Journal of Information Processing Vol.32 973–989 (Nov. 2024)

[DOI: 10.2197/ipsjjip.32.973]

Regular Paper

PhiSN: Phishing URL Detection Using Segmentation and NLP Features

Eint Sandi Aung[1,a)] Hayato Yamana[1,b)]

Received: March 6, 2024, Accepted: June 30, 2024

Abstract: Phishing is a cyberattack method employed by malicious actors who impersonate authorized personnel to illicitly obtain confidential information. Phishing URLs constitute a form of URL-based intrusion designed to entice users into disclosing sensitive data. This study focuses on URL-based phishing detection, which involves identifying phishing webpages by analyzing their URLs. We primarily address two unresolved issues of previous research: (i) insufficient word tokenization of URLs, which leads to the inclusion of meaningless and unknown words, thereby diminishing the accuracy of phishing detection, (ii) dataset-dependent lexical features, such as phish-hinted words extracted exclusively from a given dataset, which restricts the robustness of the detection method. To solve the first issue, we propose a new segmentation-based word-level tokenization algorithm called *SegURLizer*. The second issue is addressed by dataset-independent natural language processing (NLP) features, incorporating various features extracted from domain-part and path-part of URLs. Then, we employ the segmentation-based features with SegURLizer on two neural network models - long short-term memory (LSTM) and bidirectional long short-term memory (BiLSTM). We then train the 36 selected NLP-based features on deep neural networks (DNNs). Afterward, we combine the outputs of the two models to develop a hybrid DNN model, named "*PhiSN*," representing phishing URL detection through segmentation and NLP features. Our experiments, conducted on five open datasets, confirmed the higher performance of our PhiSN model compared to state-of-the-art methods. Specifically, our model achieved higher F1-measure and accuracy, ranging from 95.30% to 99.75% F1-measure and from 95.41% to 99.76% accuracy, on each dataset.

Keywords: phishing URL detection, word segmentation, segmentation feature, lexical NLP feature

1. Introduction

Phishing constitutes a criminal activity carried out by individuals known as phishers or attackers. They aim to deceive unsuspecting users into divulging confidential information, ranging from sensitive financial statements to commonly targeted data, like personal identification and passwords. Over recent years, there has been a significant increase in phishing attacks reported in the quarterly publications of the Anti-Phishing Working Group (APWG)[*1], as shown in **Fig. 1**. From the second quarter of 2020, the number of phishing websites has accelerated. The first-quarter report of 2023 alleged that the number of unique phishing websites (approximately 1.62 million) was 2.5 times larger than the initial peak of 2020 (approximately 637,000).

Phishing detection techniques are classified into five approaches: whitelist [1], [2], blacklist [3], [4], content [5], [6], [7], [8], [9], visual similarity [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], and uniform resource locator (URL)-based approaches [22], [23], [24], [25], [26].

The whitelist approach allows for two methods: direct matching and similarity checking. Direct matching involves comparing the pre-stored legitimate IP addresses (or URLs) with the incoming IP addresses (or URLs) [1]. The similarity-checking scheme uses a pattern-matching algorithm (e.g., Ratcliff/Obershelp similarity [27]) to compare pre-stored white URLs with the input URL [2]. However, whitelist-based approaches have limitations in maintaining the whitelists up-to-date because of updating delays.

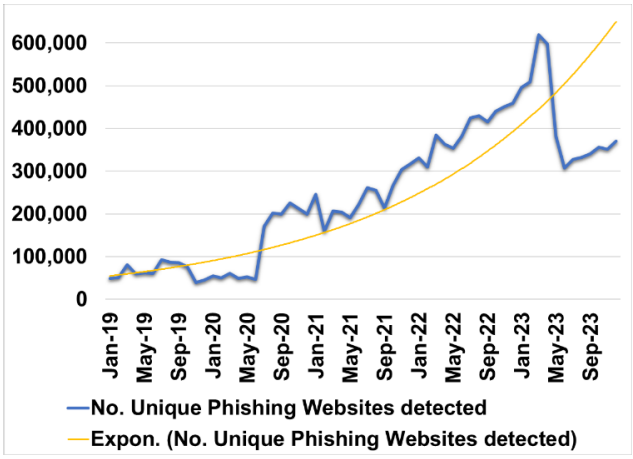


Fig. 1 Number of Phishing Websites (created based on APWG Report 2019–2023)[*1].

Similarly, the blacklist approach verifies whether the input URL is present in pre-stored blacklists. Both whitelist- and blacklist-based approaches are susceptible to failure in zero-hour attacks because of delays in updating their lists.

[1] Waseda University, Shinjuku, Tokyo 169–8555, Japan
[a)] eintsandiaung@toki.waseda.jp
[b)] yamana@waseda.jp

[*1] APWG analyzes phishing attacks reported by its member companies. Retrieved from https://apwg.org/

© 2024 Information Processing Society of Japan

The content-based approach necessitates retrieving the webpage to analyze its content, which comes with the potential risk of exposure to malware. Furthermore, the visual similarity-based approach experiences a notable drop in performance when comparing the image-based similarity of webpages [18].

Meanwhile, the URL-based approach detects phishing webpages by retrieving features only from the URL, without retrieving the webpages. URL-based approaches utilize lexical features [28], character embedding [22], word embedding [23], or word and character embedding [24] features.

Among the above-discussed approaches, this study focuses on the URL-based approach due to the following reasons:

- Ensuring that malware attacks are avoided without retrieving the webpage pointed by the URL.
- Shortening the phishing detection latency without retrieving the webpage or content/visual-similarity calculation.

The URL-based approaches adopt two kinds of features extracted from URLs: (1) segmentation-based and (2) NLP-based features. Segmentation-based features are extracted from the raw text of URLs, encompassing characters and words [22], [23], [24]. NLP-based features, also called lexical features, include the length of the URL, number of special characters, number of words, and text similarity. Hereafter, we represent lexical features as NLP features.

Previous works have two unresolved issues in URL-based phishing detection: (i) insufficient word tokenization of URLs, which leads to the inclusion of meaningless and unknown words, thereby diminishing the accuracy of phishing detection, and (ii) dataset-dependent NLP features that restrict the adaptability of the detection method to real-world scenarios.

First, regarding segmentation-based features, phishers tend to entice a bait, that is, counterfeit URLs with straightforward techniques, such as by (1) swapping letters, replacing letters with similar digits, adding random letters and/or digits, (e.g., www.amaze0nesecure.me), (2) concatenating words to legitimate keywords/brands (e.g., amazonwebservices.com), and (3) embedding legitimate keywords/brands in different parts of a URL (e.g., seller.amazon.de.analytics.com). Thus, detecting character swapping in the URL (e.g., tokenizing amaze0nesecure.me to [amaze, 0, ne, secure, me]), precisely tokenizing words (e.g., amazonwebservices.com to [amazon, web, services, com]), and extracting brand names embedded in the different positions (e.g., seller.amazondeal.de.analytics.com to [seller, amazon, deal, de, analytics, com]) are of utmost importance. Such trends indicate that highly accurate word segmentation of the URL has the potential to improve phishing detection accuracy.

However, an existing dilemma in URL segmentation is insufficient tokenization, which fails to accurately segment URLs into meaningful words, resulting in unknown words. In contrast to the tokenization in NLP, the traditional tokenization technique, delimiting a URL by special characters [24], cannot be accurate because the URL consists of a combination of characters/words without any whitespaces. Although the state-of-the-art work [23] proposed byte pair encoding (BPE) for tackling unknown words, the BPE cannot effectively retrieve meaningful words (e.g., bankofamerica to [bank, ##of, ##ame, ##rica], instead of [bank, of, america]). Based on the aforementioned considerations, the following factor must be considered to reduce unknown words in URL segmentation:

- Phishing detection systems aim to increase their precision in detecting phishing webpages but minimize false positives, where legitimate webpages are misidentified as phishing. Unknown words in URLs can introduce ambiguity and increase false positives because the systems may interpret unknown words as suspicious. Thus, reducing unknown words helps lower the rate of false positives.

Therefore, we set our goal to reduce unknown words in URL segmentation. In this regard, we propose a new method called *SegURLizer* to accurately tokenize words while identifying character swapping in URLs.

Second, with respect to NLP features, previously employed dataset-dependent lexical features [29], [30] have shown vulnerabilities to circumvention by phishers. For phishing URLs, we must never assume that phishers consistently employ the same techniques to bypass phishing detection for their malicious URLs. They continuously develop sophisticated evasion methods. Thus, it is imperative not to rely on dataset-dependent features, which are derived exclusively from specific URLs within each dataset. Examples of data-dependent features include counting the number of phish-hinted words [30] and calculating the term frequency-inverse document frequency (TF-IDF) of phish-hinted words [29] in the dataset. To enhance phishing detection accuracy, we opted to utilize dataset-independent 36 NLP features, which are less reliant on the characteristics of the dataset and can be solely extracted from individual URLs.

Our contributions comprise the following:

1. Addressing the issue of unknown words by our proposed *SegURLizer* [31], [32], which combines the use of BERT Tokenizer [*2] and WordSegment tokenizer [*3],
2. Detecting character swapping in URLs to resolve the problem of replacing letters with similar digits, resulting in the decomposition of a URL into a sequence of consecutive letters, digits, and symbols,
3. Extending our previous work [31] by introducing 36 dataset-independent NLP features exclusively extracted from URLs and evaluating their performance using comprehensive datasets.

Note that this paper is an extended version of our previous papers [31], [32]. This paper extends our previously proposed NLP features to select and add data-independent NLP features, then introduces various models incorporating BiLSTM, LSTM, and DNN. We conducted a comprehensive evaluation of phishing detection performance with state-of-the-art methods on five well-established datasets to clarify the effectiveness of our proposed phishing URL detection model, *PhiSN*, which was not shown in our previous papers. The datasets include Ebbu [22], [28], [33], PhishStorm [34], ISCXURL [35], [36], CatchPhishD1 [29], and CatchPhishD2 [29].

[*2] BERT Tokenizer – Tokenizer with BERT model. Retrieved from https://pypi.org/project/bert-tokenizer/

[*3] Word Segment – Apache2 licensed module for English word segmentation. Retrieved from https://pypi.org/project/wordsegment/

© 2024 Information Processing Society of Japan

Table 1 Comparison of Related Work.

Authors (year)	Description	Shortcomings	Datasets: number of URLs (+) : phishing, (-) : legitimate	Metrics: Performance
Marchal et al. [34] (2014)	PhishStorm: Proposing intra-URL relatedness features between the low-level domain (registered domain) and the remaining part of the URL.	Requiring third-party services (search engine).	PhishTank: (+) 53,089 DMOZ: (-) 48,009	Accuracy: 94.91% F-measure: 94.72%
Le et al. [24] (2018)	URLNet: Proposing three levels of embeddings to capture the semantic meaning of words and sequential patterns of characters, i.e., character-, word-, and character-level word embeddings, to solve the infrequent word problems.	Exponentially increased unknown words with training data. High computational complexity because of three levels of embeddings. High memory usage.	VirusTotal: (+) 949,725; (-) 14,050,275	(1M train, 10M test) TPR: 97.22% (5M train, 10M test) TPR: 98.58% *(FPR: 0.1%)
Sahingoz et al. [28] (2019)	Proposing a real-time anti-phishing system implementing word-based and NLP-based features, later fused into hybrid features.	Poor F-measures on hybrid features compared to word-based features.	Ebbu: (+)37,175; (-) 36,400	(NLP Features) Accuracy: 97.98% F-measure: 98.00%
Rao et al. [29] (2020)	CatchPhish: Proposing term frequency-inverse document frequency (TF-IDF) feature of phish-hinted words.	Requiring third-party services such as GeoLocation-based features. Dataset-dependent feature of phish-hinted words. Testing on many domain-only URLs with contradicted ground-truth label inclusion.	D1-PhishTank: (+)40,668; D1-Alexa: (-)43,189 D2-PhishTank: (+)40,668; D2-CommonCrawl: (-)42,220 D3-PhishTank: (+)40,668; D3-Alexa and CommonCrawl: (-)85,409	(Dataset D1) Accuracy: 95.67% F-measure: 95.81% (Dataset D2) Accuracy 93.95% F-measure: 94.16% (Dataset D3) Accuracy 94.26% F-measure: 95.88 %
Gupta et al. [35] (2021)	Proposing effective nine lexical features.	Cannot handle domain-only URLs efficiently.	University of Canada Brunswick: (+)9964; (-)10,000	Accuracy: 99.57% F-measure: 99.58%
Maneriker et al. [23] (2021)	URLTran: Proposing the URLTran using byte pair encoding (BPE) tokenizers to handle character subsequences.	Many unknown-word extractions (e.g., [bankofamerica] to [bank, ##of, ##ame, ##ri, ##ca]). Evaluation with a private dataset.	Microsoft Edge and IE- Training:(+)77,870; (-)961,543 Validation: (+)19,468; (-)240,386 Testing: (+)8,742; (-)1,775,413	Accuracy: 99.67% F-measure: 99.71% (@0.01FPR)TPR: 86.80%
Aung et al. [32] (2021)	Proposing the URL-Tokenizer algorithm combining character- and word-level segmentation in BiLSTM to handle the unknown-word problems.	Missing lexical features.	D1-PhishTank: (+)10,000; D1-IP2Location(-)10,000 D2-PhishTank: (+)5,000; D2-IP2Location(-)50,000	(Imbalanced) Accuracy: 97.7% (Balanced) Accuracy: 95.7%
Aung et al. [31] (2022)	Extending our work [32] with 24 NLP features to construct a hybrid model (DNN-CNN) fusing word-level segmentation-based features in CNN and NLP-based features in DNN.	Evaluation with one dataset (Ebbu [22],[28]).	Ebbu: (+)37,175; (-) 36,400	(word-CNN) Accuracy: 98.55% F-measure: 98.57% (word-DNN-CNN) Accuracy: 98.64% F-measure: 98.65%
Alani et al. [42] (2022)	PhishNot: Proposed a cloud-based phishing URL detector along with domain registration-related features and network-related features such as TTL.	Facing the risk of false-negative detection if a reliable domain also provides phishing webpages because of insufficient NLP-features.	PhishTank: (+) 30,646 Alexa: (-) 58,000	Accuracy: 97.48% FPR: 3.22% FNR: 2.18%
Ozcan et al. [22] (2023)	Proposing hybrid deep learning models, DNN-LSTM and DNN-BiLSTM, using character embedding and manually extracted NLP features.	Over-whelming unnecessary symbols in character embedding (e.g., "%2B" or "%21") without URL decoding.	D1-Ebbu: (+)37,175; (-) 36,400 D2-PhishTank: (+) 13,000 D2-PhishStorm: (+) 13,000	Accuracy: 98.79% F-measure: 98.81%
Jalil et al. [33] (2023)	Proposing a machine learning framework implementing full URL-, host- and path-based features, along with feature selection technique.	Low F-measure and precision on four benchmark datasets (*). Dataset-dependent feature of suspicious words, similar to Rao et al. [29].	D1-Kaggle: (+114,203); (-)392,801 D2-Kaggle: (+)55,914; (-)39,996 *D3-CatchPhishD3: (+)40,668; (-)85,409 *D4-CatchPhishD2: (+)40,668; (-)42,220 *D5-CatchPhishD1: (+)40,668; (-)43,189 *D6-Ebbu: (+)37,175; (-) 36,400	(D1-Kaggle) Accuracy: 96.25% F-measure: 96.20% (D2-Kaggle) Accuracy: 94.65% F-measure: 98.60% (*D3-CatchPhishD3) Accuracy: 92.20% F-measure: 87.60% (*D4-CatchPhishD2) Accuracy: 91.63% F-measure: 91.60% (*D5-CatchPhishD1) Accuracy: 94.80% F-measure: 94.70% (*Ebbu) Accuracy: 96.85% F-measure: 96.90%
Wang et al. [38] (2023)	PhishBERT: proposing a bidirectional encoder representation-based pre-trained model on a large-scale unlabeled dataset.	Low F-measure although pre-training on large scale data. Meaningless words from byte-pair encoding (BPE).	CommonCrawl: 2.85 billion Open PageRank: 338.71 million Expired Domains: 538.2 million PhishTank: 2.2 million	Accuracy (2% training data): 92.00% F-measure (2% training data): 92.00% (@0.01FPR)TPR (60% training data): 89.31% (@0.01FPR)TPR (40% training data): 70.06%
Jishnu et al. [40] (2023)	Proposing a fine-tuned BERT model, leveraging segmentation-based features with additional URL features.	Unknown-word problem because of traditional tokenization by the special characters.	PhishTank & Kaggle: (+) 100,000 Majestic Million & Kaggle: (-) 100,000	Accuracy: 97.32% F-measure: 97.32%

The rest of the paper is organized as follows. Section 2 provides related work, followed by explaining the fundamental concepts and techniques employed in our proposed algorithm in Section 3. Section 4 proposes our new phishing URL detection model *PhiSN*, which utilizes segmentation and NLP features. Section 5 evaluates our proposed *PhiSN* with five datasets, followed by discussions in Section 6. Finally, our work is concluded in Section 7.

2. Related Work

Researchers have explored various features for detecting phishing URLs, encompassing NLP features [28], [35]. However, phishers continually refine their attacks to evade conventional handcrafted NLP features. In response, researchers have adapted advanced features extracted from the raw text of URLs, which we call segmentation-based features encompassing characters and words [22], [24], leveraging the advancements in text classification and NLP for improved detection. In this section, we show an overview of the state-of-the-art approaches that align with our objectives, specifically in terms of (1) segmentation-based features [22], [23], [24], [29] and (2) NLP features [28], [34], [35], which are summarized in **Table 1**.

Note that we do not adopt Secure Socket Layer (SSL)-based features, which belong to NLP features in our definition, because SSL-based features consider domain details (such as the creation/expiration date and organization information); however, even in a certified domain, legitimate and phishing webpages are mixed, thereby, SSL-based features do not work. Besides, certified domains are not always trusted because of free certificate authority (CA) and self-signed certificates.

2.1 URL-based Phishing Detection Schemes with Segmentation-based Features

We show the prior research on segmentation-based features extracting characters or words, some of which combine segmentation-based features with NLP features.

In 2018, Le et al. [24] proposed the URLNet using CNN to de-

© 2024 Information Processing Society of Japan

tect malicious URLs. The URLNet captures the semantic meaning of words and sequential patterns of characters by adapting three levels of embeddings, i.e., character-, word-, and character-level word-embeddings. They also introduced advanced word embedding to solve the problem of infrequently appearing words, or unknown words, in the dataset, which leads to a high false-positive rate because of the unseen data. After tokenizing URLs with special characters (detailed in Section 3), they transform infrequent words, appearing less than 100 times among millions of word corpus, into unknown words. In character-level word embedding, they acquired 97.22 and 98.58 percent in true positive rate (TPR) with the same 0.1 percent in false positive rate (FPR) over 1 M and 5 M of training data size, respectively. However, their word-level embedding technique, similar to our approach, achieved only 96.64 and 96.73 percent TPR with the same 0.1 percent FPR, i.e., considerably low even for large training data. Besides, the computational complexity of the word-level CNN model is high because it performs character-level word embedding instead of naive word embedding, where word embedding = [https, www, google, com] and character-level word embedding = [[h,t,t,p], [w,w,w], [g,o,o,g,l,e], [c,o,m]]. Note that we compared their technique with our previous work in Ref. [31] to confirm that our work achieved higher accuracy and F1-measure.

Rao et al. [29], in 2020, proposed CatchPhish, a lightweight technique to detect phishing webpages by inspecting URLs. The CatchPhish was implemented with two sets of features: (1) handcrafted features comprised of hostname- and full URL-based features and (2) TF-IDF of phish-hinted words. They achieved 95.67, 93.95, and 94.26 percent accuracy in datasets D1, D2, and D3, shown in Table 1 (F1-measure: 95.81, 94.16, and 95.88 percent), respectively. However, the datasets they used misled the result because most URLs in their datasets have only domain names but lack the path part, unlike other works. Besides, their datasets contain contradicted ground-truth such that the same URLs have different labels. For example, docs.google.com is found in both legitimate and phishing labels. (discussed in Section 5.1).

Maneriker et al. [23], in 2021, developed URLTran using a transformer model with byte pair encoding (BPE) tokenizers to handle character subsequences. Using the standard BERT, their fine-tuned transformers achieved 99.67 percent accuracy (F1-measure: 99.71 and 86.80 TPR @ 0.01 FPR). However, they utilized the dataset collected from Microsoft's Edge and Internet Explorer browsing telemetry; thereby, comparing their results to other works without implementing the same model is difficult. Furthermore, the words segmented by BPE cannot form meaningful words (e.g., [bankofamerica] to [bank, ##of, ##ame, ##ri, ##ca]).

Ozcan et al. [22], in 2023, proposed a hybrid deep learning model to detect phishing URLs, which is constructed with long short-term memory (LSTM) and deep neural network algorithms (DNN-LSTM and DNN-BiLSTM) using character embedding and manually extracted NLP features. Their work obtained 98.79 percent accuracy (F1-measure: 98.81 percent). However, the limitation is that character embedding has the disadvantage of overwhelming unnecessary symbols (e.g., "%2B" or "%21"), which could be improved by adopting the URL decoder.

Jalil et al. [33], in 2023, proposed a machine learning framework to detect phishing URLs using three categories of features: URL-, domain-, and path-part features. They applied Relief feature selection technique [37] to extract the top 30 features from 100 features. However, similar to Ref. [29], they used dataset dependent features (i.e., TF-IDF of brand-name feature and phish-hinted word features), which are extracted solely based on specific URLs in a given dataset. Moreover, although their work achieved the best accuracy (ranging from 91.63 to 96.85) compared to the state-of-the-art works on four benchmark datasets as shown in Table 1, they failed to achieve a better F1-measure, compared to our work on the benchmark datasets in Section 5.6, which could be crucial on imbalanced dataset.

Wang et al. [38], in 2023, proposed PhishBERT, a bidirectional encoder representation-based pre-trained model. They applied word-level embedding, which adopted BPE, similar to Ref. [23]. They fine-tuned BERT with a regularization technique [39] to overcome the overfitting of the model. Their fine-tuned model achieved 92 percent in both accuracy and F1-measure. However, they have a low F1-measure compared to other state-of-the-art works although pre-training on 3 billion URLs. Moreover, similar to Ref. [23], BPE results in meaningless words.

Jishnu et al. [40] in 2023, proposed a hybrid model leveraging BERT with additional URL features. They adopted fine-tuned BERT model to train the concatenation of segmentation-based features, tokenized by BERT tokenizer, and lexical NLP-based features. Their proposed model achieved 97.32 percent accuracy (97.32 percent F1-measure). However, BERT tokenization resulted in meaningless words (e.g., bankofamerica to [bank, ##of, ##ame, ##ri, ##ca]).

The seven works mentioned above [22], [23], [24], [29], [33], [38], [40] decomposed raw URLs into character- or word-embeddings (or segmentation-based features); however, unknown words cannot be tokenized correctly. Additionally, the BERT tokenizer, inherited from the BPE, tokenizes a URL into a sequence of words, in which only the first appearing word is meaningful; however, the rest are a continuation of untokenized words.

2.2 URL-based Phishing Detection Schemes with NLP Features

This section systematically introduces state-of-the-art research on URL-based phishing detection schemes that utilize NLP features, also known as lexical features.

Marchal et al. [34], in 2014, introduced a novel phishing detection system called PhishStorm, which adopted an intra-URL relatedness score that expresses the relatedness among words between the registered domain name, i.e., brand name, and the rest of the URL. The intra-URL relatedness feature is calculated with the help of search engines by searching with the registered domain name. They claimed that the registered domain, such as "paypal," relates to the remaining part of the URL, such as the upper-level domain and path part. Although they achieved a promising performance of 94.91 percent in accuracy (F1-measure: 94.72 percent), their work heavily relied on third-party services, i.e., search engines, which resulted in low usabil-

© 2024 Information Processing Society of Japan

Table 2 Symbols Description.

Symbol	Definition	Example with a given URL u="www.bankofamerica.com/signonv2/appmanager/webcindario/pageunblockhel"
WI	$WI = \{wi_i \mid wi_i \in \text{URL } u, wi_i$ is the i-th word tokenized by special characters$\}$ (see Section 4.1)	{www, bankofamerica, com, signonv2, appmanager, webcindario, pageunblockhelp]
WC	$WC = \{wc_i \mid wc_i \in \text{URL } u, wc_i$ is the i-th word tokenized by special characters, followed by decomposing to consecutive letters and digits$\}$ (see Section 4.1)	{www, bankofamerica, com, **signonv**, **2**, appmanager, webcindario, pageunblockhelp}
WS	$WS = \{ws_i \mid ws_i \in \text{URL } u, ws_i$ is the i-th word tokenized by WordSegment tokenizer[3] $\}$ (see Section 4.1)	{www, bankofamerica, com, **sign**, **on**, **v**, **2**, appmanager, webcindario, **page, unblock, help**}
WR	$WR = \{wr_i \mid wr_i \in \text{URL } u, wr_i$ is the i-th word tokenized by Recursive tokenizer $\}$ (see Section 4.1)	{www, **bank, of, america**, com, sign, on, v, 2, **app, manager**, **web, cin, dario**, page, unblock, help}
WO	$WO = \{wo_i \mid wo_i \in \text{URL } u, wo_i$ is the i-th word tokenized by BERT Tokenizer[2] $\}$ (see Section 4.1)	{www, **bank, of, america**, com, sign, **on, v, 2**, **app, manager**, **web, ##cin, dario**, **page, unblock, help**}
DIC	English Dictionary	
ϕ	an empty set	

ity. Similar work relying on third-party services in extracting keywords can be found in Ref. [41].

In 2019, Sahingoz et al. [28] proposed a real-time anti-phishing system implementing both word-based features and NLP features with various classification algorithms without using any third-party services. They achieved 97.98 percent accuracy (F1-measure: 98 percent) with NLP features. Although they proposed the hybrid features in conjunction with word-based and NLP features, achieving higher accuracy to compare with NLP-based features, their models result lower F1-measures on average of their proposed classification algorithms.

Gupta et al. [35], in 2021, developed a novel approach using lexical-based machine learning in a real-time environment with carefully selected nine effective NLP features. They achieved a promising result of 99.57 percent accuracy (F1-measure: 99.58 percent). However, there is a limitation that such features are efficient only if URLs have a path part, similar to Ref. [28]. In other words, real-world URLs sometimes contain only the domain part and are missing the path part.

In 2022, Alani et al. [42] proposed a cloud-sided phishing detection approach with 14 features, including NLP features and newly defined features. The new features are (1) domain lookup time, (2) time-to-live (TTL), (3) days after domain activation, (4) days to domain expiry, and (5) autonomous system number. Although they achieved a promising performance of 97.48 percent accuracy, they have relied on cloud-side detection to gather the above features to update the machine learning models. Besides, the new features help detect domain-level reliability; however, a small set of NLP features results in the risk of false-negative detection if a reliable domain also provides phishing webpages.

2.3 Our Previous Work

We previously proposed the URL-Tokenizer algorithm [32] in 2021, with word-level segmentation. Compared to the traditional tokenizers, we successfully reduced the number of unknown words and achieved 97.7 percent and 95.7 percent accuracy in imbalanced and balanced datasets, respectively [32]. Then, we extended our work in 2022 [31] with 24 NLP features and constructed a hybrid model that utilized word-level segmentation-based features in Convolutional Neural Network (CNN) and NLP features in Deep Neural Network (DNN), followed by evaluation on Ebbu2017 [22], [28], [33] dataset. Although our previous works [31], [32] show promising result of 98.64 percent in accuracy (F1-measure: 98.65 percent), it is difficult to claim that the system is robust enough to work well on various datasets.

2.4 Summary

The works mentioned above, with their shortcomings, are summarized in Table 1. The existing problems can be stated in two categories: (1) insufficient segmentation leading to unknown-word problems and (2) dataset-dependent NLP features that cannot follow the phishers' shift in attack. In subsequent sections, we explain our previously proposed *SegURLizer* [31], [32] and newly refined 36 dataset-independent features. Then, we evaluate our new phishing URL detection model, called *PhiSN*, with state-of-the-art works on comprehensive datasets, including Ebbu [22], [28], [33], PhishStorm [34], ISCXURL [35], [36], CatchPhishD1 [29], and CatchPhishD2 [29]. Besides, we implement various models in conjunction with BiLSTM, LSTM, and DNN to compare the phishing detection performance, confirming the robustness of *PhiSN*.

3. Preliminaries

In this section, we primitively define the terminologies used in this paper, including special characters, symbols, and the structure of URLs. The common special characters that appear in URLs are defined as follows:

Special characters: colon [:], forward slash [/], dot [.], dash [-], underscore [_], semicolon [;], comma [,] equals [=], at [@], dollar [$], ampersand [&], question mark [?], plus [+], hash [#].

The symbols used in this paper are defined in **Table 2**, while the URL structure is defined in **Table 3**.

© 2024 Information Processing Society of Japan

Table 3 URL Definition.

Term	Definition	Example
Original URLs	-	• https://www.sub1.eg1.com/foo1/foo.html • https://www.sub2.sub1.eg2.co.jp:8080/foo.htm
FQDN	Fully qualified domain name consists of hostname.[optional subdomain(s)].domain.top-level-domain [43]	• www.sub1.eg1.com • www.sub2.sub1.eg2.co.jp
TLD	Top-level domain is the last part of an FQDN, which is served by the root name service [43]	• com • jp
eTLD	Effective top-level domain separates the responsibility of the registrant from registrars, which users can directly register. It is also called a public suffix[4].	• com • co.jp
eTLD+ 1	eTLD+1 is the suffix of FQDN, which users can directly register.	• eg1.com • eg2.co.jp
Brandname	Brandname is defined by subtracting eTLD from (eTLD+1)	• eg1 • eg2
Subdomain(s)	Subdomain(s) is the domain name given in the brandname organization.	• sub1 • sub2.sub1
Hostname	Host name refers to a device or a service by using an alias such as www or mail.	• www

4. PhiSN: Phishing URL Detection using Segmentation and NLP Features

URL-based phishing detection has been consistently analyzed by adopting diverse techniques such as NLP features (e.g., length of URL, number of special characters), segmentation-based features (i.e., character- and word-level tokenization, number of words, and text similarity), and hybrid features (i.e., a combination of various features). However, the remaining problems are (1) how to prevent the segmentation from extracting unknown-words and (2) how to construct dataset-independent NLP features. This section introduces two approaches as follows:

1. A new word-level tokenization algorithm called SegURLizer [32], which integrates BERT and WordSegment tokenizers, enhances the ability to tokenize a URL into meaningful words.
2. Dataset-independent NLP features are prepared as domain- and path-part features, eliminating dataset-dependent NLP features.

4.1 SegURLizer: Segmentation-based Phishing URL Tokenizer

This section presents SegURLizer (Segmentation-based URL Tokenizer) and Recursive-Tokenizer algorithms [32] to solve the unknown-word problems.

In phishing URL detection, traditional tokenization techniques using special characters (such as dot[.], dash[-], underscore[_], and slash[/] as described in Section 3) have been adopted to extract words from URLs [24]. However, such tokenization produces many unknown words because URLs do not consist of whitespace but consist of a combination of words without delimiters (e.g., bankofamerica). Meanwhile, common tokenization techniques, such as BERT or WordSegment tokenizer, cannot extract meaningful English words from URLs (e.g., [bankofamerica, appmanager] to [[bank, of, america],[app, manager]]). Meanwhile, our proposed SegURLizer [32] decomposes a URL into meaningful words.

The SegURLizer inherits WordSegment to utilize its successful decomposition of standalone words (e.g., knowledgeapplication to [knowledge, application]) or alphabetic words (e.g., webhostapp to [web, host, app]). Since WordSegment cannot decompose a complex combination of some short-term words (e.g., appmanager), the SegURLizer recursively tokenizes the remaining words with the Recursive-Tokenizer [32].

The Recursive-Tokenizer adopts the BERT tokenizer to segment the first word because the BERT tokenizer can tokenize the first-appearing word as a meaningful word. For instance, bankofamerica to [bank, ##of, ##ame, ##rica]. However, the BERT tokenizer does not correctly tokenize all the word combinations of the URL because of its "##" prefixing technique to recognize the continuation of a word sequence. Thus, the RecursiveTokenizer repeatedly decomposes a combination of words until the words can no longer be split, while using prefixing when the decomposed word is neither the first word nor exists in the English dictionary. We use PyEnchant [*5], an English spellchecking package, to validate whether the decomposed words are meaningful, i.e., English words.

SegURLizer: In **Algorithm 1**, the input set of the SegURLizer is shown as WI, which is a fragment of the URL (URL string) tokenized by the special characters. For example, if the original URL is www.bankofamerica.com/appmanager, WI consists of [www, bankofamerica, com, appmanager].

Then, the SegURLizer further decomposes each word in WI (i.e., w_i) to a list of consecutive letters or digits, shown as WC, to confirm if WI contains malicious character swapping (e.g., bank0famerica to [bank,0,famerica]). Then, WordSegment segments each word in WC (i.e., wc_i); however, WordSegment cannot segment a combination of short-term words (e.g., appmanager to [app, manager]). Thus, for each segmented word $ws_i \in WS$, we validate if ws_i is included in the English word corpus DIC. If ws_i is not an English word, we recursively apply the Recursive-Tokenizer until ws_i can no longer be decomposed. We then prefix the output word list WR with "##" if wr_i is neither the first word

[*4] Public Suffix List – Retrieved from https://publicsuffix.org/

[*5] PyEnchant – Python language bindings for the Enchant spelling checking library. Retrieved from https://pypi.org/project/pyenchant/

© 2024 Information Processing Society of Japan

Algorithm 1 SegURLizer [32]

Input : $WI \leftarrow \{wi_i \mid wi_i \in \text{URL } u\}$ // *tokenized URL u by special characters*

Output : $WO \leftarrow \{wo_i \mid wo_i \in \text{URL } u\}$ // *prefixed tokenized words by [##]*

$WO \leftarrow \phi$ // *for output set*
$WC \leftarrow \phi$ // *for decomposed set of consecutive characters*
$WS \leftarrow \phi$ // *for word-segmented set*
for $wi_i \in WI$ **do**
  if wi_i is alphanumeric **then**
    $WC \leftarrow WC \cup Decompose(wi_i)$
  else
    $WC \leftarrow WC \cup wi_i$
  end if
end for
for $wc_i \in WC$ **do**
  $WS \leftarrow WS \cup WordSegment(wc_i)$
end for
for $ws_i \in WS$ **do**
  $WR \leftarrow \phi$ // *for recursively tokenized set*
  if $ws_i \in DIC$ **then**
    $WR \leftarrow WR \cup ws_i$
  else
    $WR \leftarrow WR \cup Recursive\text{-}Tokenizer(ws_i)$
  end if
  for $wr_i \in WR$ **do**
    if $wr_{i\neq 1}$ AND $wr_i \notin DIC$ **then** // *$i \neq 1$ is the rest of the words, excluding the first word*
      $WO \leftarrow WO \cup ##wr_i$ // *for prefixing with [##]*
    else
      $WO \leftarrow WO \cup wr_i$
    end if
  end for
end for
return WO

Algorithm 2 Recursive-Tokenizer [32]

Input : *ws // the segmented word which does not exist in the English Dictionary*

Output : $WR \leftarrow \{wr_i \mid wr_i \in ws\}$ // *recursively tokenized by Bert*
$WR \leftarrow Bert\text{-}Tokenizer(ws)$

if length of $wr_i > 1$ **then**
  return $wr_1 \cup Recursive\text{-}Tokenizer(wr_{i\neq 1})$ // *$i \neq 1$ is the rest of the words excluding the first word*
else
  return wr_1
end if

nor included in *DIC*. This prefixing is adopted based on the concept of the BERT tokenizer.

Recursive-Tokenizer: The recursive-tokenizer, shown in **Algorithm 2**, tokenizes *ws* to return the no longer segmentable word list *WR*. The recursive-tokenizer applies the BERT tokenizer to extract the first meaningful word, wr_1, from the input word. The rest of the words, $wr_{i\neq 1}$, are returned from the recursive function to be further segmented. Notably, $i \neq 1$ represents the rest of the word after excluding the first word. In summary, we developed the SegURLizer to solve the problem of unknown words by taking advantage of WordSegment and BERT tokenizers.

4.2 Dataset-independent NLP Features

This section introduces the dataset-independent NLP features adopted in this study. We extracted the NLP features into three sets; domain-, path- and URL-part features. The detailed explanation of the hypothesis and applicability in different URL parts are described in Sections 4.2.1, 4.2.2, and 4.2.3. We summarized the differences among the features used in our previous study [31] and this work in each subsection.

4.2.1 Domain-part NLP Features

This section explains domain-part NLP-based features. We prepared 19 domain-part NLP features, where four features (D5, 6, 9, and 16) are the same as our previously implemented ones [31]. D13 and D17 are newly extended ones from Ref. [31]. The remaining 13 features are the previously adopted ones in the state-of-the-art works [22], [28], [29], [35]. In the following, each feature is described, where the data type and the target part in the URL are shown in parentheses.

D1, D2 - No. of Dot [.] (Integer, FQDN, and eTLD+1) represent the number of dots [.] appeared in the FQDN and eTLD+1, respectively. We hypothesize that phishing URLs include more dots to confuse users than legitimate URLs. For instance, "us-battle.net.login.en.rrweb.asia" has five dots in FQDN [us-battle.net.login.en.rrweb.asia] and one in eTLD+1 [rrweb.asia]. D1 and D2 were adopted in Refs. [29], [35]. We measure these features separately because the distribution of dots in the FQDN and eTLD+1 parts in phishing websites differs from that in legitimate websites. For example, "247.48.198.104.bc.googleuser content.com" (phishing) has the legitimate eTLD+1 [googleuser-content.com (the number of dots = 1)]; however, it has six dots in FQDN, which is larger than the ordinary legitimate website (e.g., lh5.googleusercontent.com).

D3, D4 - No. of NAN (Integer, FQDN, and eTLD+1) represent the number of non-alphanumeric characters (NAN, also known as special characters), such as hyphen [-] and underscore [_], appeared in the FQDN and eTLD+1, respectively. We hypothesize that phishers trick users by adding legitimate lookalike keywords and NAN characters. Note that we do not count dot [.] because it indicates the number of subdomains. For example, "secure--runecsape.fb-secure-awarded.in" has four and two NAN characters in the FQDN and eTLD+1, respectively. This feature was adopted in Refs. [22], [28], [35].

D5 - No. of suspicious words (Integer, FQDN) represents the total occurrence of suspicious words in the FQDN. A suspicious word is defined as a word containing both digits and letters. After tokenizing the FQDN by special characters, each extracted word is measured as suspicious if and only if the extracted word is neither lettered nor numeric but alphanumeric. We hypothesize that for the sake of mimicking legitimate URLs, phishing URLs are embedded or replaced with lookalike characters, such as '0' instead of 'o'; therefore, suspicious words can be found by finding character swapping in URLs. For example, in the case of "www.amaz0nesecure.me," the suspicious word is [amaz0n], which contains a mixture of letters and a digit (i.e., 'o' was replaced by '0'). Thus, the number of suspicious words becomes 1. This feature was adopted in Refs. [22], [28], [35] and our previous work [31].

© 2024 Information Processing Society of Japan

D6 - Entropy of NAN (Real, FQDN) represents the entropy distribution of NAN characters in the FQDN. We hypothesize that the distribution of NAN of legitimate URLs significantly differs from that of phishing URLs. It is set to '0' if no NAN is found. This feature is proposed in our previous work [44] and applied in our previous study [31].

$$\text{Entropy} = -\sum_{i \in T}^{|T|} P_i \log P_i, \tag{1}$$

where T is the set of NAN in the FQDN, and P_i is the probability that the i-th NAN character appears.

D7 - No. of Digits (Integer, FQDN) represents the number of digits in the FQDN. For instance, "77-23-239-111-dynip.superkbel.de" includes 10 digits. This hypothesis explains that the number of digits found in the phishing website is larger than that in the legitimate website because the legitimate website contains few or no digits in the FQDN. This feature was adopted in Refs. [22], [28].

D8 - No. of Alphabet (Integer, FQDN) represents the number of alphabets in the FQDN, excluding any non-alphabetic characters. For example, "us-battle.net.ok.ddweb.asia" contains 22 alphabets. This hypothesizes that the phishing website has more alphabets than the legitimate website because phishers use common words embedded in the FQDN to trick users. This idea comes from Refs. [22], [28].

D9 - IP Address Existence (Binary, FQDN) is a binary identifier to investigate whether the URL is represented by an IP address or FQDN. Taking "http://31.192.106.250" as an example. This hypothesis comes from the idea of excessive use of IP addresses instead of FQDN by phishers. This feature was applied in Ref. [34] and our previous work [31].

D10 - Average Length of Tokenized Word (Real, eTLD+1) is the average length of words after tokenized by special characters. We hypothesize that the average length of words used in phishing websites is either short with more special characters (e.g., "h3lp_dsk0-supp0t.yn.lt") or long with less special characters (e.g., "13453765871.googlegroups.com"). A similar feature (average word length) was adopted in Refs. [22], [29]; however, we rather measure the average length of words after tokenized by special characters. This can be represented as follows.

$$\sum_i LengthOf(w_i)/N, \tag{2}$$

where w_i shows the i-th word tokenized by special characters, and N is the total number of tokenized words.

D11 - Longest Length of Tokenized Word (Integer, eTLD+1) measures the longest word length among the tokenized words by special characters. Our hypothesis is that phishers tend to embed longer words as much as possible with no special character. Take "www.patienteducationconnect.com" as an example. A similar feature (longest word length) was conducted in Refs. [22], [29].

D12 - Brandname Existence (Binary, FQDN) represents whether the popular brandname exists in the subdomain(s). We filter the existence using Alexa top 500 brands and return 1 if the brandname is found in the subdomain(s); otherwise, it returns 0. This hypothesizes that phishers construct the URL, adding common brand keywords anywhere except the domain. For example, in the case of "http://https.www.amazon.com.account.security.update.informaton.jmscisb.com.amazondiscount1.000webhostapp.com," brandname ['amazon'] is mentioned in subdomains. This feature was used in Ref. [22].

D13 - Length of Brandname (Integer, eTLD+1) represents the character length of brandname in the eTLD+1. Compared to the D12 feature (i.e., direct matching brand names with Alexa 500), we define brandname as the subtraction of eTLD from (eTLD+1), as defined in Table 3. Then, the length of brandname is calculated by Eq.(3).

$$Brandname = LengthOf(eTLD + 1) - LengthOf(eTLD) \tag{3}$$

For example, in "http://www.amazon.com," eTLD+1 is [amazon.com] and eTLD is [com]. Therefore, the brandname is [amazon], and its length is 6.

D14 - Length of Domain (Integer, FQDN) represents the number of alphanumeric characters in FQDN except the hostname. For instance, "https://www.amazon.co.jp" contains 10 characters [amazon, co, jp]. This feature was implemented in Ref. [35] as the domain length.

D15 - Length of Subdomains(s) after eTLD+1 (Integer, FQDN) represents the length of characters in subdomain(s) after eTLD+1, including hostname. In case of "https://www.g00gle.abc.co.jp," eTLD+1 = [abc.co.jp], the subdomains = [www.g00gle] and the character length of subdomain = 10. It returns 0 for no subdomains of eTLD+1 (e.g., abc.co.jp) and -1 for an IP address occurrence in FQDN (e.g., 192.99.149.207). We hypothesize that phishers tend to use too-long or too-short subdomains to confuse users.

D16 - No. of Sub-level (Integer, FQDN) represents the number of sub-levels after eTLD+1. It returns 0 for an IP address occurrence in FQDN. We hypothesize that phishers use multiple sub-levels to deceive users by embedding popular keywords (e.g., in the case of "http://www.amazon.com.supportsbooks.fun," the number of subdomains = 3 [www, amazon, com] because the eTLD+1 is [supportsbooks.fun]). Notably, compared with D1 and D2, which measure the number of dots [.] regardless of their position, we separately measure D16 in the sub-levels after eTLD+1, indicating the hierarchy of the subdomains. This was applied in our previous work [31].

D17 - gTLD Existence (Binary, FQDN) checks whether a TLD uses a generic TLD (gTLD). The APWG report in the 4-th quarter of 2021 [*6] suggested that 52 percent of phishing URLs were composed of legacy gTLD. We check whether the eTLD in the URL consists of the gTLD, such as [.com, .info, .net, .org, .biz, .name, .pro, .arpa, .aero, .asia, .cat, .coop, .edu, .gov, .int, .jobs, .mil, mobi, .tel, .travel, .xxx]. This hypothesizes that the phishing URL uses gTLD more than ccTLD (country code TLD) or a new TLD, although there are reports regarding vulnerabilities found in new TLDs.

D18 - No. of Hyphen[-] (Integer, FQDN) counts the number of hyphen [-] appeared in the FQDN. We hypothesize that the phishing URL includes more hyphen [-] than the legitimate URL.

[*6] APWG – 4th quarter report of 2021. https://docs.apwg.org/reports/apwg_trends_report_q4_2021.pdf

© 2024 Information Processing Society of Japan

For instance, "id-mobile-free-frsub.com." This feature was used in Ref. [29].

D19 - No. of Underscore [_] (Integer, FQDN) counts the number of underscores [_] in the FQDN. Similar to hyphen [-], the hypothesis is that the phishers add more underscores in the URL to confuse users, while the legitimate URL contains a few underscores. For example, "https://amazon_co_jp.taobaomall.org." This was applied in Ref. [29].

4.2.2 Path-part NLP Features

This section elucidates the path-part NLP features extracted from the path part of the URL. Among these features, P6, 8, and 9 were utilized in our prior research [31], while P4 and P5 are the newly extended features. The remaining features are commonly employed in state-of-the-art works [29], [35].

P1 - No. of Alphanumeric (Integer) represents the occurrences of alphanumeric characters found in the path part. After discarding NAN characters (including slash[/] for directory), we sum up the number of characters regardless of different directories because we hypothesize that the phishing URL is generated randomly with longer combinations. For example, "canvashub.com/myfiles/1495e14e971be5156fae9cf0753e2509/" has 39 alpha-numeric characters in the path part.

P2 - No. of Directories (Integer) is used to measure the number of directories in the path part. Phishing webpages are structured with many sub-directories to distract users' poor awareness. For example, "hamtags.pingfiles.fr/wp-content/uploads/2017/09/Summary/paypal/LimitD/rebots/customercenter/customer-IDPP00C763/myaccount/signin/". This feature was used in Ref. [29].

P3 - No. of NAN (Integer) represents the number of NAN characters appeared in the path part. Phishers confuse users by adding randomly generated characters containing several NAN characters. Notably, we ignore the slash [/] because it indicates the number of directories in the path part. For example, "puspapharma.com/_private/.https-www/sellercentral.amazon.com" has five NAN characters in the path part. This feature was applied in Ref. [35].

P4, 5 - Distribution of Digits and Alphabets (Real) represent the digit (or alphabet) distribution in the path part. In addition to considering the number of characters in the path part (P1 and P3), we hypothesize that computing the character distribution across different directories might reveal a notable distinction between phishing and legitimate websites. We adopt "poison distribution" to gauge the discrete probability of the frequency with which a digit (or alphabet) appears. We introduce this feature as the probability of digits (or alphabet) in one sub-directory does not influence another sub-directory. The formula for calculating this distribution is presented below.

$$P = \frac{e^{-\lambda}\lambda^{nd}}{nd!}, \tag{4}$$

$$\lambda = \frac{w * nd}{N_{dir}}, \tag{5}$$

$$w = \frac{n}{N}, \tag{6}$$

where w is the weight of digits (alphabets), n is the number of digits (alphabets) in the path part, N is the length of the path part excluding special characters defined in Section 3 (i.e., the total number of digits and alphabets in the path part), λ is the weighted average of digits (alphabets), nd is the number of directories that digits (alphabets) appear, and N_{dir} is the number of directories in the path part.

P6 - Entropy of NAN (Real) represents the entropy distribution of NAN characters in the path part. Similar to the entropy of NAN in the domain-part (D6), it is set to '0' if no NAN is found. This feature was proposed in our previous study [44] and applied in Ref. [31].

P7 - No. of Digits (Integer) represents the number of digits in the path part. For instance, "canvashub.com/myfiles/1495e14e971be5156fae9cf0753e2509/" includes 22 digits. This hypothesis explains that the number of digits that appear in the path part of the phishing website is larger than that in the legitimate website. This feature was used in Ref. [29].

P8 - No. of Redirection (Integer) indicates whether the path-part contains an internal link [http, www] redirection. The embedded link tends to be used to redirect to a phishing webpage. We hypothesize that phishers deceive users with internal link redirection by ignoring the content before [http, www]. For instance, "https://rwwwr.com/4/http://gg.gg/919", "http://raspberrygreen.com/css/www.facebook.com/Login". Similar feature can be found in Refs. [29], [31].

P9 - Query Existence (Binary, Query) is a binary identifier to check whether a query exists in the path part. This hypothesizes that phishing URLs are more likely to contain a query than a legitimate URL. This feature was applied in Refs. [29], [31].

P10, 11 - Average and the longest Length of Tokenized Word (Real) measure similar to those in domain-part above, i.e., the average (D10) and the longest (D11) length of tokenized words, but in the path.

P12 - Brandname Existence (Binary, Path) checks whether the brandname exists in the path part. In contrast to Ref. [22], in which they calculated the similarity between the brandnames in Alexa and the URL, we applied direct matching to confirm the existence of the brandnames in the path part. We hypothesize that the phisher can add brandnames to the path part. For example, "puspapharma.com/_private/https-www/sellercentral.amazon.com."

4.2.3 URL-part NLP Features

This section explains the URL-part features measured in the entire URL. These features were adopted in Refs. [29], [35].

U1 - [@] Existence (Binary) is a binary identifier that indicates whether an embedded redirection symbol at-marks [@] exists in the URL. The hypothesis is that at-mark [@] is the character frequently used by phishers to redirect to a different web address (e.g., "https://jflkp.csb.app/index.html#a@b.com"). This feature can be found in Ref. [29].

U2 - Length of URL (Integer) represents the length of the entire URL, including special characters. We hypothesize that the length of a phishing URL tends to be longer than that of a legitimate URL. This feature was utilized in Ref. [35].

U3 - No. of [?] (Integer) represents the number of question marks [?] found in the URL. This naive feature is also implemented in [29]. We hypothesize that the number of question marks [?] used in legitimate and phishing is different because

© 2024 Information Processing Society of Japan

the phishing URL encodes the question marks into [%3F]. Therefore, we first decode the URL and measure the number of question marks present.

U4 - No. of [/] (Integer) represents the number of forward slashes [/] found in the URL without considering whether [/] is located in the URL (P2 feature) or [//] is used for redirection (P8 feature). This is a naive way of measuring forward slashes, similar to Ref. [29].

U5 - Ratio. of [-] (Real) measures the ratio of the number of hyphens [-] found in the URL to the length of the URL. This is the same feature used in Ref. [29].

4.3 Architecture of Our Proposed Model

This section addresses the preprocessing of URLs, followed by a detailed description of the architecture of our hybrid model.

4.3.1 Preprocessing

We outline a series of preprocessing tasks, including segmentation-related preprocessing and other general steps.

1. IP2Domain Conversion [*7] (used for Segmentation-based feature extraction) - It transforms an IP address in the domain part of the URL to its associated domain name. This conversion is performed by SegURLizer (see Section 4.1). Notably, this conversion does not apply when the NLP-based features are extracted.
2. Vocabulary dictionary (used for Segmentation-based Feature extraction) - It consists of the most frequently appeared top-5k words in the URLs of the training dataset. Each word is segmented by the proposed SegURLizer.
3. Truncation - We restricted the maximum number of words of URLs to 200, which is the same as Ref. [24], whereas we discarded the characters after '?' in the path-part of NLP-based features.
4. Cross-validation - We employed a 10-fold cross-validation strategy.

4.3.2 Hybrid Model Architecture

Figure 2 illustrates our hybrid model architecture combining an individual segmentation-based model (long short-term memory (LSTM) or bidirectional LSTM (BiLSTM)) with an NLP-based model trained by a deep neural network (DNN) using the 36 features in Section 4.2. Figure 2 is extended from our previously proposed hybrid model architecture [31], combining segmentation-based CNN and NLP-based DNN using 24 features.

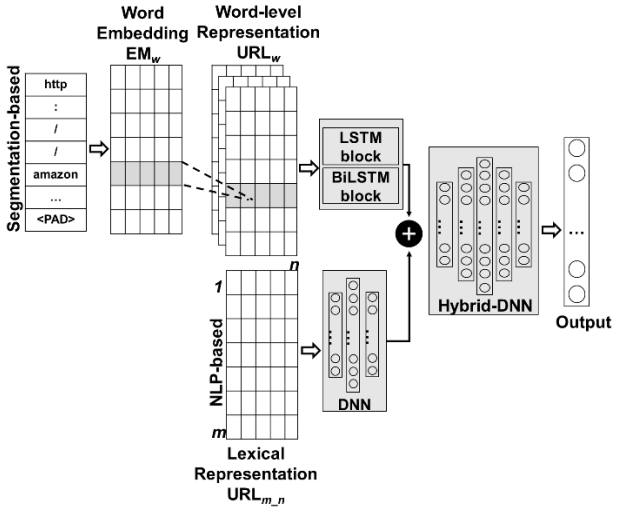


Fig. 2 Hybrid Model Architecture.

The segmentation-based model trains the tokenized URL, where the inputs to the model are word-vectors padded with a maximum length of 200 words [31]. The NLP-based DNN model adopts 36 NLP features as an input of n dimensions with an m-fold number of URLs. The model is constructed after parameter tuning in Section 5.5, varying the different number of layers. Then, we combine the NLP-DNN with the segmentation-based models, forming DNN-LSTM and DNN-BiLSTM. Finally, a hybrid DNN model is constructed to leverage the detection accuracy. In summary, distinct hybrid models are constructed to compare with various state-of-the-art works and ensure a robust detection system. Section 5.3 presents the detailed models.

5. Experimental Evaluation

In this section, we comprehensively evaluate our proposed models by comparing them to state-of-the-art works serving as baselines. We adopted five open datasets (Section 5.1), which are widely utilized in this research area, and established 15 baselines (Section 5.2). We compared our segmentation-based and hybrid-based models (explained in Section 5.3) with the baselines discussed in Section 5.6. Section 5.7 evaluates the ROC-AUC of our hybrid-based models to confirm the effectiveness of the models with the highest true positive rate (TPR) at the lowest false positive rate (FPR). Finally, Section 5.8 confirms the confidence intervals on our hybrid models to ensure the stability of the models' performance with the least fluctuated evaluation.

5.1 Datasets

The datasets used in our experiments are detailed in **Table 4**. We utilized five open datasets that are readily available and have been distributed through Refs. [28], [29], [34], [36] [*8,*9,*10,*11]. These datasets are recognized as common choices for evaluating

Table 4 Datasets.

Dataset Name	Adopted papers	Data Sources	No. of URL	
			Leg.	Phish.
Ebbu[8]	[22],[28], [33]	Yandex Search API, PhishTank	36,400	37,175
PhishStorm[9]	[34]	DMOZ, PhishTank	48,009	47,902
ISCXURL[10]	[35]	University of Canada Brunswick	10,000	9,965
CatchPhish D1[11]	[29]	Alexa, PhishTank	43,189	40,668
CatchPhish D2[11]		Common Crawl, PhishTank	42,220	40,668

[*7] Socket – Low-level networking interface to retrieve the hostname corresponding to the given IP address. Retrieved from https://docs.python.org/3/library/socket.html

[*8] Ebbu – https://github.com/ebubekirbbr/pdd/tree/master

[*9] PhishStorm – https://research.aalto.fi/en/datasets/phishstorm-phishing-legitimate-url-dataset

[*10] ISCXURL – https://www.unb.ca/cic/datasets/url-2016.html

[*11] CatchPhish – https://drive.google.com/file/d/1knSv_SKu6I--k4UpT1-fG1hj2VXaGNh1/view

© 2024 Information Processing Society of Japan

Table 5 Baselines.

Baseline Name (Author)-(S|H)-(model(/model)) S:segmentation-based H:hybrid model	Dataset for Evaluation [Reference]	Descriptions
OZ-S-BiLSTM	Ebbu [22]	BiLSTM model trained on segmentation-based character-level embedding features. (10-fold cross-validation)
OZ-S-LSTM		LSTM model trained on segmentation-based character-level embedding features. (10-fold cross-validation)
OZ-H-DNN/BiLSTM		Hybrid model built on segmentation-based BiLSTM and NLP-based DNN. (10-fold cross-validation)
OZ-H-DNN/LSTM		Hybrid model built on segmentation-based LSTM and NLP-based DNN. (10-fold cross-validation)
JA-H-RF	Ebbu [33]	Random Forest model trained on Segmentation-based feature (TF-IDF) and NLP-based features. (Train: 70%, Test: 30%)
SA-S-RF	Ebbu [28]	Random Forest model trained on Segmentation-based (Word decomposer) feature. (10-fold cross-validation)
SA-H-RF		Random Forest model trained on Segmentation-based and NLP-based features. (10-fold cross-validation)
MA-H-RF	PhishStorm [34]	Random Forest model trained on Segmentation-based word-splitter feature and its intra-URL relatedness features. (10-fold cross-validation)
GU-N-RF*	ISCXURL [35]	Random Forest model trained on NLP-based features. (Train: 80%, Test: 20%)
RA-S-RF	CatchPhishD1 CatchPhishD2 [29]	Random Forest model trained on Segmentation-based feature (TF-IDF) features. (10-fold cross-validation)
RA-H-RF		Random Forest model trained on Segmentation-based and NLP-based features. (10-fold cross-validation)

Table 6 Proposed Models.

Model	Description
AU-S-BiLSTM	BiLSTM model trained on Segmentation-based features tokenized by SegURLizer
AU-S-LSTM	LSTM model trained on Segmentation-based features tokenized by SegURLizer
AU-H-DNN/BiLSTM	Hybrid DNN model built on Segmentation-based BiLSTM and NLP-based DNN
AU-H-DNN/LSTM	Hybrid DNN model built on Segmentation-based LSTM and NLP-based DNN

phishing URL detection methods, as acknowledged in previous papers [22], [29], [34], [35]. These datasets have balanced class distribution and were collected from different sources, including PhishTank, Common Crawl, Alexa, DMOZ, Yandex Search API, and the data repository of the University of Canada Brunswick, as presented in Table 4. Notably, (1) the dataset PhishStorm was collected on June 2, 2022, and (2) we omitted the use of CatchPhishD3 because CatchPhishD3 is a combination of CatchPhishD1 and CatchPhishD2. Moreover, CatchPhishD3 includes the contradicted ground truth, where the same domain-only URLs were found in both legitimate and phishing URLs.

5.2 Baselines

In our evaluation, we compared our proposed models with 15 state-of-the-art baselines drawn from five datasets introduced in six papers [22], [29], [34], [35], as outlined in **Table 5**. These baselines are categorized as either segmentation-based or hybrid models, where hybrid models combine two distinct models. To ensure accuracy and consistency in our comparisons, we referenced the evaluation results provided in each respective paper. This approach was necessary as some of these papers did not make their source code available, preventing us from directly implementing and evaluating their models. Notably, we excluded models that did not utilize open datasets to ensure a fair and equitable comparison of results.

5.3 Proposed Models

Table 6 provides an overview of our proposed models, which encompass four variations, including both segmentation-based and hybrid models. The hybrid models incorporate a DNN to concatenate two single models, specifically the segmentation-based and NLP-based models, as depicted in Fig. 2.

5.4 Evaluation Metrics

We applied comprehensive evaluation metrics, including accuracy, precision, recall, and F1-measure, as defined in the following formulas.

$$TP = \textit{Correctly classified as Phishing} \tag{7}$$

$$TN = \textit{Correctly classified as Legitimate} \tag{8}$$

$$FP = \textit{Incorrectly classified as Phishing} \tag{9}$$

$$FN = \textit{Incorrectly classified as Legitimate} \tag{10}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$F1\ measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{14}$$

5.5 Parameter Tuning

We conducted tuning for eight hyperparameters of our proposed models, as described in **Table 7**. These models consist of four single models: 1) segmentation-based LSTM (S-LSTM), 2) segmentation-based BiLSTM (S-BiLSTM), 3) NLP-DNN (N-DNN), and 4) hybrid-DNN (H-DNN). In Fig. 2, the upper left portion shows S-LSTM or S-BiLSTM, the lower left portion shows N-DNN, and the right portion shows H-DNN.

Before initiating the tuning process, we fixed two parame-

© 2024 Information Processing Society of Japan

Table 7 Selected Parameters.

Parameter Name	Model	Parameter
Embedding Dimension	S-BiLSTM	64
	S-LSTM	128
No. of Layers	S-BiLSTM	8
	S-LSTM	2
	N-DNN	2
No. of Units	S-BiLSTM	64
	S-LSTM	32
	N-DNN	64
	H-DNN	16
Dropout Rate	S-BiLSTM	0.4
	S-LSTM	0.5
	N-DNN	0.1
	H-DNN	0.1
Initializer	S-BiLSTM	Glorot Uniform
	S-LSTM	Truncated Normal
	N-DNN	Random Uniform
	H-DNN	Glorot Uniform
Optimizer	H-DNN	Adamax
Learning Rate	H-DNN	0.001
Batch Size	H-DNN	32

ters: loss function ('binary cross entropy') and epoch size of 100. Then, the other parameters were tuned, where the parameters tuning was performed using a 5% sampled dataset from the five datasets shown in Table 4. When we tuned, we applied 10-fold cross-validation on the sampled dataset and selected the parameters with the best-averaged F1-measure. Finally, the tuned parameters are shown in Table 7.

The tuning process followed two stages. First, we tuned the three single models (S-LSTM, S-BiLSTM, and N-DNN) to identify the most suitable parameter configurations. Then, we tuned the H-DNN model, which combines two single models: 1) S-LSTM and N-DNN and 2) S-BiLSTM and N-DNN. This tuning process led to the final two hybrid models, namely AU-H-DNN/LSTM and AU-H-DNN/BiLSTM.

5.6 Evaluation Results

This section presents the evaluation results to compare our proposed models with state-of-the-art baselines. **Table 8** provides a dataset-by-dataset comparison of the results. Furthermore, the models are categorized based on their types, either segmentation-based or hybrid-based, as explained in Section 5.2. It is important to note that the evaluation results from each state-of-the-art paper were cited, resulting in different sets of baselines depending on the dataset and prohibiting us from applying statistical tests.

Table 8 shows that our proposed hybrid-based models outperformed all models across all evaluation metrics. Even when focusing on segmentation-based models, our proposed segmentation-based models excelled in all measures. We achieved F1-measure values ranging from 95.30% to 99.75%, depending on the datasets, representing a notable improvement of 0.13% to 3.83% in the absolute value of F1-measure, as illustrated in **Fig. 3**. Figure 3 visually compares our hybrid models with the best models among the baselines from each respective paper.

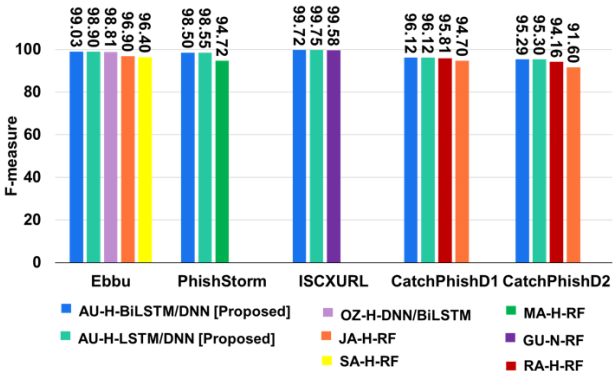


Fig. 3 Comparison of F1-measure among Our Models and Baselines.

5.7 Area Under Curve (ROC-AUC) of Our Proposed Hybrid-based Models

Figure 4 shows the ROC-AUC curve, depicting the true positive rates versus false positive rates (TPR @FPR) for our proposed hybrid-based models. ROC-AUC is a crucial evaluation measure because the phishing URL detection must have a low FPR with a higher TPR [23]. This additional comparison allows us to identify the best-performing hybrid model with a high TPR at a low FPR.

We define TPR and FPR as follows.

$$TPR = \frac{TP}{TP + FN} \tag{15}$$

$$FPR = \frac{FP}{TN + FP} \tag{16}$$

Our evaluation confirms that AU-H-DNN/LSTM performs superiorly to AU-H-DNN/BiLSTM when the FPR is sufficiently low, around 10^{-3}. However, it is important to note that this trend does not hold for the CatchPhish D1 dataset. Notably, the results for the CatchPhish datasets (D1 and D2) differ from the other datasets due to both models achieving relatively low TPR while maintaining low FPR values in these two datasets compared to the other three. We analyzed the distribution of URLs in each dataset, as shown in **Fig. 5**, to clarify why we have different results on CatchPhish datasets.

Figure 5 shows that CatchPhish datasets contain many URLs without path-part compared to the other three datasets. In CatchPhish D1, nearly all legitimate URLs lack a path-part. In CatchPhish D2, the number of URLs with a path-part is fairly evenly distributed between legitimate and phishing URLs. However, URLs without a path-part exhibit a bias towards phishing instances in CatchPhish D2, which shows that the path part is essential to detect phishing URLs.

5.8 Difference between Our Proposed Two Hybrid-based Models

This section analyzes the confidence interval (CI) between our proposed two hybrid-based models, as presented in **Table 9**. This statistical test is designed to assess the stability of the model's performance in terms of its ability to generalize to unseen data when subjected to various cross-validation folds, which are also commonly referred to as samples within the context of testing data. Moreover, we conducted a statistical test to determine the best hybrid model with the least fluctuated evaluation (i.e., narrower

© 2024 Information Processing Society of Japan

Table 8 Evaluation Results.

Dataset	Model		Metrics			
	Type	Model Name	Accuracy	Precision	Recall	F1-measure
Ebbu	Segmentation-based	OZ-S-BiLSTM [22]	97.58	N/A	N/A	97.61
		OZ-S-LSTM [22]	98.24	N/A	N/A	98.28
		SA-S-RF [28]	83.14	95.80	69.70	80.70
		AU-S-BiLSTM [Proposed]	98.97	98.63	99.35	98.99
		AU-S-LSTM [Proposed]	98.75	98.32	99.23	98.77
	Hybrid-based	OZ-H-DNN/BiLSTM [22]	98.79	N/A	N/A	98.81
		OZ-H-DNN/LSTM [22]	98.62	N/A	N/A	98.65
		JA-H-RF [33]	96.85	96.50	97.40	96.90
		SA-H-RF [28]	96.36	95.30	97.60	96.40
		AU-H-DNN/BiLSTM [Proposed]	**99.02**	**98.70**	**99.36**	**99.03**
		AU-H-DNN/LSTM [Proposed]	98.88	98.53	99.27	98.90
PhishStorm	Segmentation-based	AU-S-BiLSTM [Proposed]	98.35	98.31	98.40	98.35
		AU-S-LSTM [Proposed]	98.35	98.39	98.31	98.35
	Hybrid-based	MA-H-RF [34]	94.91	N/A	98.44	94.72
		AU-H-DNN/BiLSTM [Proposed]	98.51	98.45	**98.56**	98.50
		AU-H-DNN/LSTM [Proposed]	**98.55**	**98.59**	98.50	**98.55**
ISCXURL	Segmentation-based	AU-S-BiLSTM [Proposed]	99.72	99.73	99.71	99.72
		AU-S-LSTM [Proposed]	99.73	99.76	99.69	99.72
	Hybrid-based	GU-N-RF [35]*	99.57	99.70	99.46	99.58
		AU-H-DNN/BiLSTM [Proposed]	99.73	99.73	99.72	99.72
		AU-H-DNN/LSTM [Proposed]	**99.76**	**99.76**	**99.75**	**99.75**
CatchPhishD1	Segmentation-based	RA-S-RF [29]	94.01	94.21	94.17	94.19
		AU-S-BiLSTM [Proposed]	95.59	96.44	94.39	95.40
		AU-S-LSTM [Proposed]	95.62	96.10	94.82	95.46
	Hybrid-based	RA-H-RF [29]	95.67	96.31	95.32	95.81
		JA-H-RF [33]	94.80	94.70	94.60	94.70
		AU-H-DNN/BiLSTM [Proposed]	**96.26**	**96.63**	95.62	**96.12**
		AU-H-DNN/LSTM [Proposed]	96.25	96.44	**95.81**	**96.12**
CatchPhishD2	Segmentation-based	RA-S-RF [29]	92.58	94.76	91.04	92.86
		AU-S-BiLSTM [Proposed]	94.92	95.24	94.36	94.80
		AU-S-LSTM [Proposed]	94.85	95.38	94.07	94.72
	Hybrid-based	RA-H-RF [29]	93.95	95.71	92.66	94.16
		JA-H-RF [33]	91.63	91.70	91.50	91.60
		AU-H-DNN/BiLSTM [Proposed]	95.39	95.64	**94.94**	95.29
		AU-H-DNN/LSTM [Proposed]	**95.41**	**95.87**	94.73	**95.30**

* GU-N-RF: Extracted only NLP-based features. We compared our hybrid model with this baseline because we adopted the same dataset.

interval). Consequently, our analysis has led to the following confirmation: 1) When observing overlapping confidence intervals with marginal variation within the range of 95% confidence intervals, it becomes evident that choosing between the two models does not yield a definitive distinction, and 2) our investigation has revealed that there is no discernible difference in performance between AU-H-DNN/BiLSTM and AU-H-DNN/LSTM.

6. Discussion

This section discusses (1) the impact of Internationalized Domain Names (IDNs), (2) the impact of percent-encoding, (3) the impact of path-part, (4) the characteristics of misclassified URLs, and (5) the significant URL trends over time.

6.1 Impact of Internationalized Domain Names (IDNs)

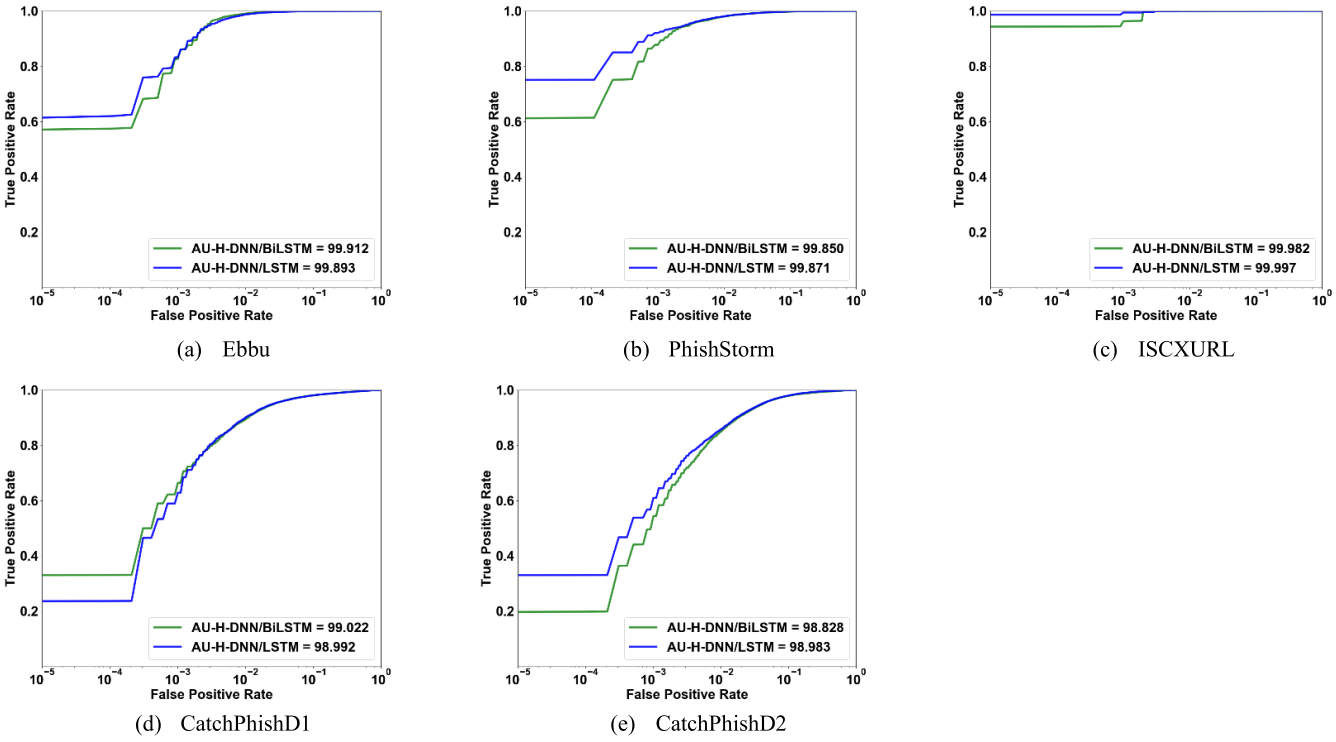
Punycode encoding is used for representing the Internationalized Domain Names (IDNs), which is a simple and efficient transfer encoding syntax designed for use in Internationalized Domain Names in Application (IDNA) [*12]. In our proposed method, we neither decode nor check URLs whether they embed Punycode as IDN. Instead, we treat URLs as they are and extract NLP features.

In order to compare the performance difference between URLs w/ and w/o IDNs, we analyzed the benchmark datasets. We extracted URLs containing "xn- -" from the datasets, where Punycode starts with "xn- -". Punycode is used in (1) domain-parts (used for IDN, e.g., [ドメイン名例.JP as xn--eckwd4c7cu47r2wf.jp]) and (2) path-parts (used for multilingual characters, e.g., [/myetherwałlett.net/signmsg.html as /xn--myetherwaett-lcca.net/signmsg.html]). We discovered a small number of URLs embedding Punycode; e.g., [paypàl.de-aktualisierung.info as xn--paypl-uqa.de-aktualisierung.info]. The comparison between the results of the URLs containing IDNs and all URLs is shown in **Table 10**.

As shown in Table 10, the F1-measure of the URLs containing IDNs is smaller than that of all URLs except for the PhishStorm dataset, showing that we have space to enhance the performance of phishing detection targeting the URLs embedding Punycode, which is remained as our future work.

[*12] Internationalized Domain Names in Application (IDNA) – https://www.ietf.org/rfc/rfc3492.txt

© 2024 Information Processing Society of Japan



(a) Ebbu (b) PhishStorm (c) ISCXURL

(d) CatchPhishD1 (e) CatchPhishD2

Fig. 4 Comparison of ROC-AUC curve of Our Proposed Two Hybrid Models on Five Datasets.

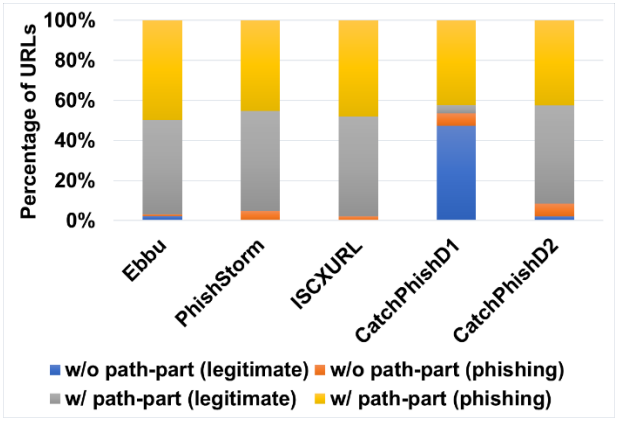


Fig. 5 Distribution of URLs in Each Dataset.

Table 9 Confidence Intervals for Each Dataset.

Dataset	Hybrid Model	ACC	CI (lower)	CI (upper)
Ebbu	AU-H-DNN/BiLSTM	99.02	0.9896	0.9907
	AU-H-DNN/LSTM	98.88	0.9866	0.9910
PhishStorm	AU-H-DNN/BiLSTM	98.51	0.9843	0.9859
	AU-H-DNN/LSTM	98.55	0.9847	0.9863
ISCXURL	AU-H-DNN/BiLSTM	99.73	0.9966	0.9979
	AU-H-DNN/LSTM	99.76	0.9971	0.9980
CatchPhish D1	AU-H-DNN/BiLSTM	96.26	0.9610	0.9641
	AU-H-DNN/LSTM	96.25	0.9604	0.9647
CatchPhish D2	AU-H-DNN/BiLSTM	95.39	0.9520	0.9554
	AU-H-DNN/LSTM	95.41	0.9527	0.9556

6.2 Impact of Percent-encoding

In our proposed method, we decode URLs by replacing "%xx" with its single-character equivalent using URL parser[*13] to extract NLP features (such as U3 Feature). The comparison between the result of the URLs containing percent-encoding and all URLs is shown in **Table 11**. As shown in Table 11, the F1-measure of the URLs containing percent-encoding is comparable with that of all URLs, indicating that we can precisely detect URLs embedding percent-encoding.

[*13] URL Parser – https://docs.python.org/3/library/urllib.parse.html

Table 10 Comparison of URLs containing IDNs and all URLs.

Dataset	URLs containing IDNs					All URLs
	TP	FN	TN	FP	F1-measure	F1-measure (Table 8)
Ebbu	31	0	0	2	96.88	**99.03**
PhishStorm	32	0	0	0	**100.00**	98.55
ISCXURL	7	1	0	0	93.33	**99.75**
CatchPhishD1	66	21	86	1	85.71	**96.12**
CatchPhishD2	80	7	103	11	89.89	**95.30**

Table 11 Comparison of URLs containing Percent-encoding and all URLs.

Dataset	URLs containing percent-encoding					All URLs
	TP	FN	TN	FP	F1-measure	F1-measure (Table 8)
Ebbu	1,178	0	1,304	16	**99.33**	99.03
PhishStorm	3,197	8	174	5	**99.80**	98.55
ISCXURL	365	3	1,949	4	99.05	**99.75**
CatchPhish D1	1,554	10	340	48	**98.17**	96.12
CatchPhish D2	1,520	44	2,222	147	94.09	**95.30**

6.3 Impact of Path-part

To analyze the impact of path-part, we compared our results among (i) All URLs, (ii) URLs w/ path-part, and (iii) URLs w/o path-part, as shown in **Fig. 6**. Our models achieved 95% to 99% of F1-measure w/ path-part (with small deviations), whereas 70%

© 2024 Information Processing Society of Japan

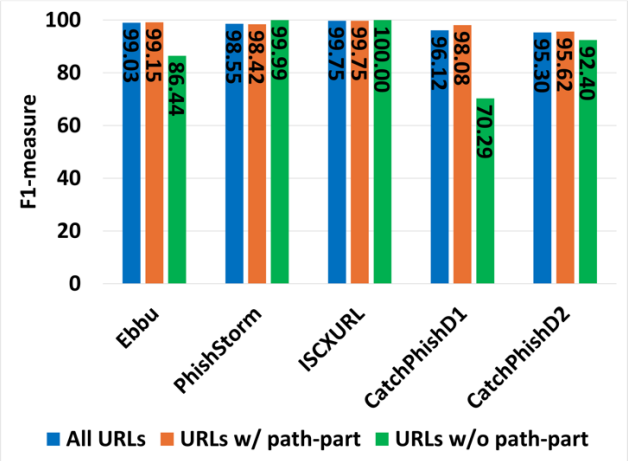


Fig. 6 Comparison of F1-measure among (i) All URLs, (ii) URLs w/ Path-part, and (iii) URLs w/o Path-part.

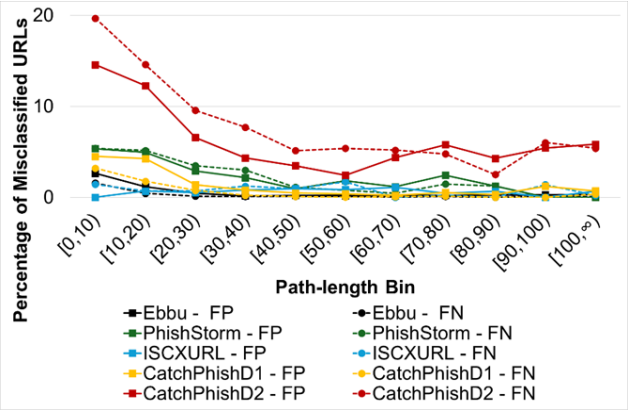


Fig. 7 Misclassifications over Path-length.

to 100% of F1-measure w/o path-part (with high deviations). This finding supports our stance on the necessity of the path-part discussed in Section 5.7.

6.4 Characteristics of Misclassified URLs (FP and FN)

To analyze characteristics in URLs classified as FP and FN, we illustrate a line graph of FP and FN @ same bin length of path-part, i.e., the ratio of the number of misclassifications to the total number of URLs in corresponding bins ranging from $[0, 10)$ to $[100, \infty)$. We hypothesize that there is a crucial and common influence in the length of path-part on all datasets. **Figure 7** shows that the shorter the length of the path-part, the larger the number of misclassified URLs (FP and FN), which applies to all five datasets. This finding supports our stance of the necessity of path-part discussed in Section 5.7, which shows the limitation of URL-based phishing detections for short URLs.

6.5 Significant URL Trends over Time

We analyzed the performance difference between our used benchmark datasets and a newly collected phishing dataset called PhishTank 2023 to confirm the robustness of our proposed method.

The PhishTank 2023 dataset consists of 254,502 phishing URLs newly registered on phishtank.org in 2023. Our findings in the PhishTank 2023 dataset include (1) the existence of new

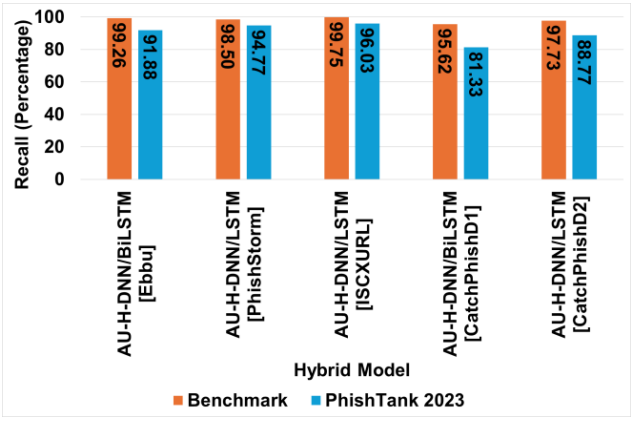


Fig. 8 Performance Comparison on the New Dataset - PhishTank 2023.

TLDs, and (2) a significant number of random characters in the path-part of URLs.

To confirm that our models work well on new unseen data, we tested each of our proposed best models trained on each benchmark dataset with the PhishTank 2023 dataset. Then, we compared the recall of phishing detection. It is worth to notice that the PhishTank 2023 dataset consists only of phishing URLs; thus, we use the performance metric of recall. The best model is shown as (Author)-(H)-(Model/(Model))[Dataset], such as AU-H-DNN/LSTM[PhishStorm], which is the same notation defined in Section 5. **Figure 8** shows the performance comparison between our used benchmark datasets and the PhishTank 2023 dataset. The degradation of recall is 3.72% to 14.29%, depending on the model trained on different benchmark datasets. Significant degradation is exhibited, especially with the model trained on CatchPhishD1 and CatchPhishD2. This is because CatchPhish datasets contain a large number of URLs w/o path-part, indicating a bias from the trained models. Thus, it is necessary to use an exhaustive dataset for training to increase robustness.

In addition, periodic model updates are important to catch up with new phishing URL trends. We found that (1) the URLs with newly appeared TLDs tended to be misclassified because such new TLDs were not included in NLP-feature D17 in Section 4.2.1, and (2) approximately one-third of misclassified URLs had random characters in the path part, which showed the importance of measuring the randomness of URLs.

7. Conclusion

This study extends our previous studies [31], [32] by adding dataset-independent NLP features, followed by implementing various models in conjunction with BiLSTM, LSTM, and DNN to compare the phishing detection performance with state-of-the-art works on five well-known comprehensive datasets.

We constructed two ensemble hybrid models (i.e., AU-H-DNN/LSTM and AU-H-DNN/BiLSTM) to evaluate the five datasets. Then, we compared our evaluation results with 15 baselines regarding accuracy, precision, recall, and F1-measure. Our experimental evaluation shows that our proposed hybrid models outperform the state-of-the-art baselines in accuracy and F1-measure, achieving an impressive range of 95.30% to 99.75% in

© 2024 Information Processing Society of Japan

F1-measure across each dataset. Importantly, our two hybrid-based models exhibit comparable performance with no statistically significant differences. Hence, we can confidently assert that our two hybrid-based models demonstrate robustness in detecting phishing URLs across all the datasets employed in this study.

Acknowledgments A part of this work was funded by LY Corporation, Japan.

References

[1] Cao, Y., Han, W. and Le, Y.: Anti-phishing based on automated individual whitelist, *Proc. 4th ACM workshop on Digital Identity Management*, pp.51–60, ACM, DOI: 10.1145/1456424.1456434 (2008).

[2] Kang, J. and Lee, D.: Advanced white list approach for preventing access to phishing sites, *Proc. International Conference on Convergence Information Technology*, pp.491–496, IEEE, DOI: 10.1109/ICCIT.2007.50 (2007).

[3] Prakash, P., Kumar, M., Kompella, R.R. and Gupta, M.: Phishnet: predictive blacklisting to detect phishing attacks, *Proc. IEEE INFOCOM*, pp.1–5, IEEE, DOI: 10.1109/INFCOM.2010.5462216 (2010).

[4] Sharifi, M. and Siadati, S.H.: A phishing sites blacklist generator, *Proc. IEEE/ACS International Conference on Computer Systems and Applications*, pp.840–843, DOI: 10.1109/AICCSA.2008.4493625 (2008).

[5] Aburrous, M. and Khelifi, A.: Phishing detection plug-in toolbar using intelligent fuzzy-classification mining techniques, *International Journal of Soft Computing and Software Engineering*, Vol.3, No.3, pp.54–61, Advance Academic, DOI: 10.7321/jscse.v3.n3.9 (2013).

[6] Ramanathan, V. and Wechsler, H.: Phishing website detection using latent dirichlet allocation and adaboost, *Proc. IEEE International Conference on Intelligence and Security Informatics*, pp.102–107, IEEE, DOI: 10.1109/ISI.2012.6284100 (2012).

[7] Xiang, G., Hong, J., Rose, C.P. and Cranor, L.: Cantina+: A feature-rich machine learning framework for detecting phishing web sites, *ACM Trans. Information and System Security*, Vol.14, No.2, Article.21, p.28, ACM, DOI: 10.1145/2019599.2019606 (2011).

[8] Xiang, G. and Hong, J.I.: A hybrid phish detection approach by identity discovery and keywords retrieval, *Proc. 18th International Conference on World Wide Web*, pp.571–580, ACM, DOI: 10.1145/1526709.1526786 (2009).

[9] Zhang, Y., Hong, J.I. and Cranor, L.F.: Cantina: A content-based approach to detecting phishing web sites, *Proc. 16th International Conference on World Wide Web*, pp.639–648, ACM, DOI: 10.1145/1242572.1242659 (2007).

[10] Maurer, M.E. and Herzner, D.: Using visual website similarity for phishing detection and reporting, *CHI Extended Abstracts on Human Factors in Computing Systems*, pp.1625–1630, ACM, DOI: 10.1145/2212776.2223683 (2012).

[11] Zhang, H., Liu, G., Chow, T.W.S. and Liu, W.: Textual and visual content-based anti-phishing: A bayesian approach, *IEEE Trans. Neural Networks*, Vol.22, No.10, pp.1532–1546, IEEE, DOI: 10.1109/TNN.2011.2161999 (2011).

[12] Liu, G., Qiu, B. and Wenyin, L.: Automatic detection of phishing target from phishing webpage, *Proc. 20th International Conference on Pattern Recognition*, pp.4153–4156, IEEE, DOI: 10.1109/ICPR.2010.1010 (2010).

[13] Chen, T.C., Dick, S. and Miller, J., Detecting visually similar web pages: application to phishing detection, ACM, *Trans. Internet Technology*, Vol.10, No.2, Article.5, p.38, ACM, DOI: 10.1145/1754393.1754394 (2010).

[14] Sanglerdsinlapachai, N. and Rungsawang, A.: Using domain top-page similarity feature in machine learning-based web phishing detection, *Proc. 3rd International Conference on Knowledge Discovery and Data Mining*, pp.187–190, IEEE, DOI: 10.1109/WKDD.2010.108 (2010).

[15] Hara, M., Yamada, A. and Miyake, Y.: Visual similarity-based phishing detection without victim site information, *Proc. IEEE Symposium on Computational Intelligence in Cyber Security*, pp.30–36, IEEE, DOI: 10.1109/CICYBS.2009.4925087 (2009).

[16] Afroz, S., Greenstadt, R.: Phishzoo: An automated web phishing detection approach based on profiling and fuzzy matching, *Proc. 5th IEEE International Conference on Semantic Computing*, pp.1–11, IEEE (2009).

[17] Chen, K.T., Chen, J.Y., Huang, C.R. and Chen, C.S.: Fighting phishing with discriminative keypoint features, *IEEE Internet Computing*, Vol.13, No.3, pp.56–63, IEEE, DOI: 10.1109/MIC.2009.59 (2009).

[18] Fu, A.Y., Wenyin, L. and Deng, X.: Detecting phishing web pages with visual similarity assessment based on Earth Mover's Distance (EMD), *IEEE Trans. Dependable and Secure Computing*, Vol.3, No.4, pp.301–311, IEEE, DOI: 10.1109/TDSC.2006.50 (2006).

[19] Wenyin, L., Huang, G., Xiaoyue, L., Min, Z. and Deng, X.: Detection of phishing webpages based on visual similarity, *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, pp.1060–1061, ACM, DOI: 10.1145/1062745.1062868 (2005).

[20] Fu, A.Y., Liu, W. and Deng, X.: EMD based visual similarity for detection of phishing webpages, *Proc. International Workshop on Web Document Analysis*, Vol.2005 (2005).

[21] Wenyin, L., Huang, G., Xiaoyue, L., Deng, X. and Min, Z.: Phishing web page detection, *Proc. 8th International Conference on Document Analysis and Recognition*, pp.560–564, IEEE, DOI: 10.1109/ICDAR.2005.190 (2005).

[22] Ozcan, A., Catal, C., Donmez, E. and Senturk, B.: A hybrid DNN-LSTM model for detecting phishing URLs, *Neural Computing and Applications*, Vol.35, No.7, pp.4957–4973, Springer, DOI: 10.1007/s00521-021-06401-z (2023).

[23] Maneriker, P., Stokes, J.W., Lazo, E.G., Carutasu, D., Tajaddodianfar, F. and Gururajan, A.: URLTran: Improving phishing URL detection using Transformers, *Proc. IEEE Military Communications Conference*, pp.197–204, IEEE, DOI: 10.1109/MILCOM52596.2021.9653028 (2021).

[24] Le, H., Pham, Q., Sahoo, D. and Hoi, S.C.: URLNet: Learning a URL representation with deep learning for malicious URL detection, arXiv preprint, arXiv: 1802.03162 (2018).

[25] Lin, M.S., Chiu, C.Y., Lee, Y.J. and Pao, H.K.: Malicious URL filtering—A big data application, *Proc. IEEE International Conference on Big Data*, pp.589–596, IEEE, DOI: 10.1109/BigData.2013.6691627 (2013).

[26] Le, A., Markopoulou, A. and Faloutsos, M.: Phishdef: URL names say it all, *Proc. IEEE INFOCOM*, pp.191–195, IEEE, DOI: 10.1109/INFCOM.2011.5934995 (2011).

[27] Ratcliff, J.W. and Metzener, D.: Pattern matching: The gestalt approach, *Dr. Dobb's Journal*, Vol.13, No.7, p.46 (1988).

[28] Sahingoz, O.K., Buber, E., Demir, O. and Diri, B.: Machine learning based phishing detection from URLs, *Expert Systems with Applications*, Vol.117, pp.345–357, Elsevier, DOI: 10.1016/j.eswa.2018.09.029 (2019).

[29] Rao, R.S., Vaishnavi, T. and Pais, A.R.: CatchPhish: detection of phishing websites by inspecting URLs, *Journal of Ambient Intelligence and Human Computing*, Vol.11, pp.813–825, Springer, DOI: 10.1007/s12652-019-01311-4 (2020).

[30] A, T., John, A.: Phishing website detection using LGBM classifier with URL-based lexical features, *Proc. IEEE Silchar Subsection Conference*, pp.1–7, IEEE, DOI: 10.1109/SILCON55242.2022.10028793 (2022).

[31] Aung, E.S. and Yamana, H.: Hybrid phishing URL detection using segmented word embedding, *Information Integration and Web Intelligence*, Vol.13635, pp.507–518, Springer, DOI: 10.1007/978-3-031-21047-1_46 (2022).

[32] Aung, E.S. and Yamana, H.: Segmentation-based phishing URL detection, *Proc. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp.550–556, ACM, DOI: 10.1145/3486622.3493983 (2021).

[33] Jalil, S. and Usman, M.: Highly accurate phishing URL detection based on machine learning, *Journal of Ambient Intelligence and Human Computing*, Vol.14, No.7, pp.9233–9251, Springer, DOI: 10.1007/s12652-022-04426-3 (2023).

[34] Marchal, S., François, J., State, R. and Engel, T.: PhishStorm: detecting phishing with streaming analytics, *IEEE Trans. Network and Service Management*, Vol.11, No.4, pp.458–471, IEEE, DOI: 10.1109/TNSM.2014.2377295 (2014).

[35] Gupta, B.B., Yadav, K., Razzak, I., Psannis, K., Castiglione, A. and Chang, X.: A novel approach for phishing URL detection using lexical based machine learning in a real-time environment, *Computer Communications*, Vol.175, pp.47–57, Elsevier, DOI: 10.1016/j.comcom.2021.04.023 (2021).

[36] Legitimate and Phishing URLs collected from University of Canada Brunswick (2016) (online), available from ⟨https://www.unb.ca/cic/datasets/url2016.html⟩ (accessed 2020-05-10).

[37] Kira, K. and Rendell, L.A.: A Practical Approach to Feature Selection, Sleeman, D., Edwards, P. (Eds.), *Machine Learning Proc. 1992*, pp.249–256, Morgan Kaufmann, DOI: 10.1016/B978-1-55860-247-2.50037-1 (1992).

[38] Wang, Y., Zhu, W., Xu, H., Qin, Z., Ren, K. and Ma, W.: A large-scale pretrained deep model for phishing URL detection, *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp.1–5, IEEE, DOI: 10.1109/ICASSP49357.2023.10095719 (2023).

© 2024 Information Processing Society of Japan

[39] Jiang, H., He, P., Chen, W., Liu, X., Gao, J. and Zhao, T.: SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization, *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, pp.2177–2190, ACL, DOI: 10.18653/v1/2020.acl-main.197 (2020).
[40] Jishnu, K.S. and Arthi, B.: Enhanced phishing URL detection using leveraging BERT with additional URL feature extraction, *Proc. 5th International Conference on Inventive Research in Computing Applications*, pp.1745–1750, IEEE, DOI: 10.1109/ICIRCA57980.2023.10220647 (2023).
[41] Tan, C.L., Chiew, K.L. and Sze, S.N.: Phishing webpage detection using weighted URL tokens for identity keywords retrieval, *Proc. 9th International Conference on Robotic, Vision, Signal Processing and Power Applications*, Vol.398, pp.133–139, Springer, DOI: 10.1007/978-981-10-1721-6_15 (2017).
[42] Alani, M.M. and Tawfik, H.: PhishNot: A cloud-based machine-learning approach to phishing URL detection, *Computer Networks*, Vol.218, p.109407, Elsevier, DOI: 10.1016/j.comnet.2022.109407 (2022).
[43] Howser, G.: Domain Name Service, *Computer networks and the Internet: A Hands-On Approach*, pp.333–374, Springer, DOI: 10.1007/978-3-030-34496-2_20 (2020).
[44] Aung, E.S. and Yamana, H.: 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, ACM, DOI: 10.1145/3366030.3366064 (2019).



Eint Sandi Aung is a Ph.D. candidate specializing in Computer Science and Communications Engineering at Waseda University. Her research focuses on cybersecurity, with a particular interest in phishing detection. Her expertise lies in specialized area such as phishing URL detection and the extraction of various features to enhance detection algorithms and strategies.



Hayato Yamana has been a professor at the faculty of science and engineering, Waseda University, since 2005. He is a member of DBSJ, IPSJ, IEICE, IEEE, and ACM. He is a fellow of IPSJ and IEICE.

© 2024 Information Processing Society of Japan