Phishing URL Detection using Information-rich Domain and Path Features

Eint Sandi Aung[†a)]   Hayato YAMANA[†b)]

†Department of Computer Science and Communication Engineering, Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo, 169-8555, Japan.

E-mail : a) eintsandiaung@toki.waseda.jp, b) yamana@info.waseda.ac.jp

Abstract Malicious or phishing detection has been drawing a serious concern since the early 21st century because of tremendous electronic transfer accesses such as financial transitions and identity theft via online services. Amongst various detection schemes, including content-based approach, uniform resource locator (URL)-based detection is widely used not only for its comparable performance w.r.t accuracy but also for its adaptability to any other forms (for example, embedding URLs in spam messages or emails). In phishing URL detection, feature engineering is a crucial yet challenging way to improve performance. Manually-generated features are risky and highly dependent on datasets. Thus, recently, researchers tend to focus on information-based features, which extracts features based on the URL's texts. To put it simply, researchers adapt the neural network to extract characters/words which are rich in indicating valuable information in the URLs. Our research focuses on information-based features applying a neural network-based model in which we consider both domain-based and path-based features. Then, we analyze and compare our results with previous papers and summarize opinions for a better detection system.

Keyword Malicious URL, Phishing, Security, Domain-level, Path-level

1. Introduction

Cyber phishing has been a global concern for decades because of resulting loss in hundreds of millions of dollars in 2020, according to Keepnet's latest phishing statistics[1]. Phishing attacks not only target at stealing personal information such as financial details but also intrude into organizations via installing malicious programs such as ransomware. Keepnetlabs reported 540 data breaches in the USA in the first half of 2020. According to Verizon's 2020 data breaches investigation report, 22 percent of data breaches include phishing attacks. Keepnet mentioned that 1 in 8 employees share information to phishing websites, and over 60,000 phishing websites are reported in 2020 March alone.

Meanwhile, APWG's 2020 statistics[2] reported that the number of phishing attacks has increased since March. It said that most phishing attacks are activated by a small number of registrars, domain registries, and host providers. Surprisingly, 80 percent of phishing sites have SSL encryption enabled to deceive victims. In the Q3 report of APWG, 40 percent of all SSL certificates that phishers used were issued by "Let's Encrypt." Moreover, recently phishers in Brazil avoid domain names that can attract attention, which is different from their previous behaviors of drawing victims' focus by mimicking or compelling catchwords. Since APWG has been measuring more precisely to analyze how phishers are constructing phishing URLs, it reveals a significant improvement of unique phishing sites since March 2020. In the Q3 report, the number of unique phishing websites detected (approx. 572000) is over 1.5 times that of unique phishing emails (approx. 367300).

The statistics mentioned above showed that phishing URLs are now receiving growing attention recently. However, analyzing URLs has not been an easy research area because most URLs have randomly generated information yet challenging research. Thus, our research focuses on detecting phishing URLs from which we try to gather as much information as we can to dig in for information-rich features.

We organize this paper into six different sections: Section 2 describes the background of phishing w.r.t different detection schemes. In Section 3, we analyze the importance of features in the detection area and discuss the nature of features in detail. It is followed by Section 4 of our approach in which we explain different embedding techniques applied in this research. Then, Section 5

[1] Keepnet: https://www.keepnetlabs.com

[2] APWG: https://apwg.org

describes detailed step-by-step implementations from scratch. We describe different evaluation results and opinions illustrating comparative performance in Section 6. Eventually, Section 7 concludes our paper with its future work.

2. Related Work

Phishing attacks steal users' personal information via online services. Those attacks come in more sophisticated forms; however, to put the detection schemes simply, we can organize malicious or phishing detection into two types, content-oriented and URL-oriented. The content-oriented approach analyzes the content of a webpage [13,40], such as text, visual similarities, CSS, and javascript styles, while URL-oriented schemes mainly focus on the structure or string patterns of URLs and its features[33,35], such as blacklist[36,38] or white-list URLs[37,39]. There are also hybrid approaches[**11,31,32**] that comprise features from both of them. Although both schemes are complicated in their ways, URL-oriented detection has an advantage of comparable performance to content while maintaining less security risks, the concern in content-based detection (Figure 1.1). Furthermore, a URL-based scheme brings an early detection for newly generated phishing websites.

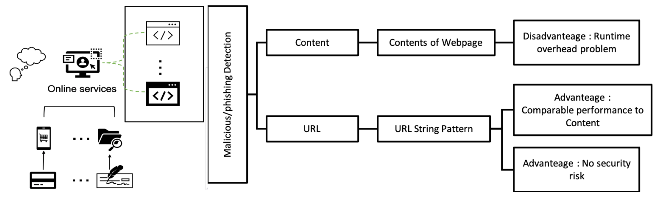

Online services
Malicious/phishing Detection
Content
Contents of Webpage
Disadvanteage : Runtime overhead problem
URL
URL String Pattern
Advanteage : Comparable performance to Content
Advanteage : No security risk

Figure 1.1. Phishing Attacks and Detection Schemes

Moreover, we can further categorize detection techniques into machine learning-based (ML) and neural network-based (NN) detection. We introduce a brief description of ML and NN-based detection techniques in Sections 2.1 and 2.2.

2.1. Machine Learning based Detections

Researchers and data analysts have been applying machine learning for a few decades because of its comparable performance in terms of accuracy and precision in data analysis. In addition, ML-based algorithms have more interpretability because of their simplicity to trace how the data shaped and worked inside. In this section, we brief some of the recent research for a better explanation.

B.Sabir [1] in 2020 addressed the security vulnerabilities of ML-based phishing URL detection(MLPU) systems and proposed an evasion attack framework to the systems. They reproduced 41 MLPU systems and simulated an attack on them. They emphasized their work for a better future detection system.

D. Sahoo [3,30] published comprehensive survey papers on malicious URL detection using machine learning in 2019 and 2017. The papers reviewed noble contributions and addressed various perspectives in terms of feature representation and algorithm designs.

O. Sahingoz [6] in 2019 proposed a real-time anti-phishing system using NLP features that are independent of third-party services. This work had the advantage of language independence with real-time execution.

C.Wu [7] in 2019 proposed a detection system focused on URLs, and in this work, they considered URLs in the content of a webpage combined with those in the source code of the webpage. They applied Levenshtein distance to calculate URL strings' similarity between the main URL and its contained URLs.

K.Chiew [10] in 2019 contributed a new feature selection framework, named as the Hybrid Ensemble Feature Selection (HEFS). They utilized a cumulative distribution function gradient algorithm to produce primary feature subsets, later being fed to output secondary feature subsets that are then used to select baseline features by using perturbation ensemble.

Similar ML-based detection works can be found in [8][9][13][14][15][16][17][18][19][33].

2.2. Neural Network based Detections

In the current days, the neural network has acquired a spotlight in data science as its deep learning to achieve accurate evaluation outcomes. Although most of its application area still exists in image/computer-vision related works, research on text analysis such as summarization and classification has been growing over time. In this section, we describe some of the previous researchers' work.

F.Tajaddodianfar [2] proposed a novel deep learning architecture called Texception. In this work, they considered both character-level and word-level information from the incoming URL. It has the advantage of not relying on manually generated features and multiple parallel convolutional layers to expand deeper and wider.

Y.Haung [4] in October 2019 proposed a capsule-based neural network for phishing URL detection. They implemented two capsule layers: primary capsule layer and classification capsule layer. In the primary layer, they extracted accurate features from shallow features generated by the former convolution layer. Classification capsule layer used a dynamic routing algorithm and squashing function and averaged outputs from all branches.

Y.Haung [5] in August 2019 proposed a deep learning based phishing URL detection called PhishingNet. In this paper, CNN for character-level feature extraction and attention-based hierarchical RNN for word-level feature extraction is performed separately and then fused them into CNN. They improved generalization ability on newly emerged URLs.

Similar works can be found in [20][21][22][23][24][25][26][27][28].

3. Importance of Features

Handling raw data has never been an easy task in a URL-based scheme. To better outperforming accuracy, a system needs a large set of features. Moreover, defining features needs experts' knowledge. Extracting features from URLs takes an enormous amount of time.

ML-based detection mainly consists of various sophisticated features ranging from text-focused (e.g., similarities or relations between words) to URL characteristics-focused (e.g., lexical features such as IP address, redirection, Bag of Words features, distance-based features, and third-party features) features[1]. Meanwhile, NN-based detection analyzes either features used in ML or raw character- and/or word-level features[2], and it needs to transform raw data into integer-coded character- or word-vector representation. Transforming raw data to meaningful information, called feature vector, is challenging because of the randomness of characters/words in URL.

3.1. Manually Generated Features

Manually generated features are fixed features where phishers can bypass by a small change of URL structures. These features are the ones we implement in our previous work [11]. To define an effective feature to the system, it needs not only knowledge of featuring engineering experts but also enough durability to make sure phishers cannot easily deceive. Furthermore, its major weakness is that the performance of the detection highly relies on them. One feature could easily drop the detection rate as a noisy feature.

3.2. Information-rich Features

We call the features extracted from raw URLs directly Information-rich features, such as words/characters, which we transform integer-encoded vector representation. Simply, we extract words or characters from the URL text and consider them as features themselves, as shown in Figure 3.1. Such features are neither risky to detection rate like manually generated features. We define such features as information-rich features as they contain useful information (e.g., alphanumeric characters and meaningful words)

https	fs	yama	info	waseda	ac	jp	users	Sign_in

Figure 3.1. Example of URL words after tokenization

4. Our Approach

We perform our system shown in Figure 4.1. As we aim to overcome the bottleneck of manually generated features, we target information-rich features by extracting meaningful words from them.

In the model, we perform two different embeddings layers with different tokenization techniques as shown in Section 4.2. To our best knowledge, this is our new way of adapting two embedding techniques in phishing URL detection. We apply LSTM after dropout layers from each embedding and then concatenate the outputs into one vector and performed batch-normalization followed by dense layer.

4.1. Features

In our preprocessing stage, we split URLs into two parts: domain and path, because we assume that URL patterns in the domain are less random than those in the path. The domain part consists of the URL components until the end of the domain name, whereas the path part includes the rest of the URLs until the non-alphanumeric character "?." We encode them into character or word level and then feed them into the model.

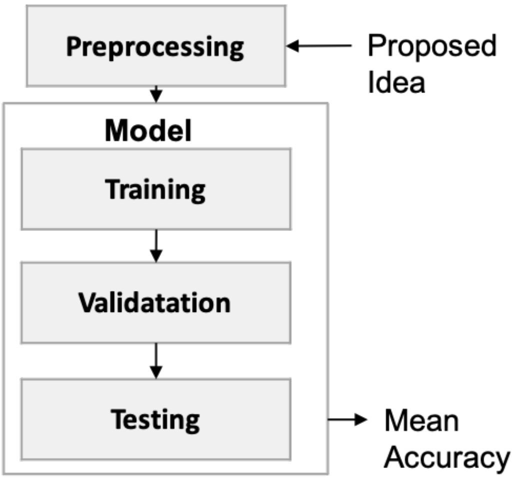


Figure 4.1. Our System

4.1.1. Domain based Features

In the domain part, we transform URLs into a set of words by tokenizing because meaningful words/brand names can mostly be found in the domain part, as shown in Figure 4.2.

Here, the extracted/tokenized features are the word features mixed with non-alphanumeric characters.

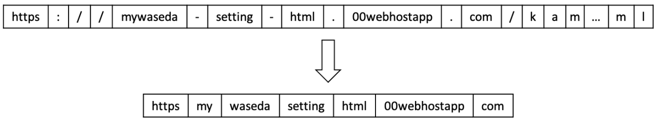


Figure 4.2. Domain based Features

4.1.2. Path based Features

In the path part, the characters appear more randomly and less meaningful than those in the domain part shown in Figure 4.3. Thus, we adopt character level features in the path part and split them into characters.

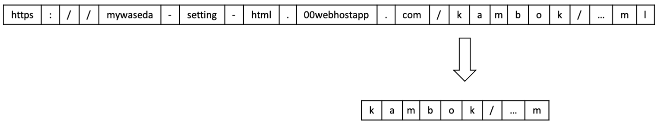


Figure 4.3. Path based Features

4.2. Embedding Techniques

We apply two embedding techniques, such as Keras embedding and ELMo embedding, to train two LSTM models in our work. Then, we prepare our input vectors for the models.

4.2.1. Keras Embedding

Keras provides an embedding layer to be used on text data in the neural network. As it needs integer-encoded input data, we encode words whose frequencies are larger than 1 with unique IDs and later save them in a vocabulary dictionary in ascending order of their frequencies. We reserve <UNKNOWN> as for those with frequency is 1. This assumption is because we might encounter unknown words in the validation and testing phase.

The embedding layer is initialized with random weights to learn all the words in model training. We specify the embedding layer as the network's first hidden layer with three arguments: input_dim=length of vocabulary dictionary, output_dim=32, and input_length=200. As for the input vector, we tokenize domain-level words and path-level characters as shown in 5.1.1.

4.2.2. ELMo Embedding

ELMo is a deep contextualized word representation model. It uses a deep bi-directional LSTM to create word representation instead of a vocabulary dictionary. ELMo analyzes words in their context and is character-based embedding, which forms word representation even if there are out-of-vocabulary words. Instead of a dictionary look-up, ELMo creates representation vectors while passing texts through the model.

We load a fully trained model tensorflow hub for our embedding. We apply ELMo embedding with output_dim of 1024. In our input sequence of texts, we consider domain level words and preprocess them with word segmentation described in 5.1.2.

4.3. Dataset

We mainly adopted a legitimate dataset from DMOZ (later known as Curl) and PhishTank collected in our previous work [11]. We prepared our dataset for two cases: imbalanced and balanced training. The imbalanced dataset has the ratio of phishing to legitimate as approx. 1:6. We split our dataset into 80 percent of training and 20 percent of testing as shown in Figure 4.4.

Table 4.1. Dataset details

Dataset	Size		Type
	Legitimate	Phishing	
D1	15,000	15,000	Balanced
D2	99,383	15,056	Imbalanced

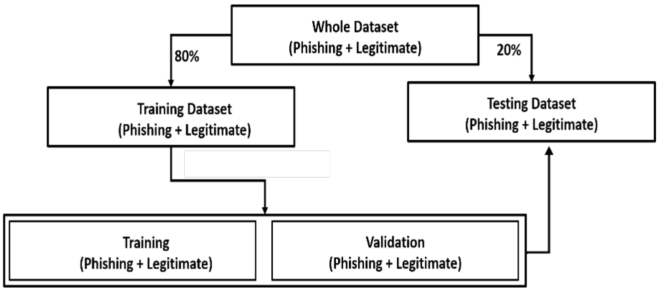


Figure 4.4. Dataset Partition

For the balanced dataset, as we do not want to choose 15,000 legitimate and 15,000 phishing manually from the whole dataset, we apply dataframe.sample() at the beginning of the preprocessing and then perform splitting in the same way as in the imbalanced dataset.

5. Implementation

We implement our work using tensorflow 2 on the machine with GPU GeForce GTX 1060.

We initially performed preprocessing of raw URLs and constructed a vocabulary dictionary for Keras Embedding. We adapted two tokenization techniques as shown in Section 5.1.1. and 5.1.2.

5.1. Preprocessing

We need to transform raw data into integer-encoded vectors before adapting to the model. Thus, we performed the preprocessing stage. Firstly, we discard the rest of the URL parts after “?” because the query values (key, value pairs) are randomly generated and could bring noise. Then, we tokenize them depending on parts and encode domain-level words or path-level characters. Finally, we feed them into two embedding layers separately as shown in Figure 5.1.

5.1.1. Tokenization in Keras Embedding

We adopt the Keras embedding in which we input a vector of concatenated words (of the domain) and character sequences (of the path). In this embedding, we tokenize domain level words in the same way as previous research [1]. However, we consider non-alphanumeric characters as mentioned in the previous section.

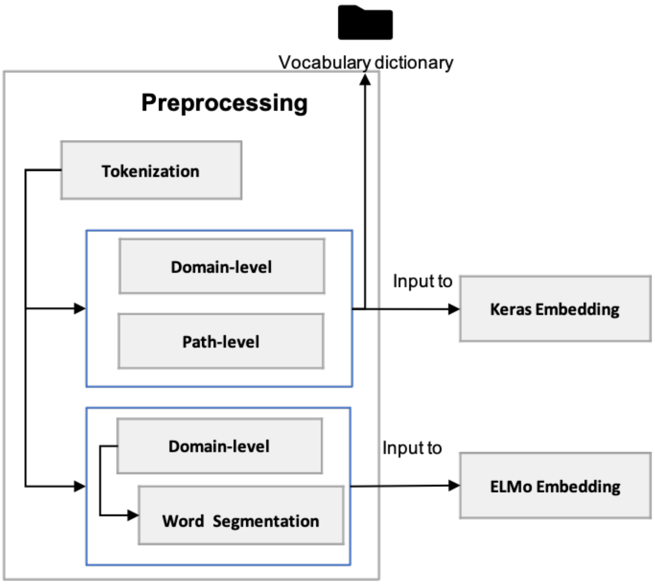
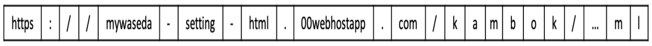


Figure 5.1. Preprocessing Stage

We transform the input URL, for example, *https://www.mywaseda-setting-html.00webhostapp.com/ kambok/...ml,* into a vector as shown in Figure 5.2.

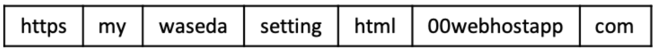


https	:	/	/	mywaseda	-	setting	-	html	.	00webhostapp	.	com	/	k	a	m	b	o	k	/	...	m	l

Figure 5.2. Tokenizing in Keras Embedding

5.1.2. Tokenization in ELMo Embedding

In ELMo embedding, we utilize the “wordsegment” library, which is based on Google Trillion Word Corpus and frequently used in the natural language processing (NLP) area, to extract/tokenize meaningful words from URLs. For example, a URL *https://mywaseda-setting-html.00webhostapp.com* is split into the sequence of words shown in Figure 5.3.



https	my	waseda	setting	html	00webhostapp	com

Figure 5.3. Tokenizing in ELMo Embedding

5.2. Model

We design our model with two input layers as shown in Figure 5.4: (1) input to Keras embedding and (2) input to ELMo embedding. We input a concatenated input vector (domain-level words and path-level character) into Keras embedding layer while the domain-level words alone to ELMo embedding layer. The embedding layer is then followed by the dropout layer with a value of 0.2. We adapt LSTMs with a recurrent dropout of 0.2 and later dense them before the concatenating layer. We then perform batch-normalization and again dense into 128 output size, followed by output layer.

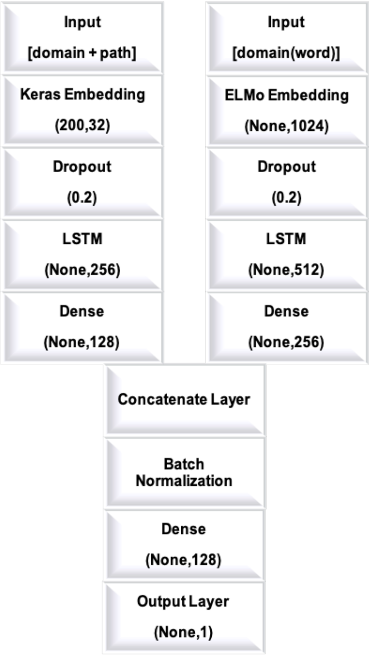


Figure 5.4. Model Architecture

6. Experimental Evaluation

This section describes the model setup, parameters, and analysis of experimental results in details as follows.

6.1. Parameter Setting

We used StratifiedKFold from the sklearn library for training. Our parameter setting is described in Table 6.1.

Table 6.1. Model Parameter

Parameter	Value
Training: Testing Split	8:2
Epoch Size	10,20
Batch Size	128
Learning Rate	0.001
Loss	Binary_crossentropy
Activation	Relu, sigmoid
Keras Embedding Dimension	(200,32)
ELMo Embgedding Dimension	(None,1024)

6.2. Evaluation Result

Due to the page limitation, we show the results with epoch=10. We train our model on both balanced (D1) and imbalanced (D2) datasets.

Figure 6.1 shows the evaluation performance of the balanced dataset (D1).

We measured four evaluation metrics (accuracy, precision, recall, and loss) in which we achieve 93 percent and a loss of 27 phases in the testing phase.

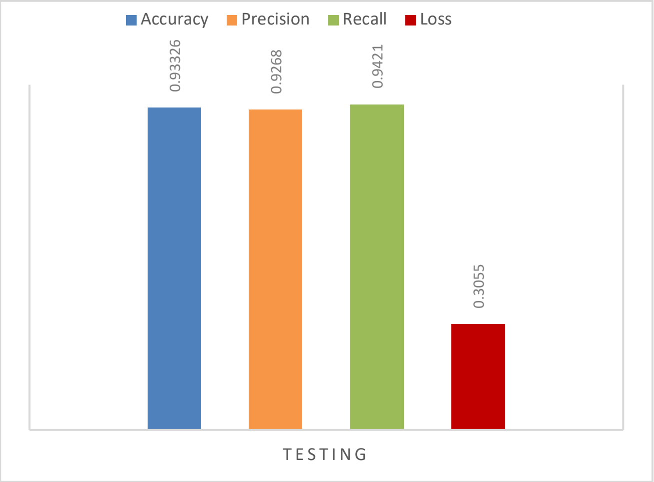


Figure 6.1. Evaluation Result for D1

We then measured the distribution of evaluation metrics on D1 dataset using boxplot as shown in Figure 6.2 to analyze how the metric scores diverge from average scores.

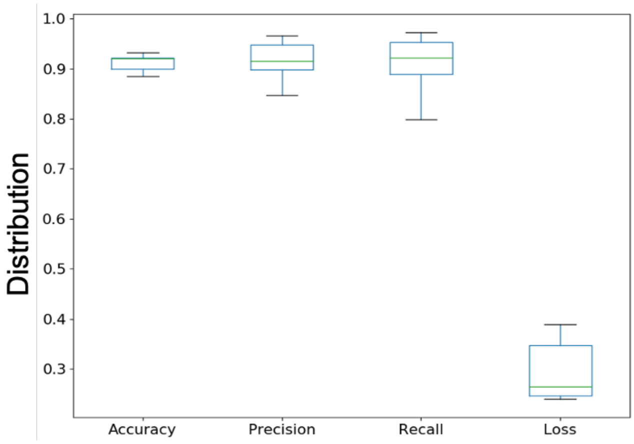


Figure 6.2. Distribution of Evaluation Metrics of the balanced dataset (D1)

When we adapt our approach in the imbalanced dataset (D2), we measure mean accuracy, precision, recall, and loss as similar to D1, and we achieve approx. 97 percent of accuracy with a loss of 11 percent in the testing phase as shown in Figure 6.3. Compared to the evaluation result of D1, the more the data size is growing, the more we optimize the loss because of the growing vocabulary dictionary size. We again measured the distribution on evaluation scores and illustrated as boxplot in Figure 6.4 and analyze that the larger the dataset size, the lesser the deviation.

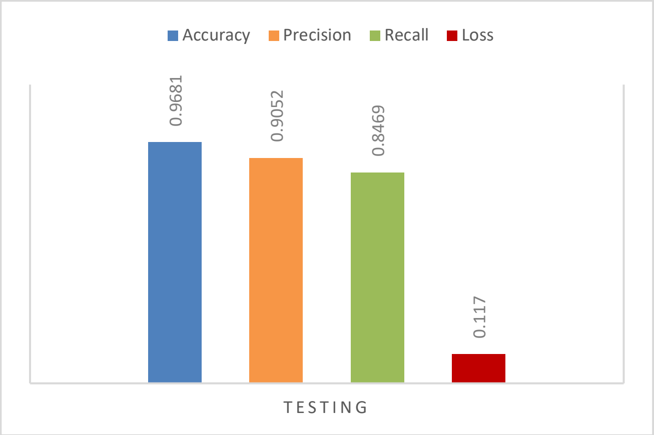


Figure 6.3. Evaluation Result for D2

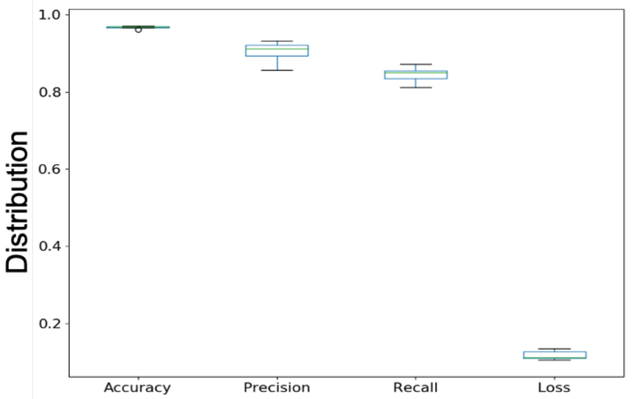


Figure 6.4. Distribution of Evaluation Metrics of the imbalanced dataset (D2)

6.3. Comparison with Baseline

We compared our accuracy with the baseline model, a previous research work (URLNet[27]). We choose delimit-mode that delimits URL by special characters and treat each special characters as words, which is similar to our work.

We also choose embedding-mode as character and word CNN in their work. Then, we adapted our dataset, and eventually, we measure accuracy and compare it with our result as shown in Figure 6.5. We outperform approx. 6% and 1% of accuracy in D1 and D2, respectively.

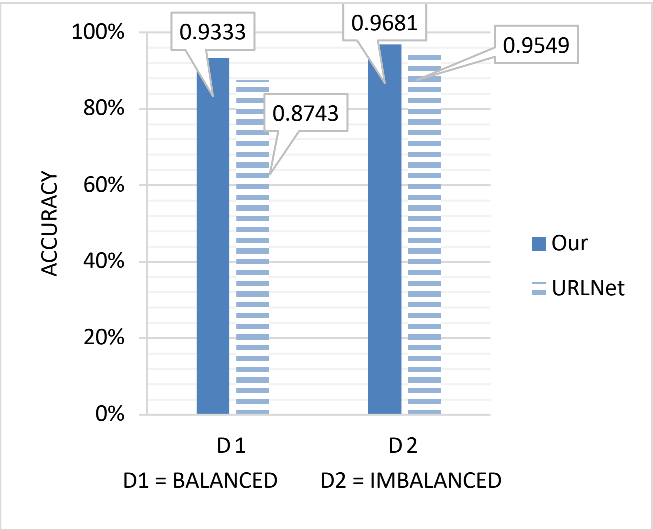


Figure 6.5. Comparison of Accuracy

7. Conclusion

In conclusion, we approached phishing URL detection with two types of segmentation (traditional segmentation for Keras embedding and meaningful NLP-based word segmentation for ELMo embedding).

We applied our proposed two-level features such as domain-level word and path-level character features to adapt in Keras embedding.

We then designed concatenated layers from the outputs of two embedding layers. We ran our program 10 fold cross-validation and averaged mean evaluation metrics, which we achieved approx. 93 percent and 97 percent in D1 and D2 datasets, respectively.

We finally compared our evaluation in terms of accuracy with URLNet by selecting parameters similar to our work and improved approx. 5% in D1 and 1% in D2 using our dataset.

As for the future work, we need to minimize loss since learning fewer vocabulary in the dataset has been the cause of high loss.

Acknowledgement

A part of this research was funded by Grants-in-Aid for Scientific Research(**KAKENHI**) Program (17KT0085).

References

[1] B. Sabir, M. Babar, and R. Gaire, "An evasion attack against ML-based phishing URL detection," in ArXiv, vol.abs/2005.08454v1, 2020.
DOI:10.1145/1122445.1122456.

[2] F. Tajaddodianfar, J. W. Stokes, and A. Gururajan, "Texception: a character/word-level deep learning model for phishing URL detection," Proc. ICASSP 2020, 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.2857-2861, 2020.
DOI:10.1109/ICASSP40776.2020.9053670.

[3] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning : a survey," In ArXiv, vol.abs/1701.07179v3, 2019.

[4] Y. Huang, J. Qin, and W. Wen, "Phishing URL detection via capsule-based neural network," Proc. 13th International Conference on Anti-counterfeiting, Security, and Identification, pp.22-26, 2019.
DOI:10.1109/ICASID.2019.8925000.

[5] Y. Huang, Q. Yang, J. Qin, and W. Wen, "Phishing URL detection via CNN and attention-based hierarchical RNN," Proc. 18th International Conference on Trust, Security and Privacy in Computing and Communications and 13th International Conference on Big Data Science and Engineering, pp.112-119, 2019.
DOI:10.1109/TrustCom/BigDataSE.2019.0024

[6] O. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," in Journal of Expert Systems with Applications, vol.117, pp.345-357, 2019. DOI:10.1016/j.eswa.2018.09.029

[7] C. Wu, C. Kuo, and C. Yang, "A phishing detection system based on machine learning," Proc. International Conference on Intelligent Computing and its Emerging Applications (ICEA 2019), pp.28-32, 2019.

[8] R. Patgiri, H. Katari, R. Kumar, and D. Sharma, "Empirical study on malicious URL detection using machine learning," Proc. 15th International Conference on Distributed Computing and Internet Technology, pp.380-388, 2019. DOI:10.1007/978-3-030-05366-6-31

[9] M. T. Suleman, and S. M. Awan, "Optimization of URL-based phishing websites detection through genetic algorithms," in Journal of Automatic Control and Computer Sciences, vol.53, no.4, pp.333-341, 2019.
DOI:10.3103/s0146411619040102

[10] K. Chiew, C. Tan, K. Wong, and K. Yong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," in Journal of Information Sciences, vol.484, pp.153-166, 2019.
DOI:10.1016/j.ins.2019.01.064

[11] E. Aung, and H. Yamana, "URL-based phishing detection using the entropy of non-alphanumeric characters," Proc. 21st International Conference on Information Integration and Web-Based Applications & Services, pp.385-392, 2019. DOI: 10.1145/3366030.3366064

[12] A. Oest, Y. Safei, A. Doupe, G. J. Ahn, B. Wardman, G. Warner, "Inside a phisher's mind: understanding the anti-phishing eco system through phishing kit analysis," Proc. 2018 APWG Symposium on Electronic Crime Research (eCrime), 2018.

[13] H. Shirazi, B. Bezawada, and I. Ray, "Know thy doma1n name: unbiased phishing detection using domain name based features," Proc. the 23rd ACM on Symposium on Access Control models and Technologies, pp.69-75, 2018. DOI:10.1145/3205977.3205992

[14] C. Liu, L. Wang, B. Lang, and Y. Zhou, "Finding effective classifier for malicious URL detection" Proc. 2nd International Conference on Management Engineering, Software Engineering and Service Sciences, pp.240-244, 2018. DOI:10.1145/3180374.3181352

[15] W. Daffa, O. Bamasag, and A. AlMansour, "A survey on spam URLs detection in Twitter," Proc. 1st International Conference on Computer Applications and Information Security, pp.1-6, 2018. DOI:10.1109/CAIS.2018.8441975

[16] H. Yuan, X. Chen, Y. Li, Z. Yang, and W. Liu, "Detecting phishing websites and targets based on URLs and webpage links," Proc. 24th International Conference on Pattern Recognition, pp.3669-3674, 2018.

DOI:10.1109/ICPR.2018.8546262

[17] D. Patil, and J. Patil, "Malicious URL detection using decision tree classifiers and majority voting technique," in Journal of Cybernetics and Information Technologies, vol.18, no.1, pp.11-29, 2018. DOI:10.2478/cait-2018-0002

[18] S. Parekh, D. Parikh, S. Kotak, and S. Sankhe, "A new method for detection of phishing websites: URL detection," Proc. 2nd International Conference on Inventive Communication and Computational Technologies, pp.949-952, 2018. DOI:10.1109/ICICCT.2018.8473085

[19] A. Jain, and B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," in Journal of Ambient Intelligence and Humanized Computing, vol.10, pp.2015-2028, 2018.

DOI:10.1007/s12652-018-0798-z

[20] I. Arnaldo, A. Arun, and S. Kyathanahalli, "Acquire, adapt and anticipate: continuous learning to block malicious domains," Proc. International Conference on Big Data, 2018. DOI:10.1109/BigData.2018.8622197

[21] M. Trivesan, and I. Drago, "Robust URL classification with generative adversarial networks," in Journal of ACM SIGMETRICS Performance Evaluation Review, vol.46, no.3, pp. 143-146, 2018. DOI:10.1145/3308897.3308959

[22] A. Anand, K. Gorde, J. R. A. Moniz, N. Park, T. Chakraboty, and B. Chu, "Phishing URL detection with oversampling based on text generative adversarial networks," Proc. International Conference on Big Data, pp.1168-1177, 2018.

DOI:10.1109/BigData.2018.8622547

[23] S. Shivangi, P. Debnath, K. Sajeevan, and D. Annapurna, "Chrome extension for malicious URLs detection in social media applications using artificial neural networks and long short term memory networks," Proc. 18th International Conferences on Advances in Computing, Communications and Informatics, pp.1993-1997, 2018.

DOI:10.1109/ICACCI.2018.8554647

[24] A. Vazhayil, R. Vinayakumar, and K. P. Soman, "Comparative study of the detection of malicious URLs using shallow and deep networks," Proc. 9th International Conference on Computing, Communication and Networking Technologies, pp.1-6, July, 2018. DOI:10.1109/ICCCNT.2018.8494159

[25] A. C. Bahnsen, I. Torroledo, D. Camacho, and S. Villegas, "DeepPhish: simulating malicious AI," in APWG Symposium on Electronic Crime Research, 2018.

[26] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," in Journal of Neural Processing Letters, vol.48, pp.1347-1357, 2018. DOI:10.1007/s11063-017-9666-7

[27] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "URLNet: learning a URL representation with deep learning for malicious URL detection," in ArXiv, vol.abs/1802.03162, 2017.

[28] A. C. Bahnsen, and E. C. Bohorquez, "Classifying phishing URLs using recurrent neural networks," in APWG Symposium on Electronic Crime Research, pp.1-8, 2017. DOI:10.1109/ECRIME.2017.7945048

[29] D. Sahoo, C. Liu, and S. C. H. Hoi, "Malicious URL detection using machine learning : a survey," in ArXiv, vol.abs/1701.07179v1, 2017.

[30] A. Hodzic, J. Kevric, "Comparison of machine learning techniques in phishing website classification," Proc. International Conference on Economic and Social Studies (ICESoS'16), vol.3, pp.249-256, 2016.

[31] M. Dadkhah, S. Shamshirband, A. Wahab, "A hybrid approach for phishing web site detection," in the Electronic Library, vol.34, no.6, pp.927-944, 2016.

[32] M. N. Feroz, S. Mengel, "Phishing URL detection using URL ranking," Proc. 2015 IEEE International Congress on Big Data, pp.635-638, 2015.

[33] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: detecting phishing with streaming analytics," in IEEE Transactions on Network and Service Management, vol.11, no.4, pp.458-471, 2014.

[34] E. Sorio, A. Bartoli, E. Medvet, "Detection of hidden fraudulent URLs within trusted sites using lexical features," Proc. 2013 International Conference on Availability, Reliability and Security, 2013.

[35] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," Proc. IEEE INFOCOM, 2010, pp.1-5, 2010.

[36] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," Proc. the 4th ACM Workshop on Digital Identity Management, pp.51–60, 2008.

[37] M. Sharifi and S. H. Siadati, "A phishing sites blacklist generator," Proc. IEEE/ACS International Conference on Computer Systems and Applications, pp. 840-843, 2008.

[38] J. Kang and D. Lee, "Advanced white-list approach for preventing access to phishing sites," Proc. International Conference on Convergence Information Technology (ICCIT 2007), pp.491-496, 2007.

[39] L. Wenyin, G. Huang, L. Xiao Yue, Z. Min, X. Deng, "Detection of phishing webpages based on visual similarity," in Special interest tracks and posters of the 14th International Conference on World Wide Web, pp. 1060-1061, 2005.