64_redirect	F24	ipv6_numbers
F11	same_extern_domain_script_rate	F25	organization
F12	same_external_domain_link_rate	F26	is_alias
F13	same_external_domain_img_rate	F27	is_weird_serial
F14	external_a_tag_same_domain	F28	get_day_age

C. Data Generation

The feature concatenating module aggregates the features that are extracted by html embedding extraction module and feature extraction module, and then sends the aggregated vector to the data generation module for data generation. In practice, the amount of data on legitimate websites is much larger than that on phishing websites. Therefore, to solve the problem of data imbalance, the Synthetic Minority Over-sampling TEchnique (SMOTE) is used herein to generate more data.

The theory of SMOTE is to find K nearest neighbor data points of the same category and draw lines among them, and then to generate similar data from the data points on the lines. The relevant formula is shown in (1).

$$X_{new} = X_i + (X' - X_i) \bullet \delta \quad (1)$$

Xi is a data point that is randomly select from the minority class dataset; *X'* is the data point that is closest to *Xi*; δ is between 0 and 1.

D. Learning Algorithm

After data generation, all data and features are propagated to the intelligent learning module for learning. The intelligent learning module trains and optimizes the phishing detection model, based on the input data. The results of the analysis based on AI@ntiPhish1.0 reveal that the XGBoost model has substantially greater accuracy and stability in phishing detection than other algorithms. Additionally, the data that are generated using the SMOTE method effectively increase the accuracy of the detection model. In this research, all data

(including generated data) and features are trained and optimized through XGBoost, and the benefits of the html embedding learning module, proposed in this research, in the detection of phishing websites is evaluated.

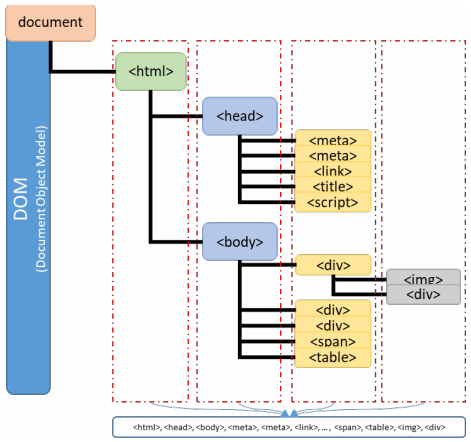

Fig. 4. Extracted tag sequence from DOM

E. Evaluation Standard

The following performance evaluation mechanism is used to evaluate the performance of the proposed model. The performance indicators are accuracy, precision, recall, and f1-score. The calculation formula is shown in Table II.

TABLE II. EVALUATION INDICATORS

Symbol	***Feature***
Accuracy	(TP+TN)/(TP+FN+FP+TN)
Precision	TP/(FP+TP)
Recall	TP/(FN+TP)
F1 score	(2*Precision*Recall)/(Precision+Recall)

IV. EXPERIMENTAL RESULTS

A large amount of data from PhishTank, Alexa, and VirusTotal, were collected. The training and parameter selection of the detection model were performed using 100,000 legitimate websites and 25,000 phishing websites. The training data were collected from 2020/12/01 to 2020/12/31. Since the training of the html embedding learning module is a form of unsupervised learning, 1,000,000 html structures were randomly selected. For the testing data of the model, we select 1000 phishing websites and 1000 legitimate websites from 2021/01/01 to 2021/01/10 in order to avoid duplicate data between training and testing, and also simulate testing circumstances in real-world. Table III and Table IV present detailed hardware configuration and parameter settings for training the html embedding.

Ten-fold cross-validation is used to evaluate the effect and benefits of the XGBoost algorithm with different numbers of estimators. The number of estimators that provides the best performance and the highest stability are selected. After that, we evaluate the performance of the model trained with the optimization parameters on the testing dataset. The experiment evaluates the model in the following four scenarios; XGBoost trained with 28 features, XGBoost trained with 28 features and html embedding features, XGBoost trained with 28 features and combined with the SMOTE method, and XGBoost trained with 28 features and html embedding features and combined the SMOTE method.

TABLE III. HARDWARE CONFIGURATION

Symbol	***Feature***
CPU	Intel(R)Xeon(R)CPUE5-2620 v4
Memory / HDD / SSD	32GB/2TB/256GB
Linux	Ubuntu 16.04
GPU	3*MSI GTX 1080Ti 11G

TABLE IV. PARAMETER OF HTML EMBEDDING LEARNING FRAMEWORK

Symbol	***Feature***
Max sequence length	500
Batch size	16
Epoch	1000
LSTM cell	64
Learning rate	0.0001
Dropout	0.6

This experimental design, involving four scenarios, not only reveals the impact of 28 features on the training of phishing detection models that were developed in recent years, it also provides insight into on the extent to which the html embedding features and SMOTE algorithm can improve the detection model. During the analysis, this study also evaluated the number of estimators. The number of estimators was adjusted from 5 to 50 in steps of five estimators and the accuracy of the model in each case was calculated. The orange line and the blue line in Fig. 5 represent the effects of adding and not adding html embedding features, respectively. According to the experimental results, for any number of estimators, the accuracy of the model with additional html embedding features significantly exceeds that without html embedding features. The gray line and yellow line in Fig. 5 indicate that the use of the SMOTE algorithm for data generation is effective models that include and exclude html embedding features. The results of this analysis indicate that using the SMOTE algorithm can enhance model training and detection, as revealed by the blue and gray lines or the orange and yellow lines in Fig. 5. When the html embedding features are added and the SMOTE algorithm is used to generate data, the accuracy of the detection model can be improved again. The XGBoost model performs best in all of the scenarios when the number of estimators is 45. Fig. 6 displays a detailed performance evaluation of the XGBoost model with 45 estimators.

The extent to which the addition of an html embedding feature can improve the model training without use of the SMOTE algorithm is evaluated. The results show that when XGBoost is trained with html embedding features, the accuracy is increased from 74.80% to 79.40%, and Recall and Precision are also greatly improved. When the model training is conducted using the SMOTE algorithm for data generation, it can bring more help to the diversity of the model. According to the experimental results, using the SMOTE algorithm increases the accuracy by 10%. The method that was proposed by AI@ntiPhish1.0, which does not include the html embedding feature has an accuracy of 82%, which exceeds that of both

XGBoost with html embedding and that of XGBoost without html embedding. Finally, when html embedding features and the SMOTE algorithm for data generation are added, an accuracy of 87.2% is obtained; this is nearly 5% higher than the accuracy of the model that was trained by the AI@ntiPhish1.0 architecture.

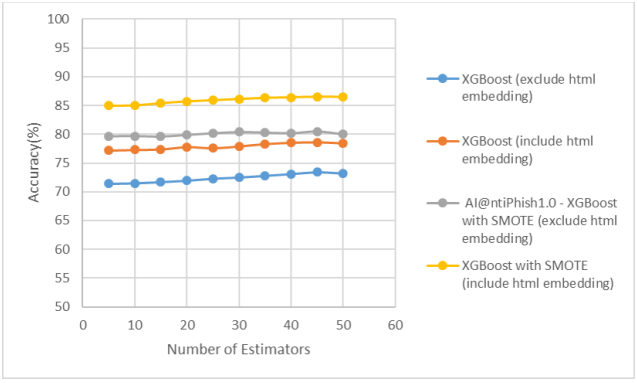

Fig. 5. Accuracy of XGBoost with imbalanced learning, and html embedding

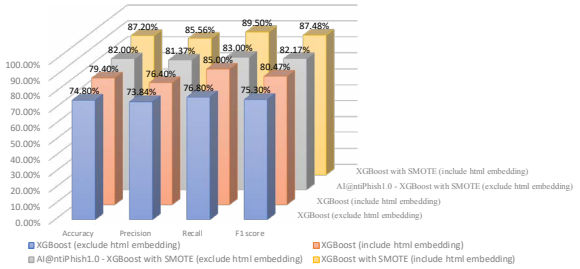

Fig. 6. Accuracy of XGBoost for testing data with imbalanced learning, and html embedding (Estimator=45)

V. CONCLUSIONS

This paper proposed a framework for phishing detection; it involves a feature extraction module that consists of two parts; one uses the 28 features that were proposed by AI@ntiPhish1.0, and the other is the html embedding feature. The html embedding feature is extracted from the sequence-to-sequence model that was composed of LSTM in the html embedding learning module that was proposed by this research. Finally, we train and evaluate the html embedding features generated by the html embedding learning module and the 28 features used by AI@ntiPhish1.0. The relevant experimental results show that when the html embedding feature is added to the trained detection model, its accuracy in detecting phishing websites is significantly improved. In the future, we can use different algorithms to perform the embedding. The embedding does not have to be performed using deep learning. Perhaps the same enhancement of accuracy can be achieved using traditional machine learning methods. In addition, we can also try to add the attributes of the html structure to the learning of embedding to enrich the html representation. In recent years, research into graph neural networks have accelerated. The html embedding that is conducted in this research can be combined with files or email applications to form a correlation graph. Our future research will focus on how to use GNN or GCN analytical methods to learn graphs or extend to detect attacks on various applications.

ACKNOWLEDGMENT

Authors thank for financial supports of the Ministry of Science and Technology, Taiwan under contract MOST 108-2221-E-011-068-MY2.

REFERENCES

[1] A. Odeh, I. Keshta and E. Abdelfattah, "Machine LearningTechniquesfor Detection of Website Phishing: A Review for Promises and Challenges," Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0813-0818, 2021.

[2] R. Zaimi, M. Hafidi and M. Lamia, "Survey paper: Taxonomy of website anti-phishing solutions," Proceedings of the 2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS), pp. 1-8, 2020.

[3] A. Oest, Y. Safei, A. Doupé, G. Ahn, B. Wardman and G. Warner, "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis," Proceedings of the 2018 APWG Symposium on Electronic Crime Research (eCrime), pp. 1-12, 2018.

[4] APWG, PHISHING ACTIVITY TRENDS REPORTS, https://apwg.org/trendsreports/

[5] S. Haruta, F. Yamazaki, H. Asahina and I. Sasase, "A Novel Visual Similarity-based Phishing Detection Scheme using Hue Information with Auto Updating Database," Proceedings of the 2019 25th Asia-Pacific Conference on Communications (APCC), pp. 280-285, 2019.

[6] T. Phoka and P. Suthaphan, "Image Based Phishing Detection Using Transfer Learning," Proceedings of the 2019 11th International Conference on Knowledge and Smart Technology (KST), pp. 232-237, 2019.

[7] A. Ghimire, A. Kumar Jha, S. Thapa, S. Mishra and A. Mani Jha, "Machine Learning Approach Based on Hybrid Features for Detection of Phishing URLs," Proceedings of the 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 954-959, 2021.

[8] Y. Kumar and B. Subba, "A lightweight machine learning based security framework for detecting phishing attacks," Proceedings of the 2021 International Conference on COMmunication Systems & NETworkS (COMSNETS), pp. 184-188, 2021

[9] E. S. Gualberto, R. T. De Sousa, T. P. De Brito Vieira, J. P. C. L. Da Costa and C. G. Duque, "The Answer is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering," Journal of IEEE Access, Vol. 8, pp. 223529-223547, 2020.

[10] Y. H. Chen and J. L. Chen, "AI@ntiPhish — Machine Learning Mechanisms for Cyber-Phishing Attack," Journal of the 2019 IEICE Transactions on Information and Systems, Vol. E102-D, No. 5, pp. 878-887, 2019.

[11] U. Alon, M. Zilberstein, O. Levy and Eran Yahav, "Code2vec: learning distributed representations of code," Proceedings of ACM Program, pp. 1-29, 2019.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is All You Need," Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000-6010, 2017.