Proceedings of the SMART–2022, IEEE Conference ID: 55829
11th International Conference on System Modeling & Advancement in Research Trends, 16th–17th, December, 2022
College of Computing Sciences & Information Technology, Teerthanker Mahaveer University, Moradabad, India

2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART) | 978-1-6654-8734-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/SMART55829.2022.10046882

Malicious Domain Detection using NLP Methods – A Review

Pradeepa G.[1] and Dr. Devi R.[2]
[1]Research Scholar, Department of Computer Applications
Vels Institute of Science, Technology & Advanced Studies, Chennai, India
[2]Associate Professor, Department of Computer Applications,
Vels Institute of Science, Technology & Advanced Studies, Chennai, India
Email: [1]pradeepa25.ganesan@gmail.com, [2]devi.scs@velsuniv.ac.in

***Abstract*—Cyber security has emerged as a serious problem in today's business world. Attackers use different techniques to deceive the internet user to perform illegal activities such as stealing sensitive information, injecting malware, using the victim's system for further attack, etc. The majority of attacks are disseminated via social media posts, emails, and SMS messages. The message or webpage has embedded malicious links that will lead the user to the attacker's nest. Consequently, the issue of identifying malicious domain names has sparked significant concern. Researchers employ a variety of methods to find such malicious pages, including black or white lists, rules, machine or deep learning, visual similarity, and language processing (NLP). the attacker's employment of various tricks to confuse internet users, detecting or blocking an attack through a malicious domain is a difficult task. NLP with Machine Learning and NLP with Deep Learning methodologies receive the most interest from researchers due to their processing power. This paper provides an overview of several NLP text processing techniques as well as recent domain-related research.**

Keywords: NLP, Machine Learning, Deep Learning, Malicious URL, Phishing URL

I. Introduction

Attacks on the internet are growing more frequent every day. According to recent cyber threat security studies, 80.7% of systems would experience at least one system penetration by 2020 [1]. Cybercriminals trick victims, gaining their credentials or sensitive information through specially crafted URLs known as malicious URLs, and conducting malicious acts on the victim's side. Forms of attacks include phishing, spamming, drive-by downloads, and other similar methods. To resist such attacks, one must be aware of them and use the right tools to detect the presence of malicious URLs. Blacklist-based techniques are the most prevalent method for detecting malicious URLs [2].

A simple and low-false-positive-rate technique for detecting dangerous URLs based on a big list of URLs that have already been blacklisted as malicious. Another is the heuristic technique, which creates generic rules for identifying newly formed dangerous URLs and extracts features from malicious URL lists. The third method is based on machine learning or deep learning and extracts a significant number of features from a list of malicious URLs in order to train a prediction model for the differentiation between malicious and benign URLs [3]. The final one is the hybrid strategy, which combines many methods to offer a more rigid result [4]. In comparison to other approaches, NLP with Machine Learning and NLP with Deep Learning receive the most attention from researchers due to their processing capabilities. NLP stands for "natural language processing." This type of processing looks at the text on a character-by-character and word-by-word basis. This helps researchers look at the URL and content of websites to decide if they are malicious or benign.

Each section of the paper is as follows: Various methods in NLP which support text processing are presented in Section 2. Existing research works are discussed in section 3. Research challenges are presented in Section 4. The conclusion of the paper is presented in Section 5.

II. Natural Language Processing

A wide definition of natural language processing describes it as the automatic manipulation of natural languages, such as speech and text, by computer software [5]. Computers are only capable of understanding binary code; they cannot comprehend human languages like Hindi or English. Natural Language Processing enabled a computer to comprehend English and Hindi. A few more NLP preprocessing techniques help to convert them into numerical vectors.

Vectorization is the process of transforming a dataset's vocabulary words or tokens into a numerical vector. These vectors make up the data that machine learning (ML) and deep learning models use as input. The following are a few of the typical vectorization techniques:

A. Bag of Words (BoW) – Count Vectorizer

The bag-of-words model [6] models text utilizing machine learning methods. The BoW is a method of vectorization that uses numerical feature vectors to represent text. The BoW model keeps the count of words

concerning the document it occurred. Each vector acts as a feature column for the ML model as shown in Table 1.
Consider an example
Sentence 1: This food item is not good at all.
Sentence 2: This item is good.
Words in the sentences.

1. Food, 2. Item, 3. Good, 4. All, 5. Not

Table 1: Vector Representation

Sentence	Food	Item	Good	All	Not	Length
1	1	1	1	1	1	8
2	0	1	1	0	0	4

Problems:

- The vector length will increase if the new sentences contain new words.
- The vectors would have numerous 0s, creating a sparse matrix
- Both the word order and the grammatical structure of the sentences are being discarded.

B. N-gram

An N-Gram [7] comprises a collection of N words, phrases, or sentences that are all linked together. Large blocks of words or smaller groups of syllables could make up the N-Gram. N-gram models are created by counting word sequences in corpus text and estimating probabilities. This expands the vocabulary and helps the bag-of-words understand the document better. N-gram models can be employed at character or word level.
Consider an example
Sentence 1: This food item is not good at all.
Sentence 2: This item is good.
Bigram

Sentence 1:
This food,
food item,
item is,
is not,
not good,
good at,
at all

Problem:

- The N-gram model doesn't capture long-distance context well.

C. Term Frequency - Inverse Document Frequency (TF-IDF)

A statistical technique called term frequency-inverse document frequency evaluates the importance of a word within a group of documents. Multiplying the total number of times, a word appears in a document (TF) and the word's inverse document frequency (IDF) across a collection of texts produces this result [8]. This approach creates an association between words and sentences, but it only examines them from the standpoint of frequency.

Term Frequent (TF) is a way to measure how often a phrase or term (t) appears in a document(d).

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document} \quad \ldots\ldots(1)$$

The term "t" appears in the text "d" n times.
Consider an Example:
document 1: Good boy
document 2: Good Girl
document 3: Boy Girl Good.

Table 2 shows the Term Frequency of the above three documents.

Table 2: Term Frequency (TF)

TF			
Word	**document 1**	**document 2**	**document 3**
Good	1/2	1/2	1/3
Boy	1/2	0	1/3
Girl	0	1/2	1/3

IDF is the inverse of document frequency, a metric used to evaluate a term's information quality. For common words like stop words, IDF is low. IDF measures word importance.

$$idf_t = \log \frac{\text{number of documents}}{\text{number of documents with term } t} \quad \ldots\ldots(2)$$

Table 3 shows the IDF for the above three documents.

Table 3: Inverse Document Frequency

IDF	
Word	**IDF**
Good	Log (3/3)=0
Boy	Log (3/2)=0.176
Girl	Log (3/2)=0.176

Next, calculate the TF-IDF score for each word in the corpus as shown in table 4. Words with higher scores are more important:

$$(tf_{idf})_{t,d} = tf_{t,d} * idf_t \quad \ldots\ldots(3)$$

Table 4: TF-IDF

TF*IDF			
	Good	**Boy**	**Girl**
Sentence 1	1/2 * 0 =0	1/2* 0.176=0.088	0
Sentence 2	1/2 *0 =0	0	1/2*0.176 =0.088
Sentence 3	1/3 *0 =0	1/3 *0.176=0.058	1/3 *0.176=0.058

D. Word Embedding

Word embedding [9] is a type of word representation that bridges the gap between the human and machine understanding of language. They have acquired text representations in an n-dimensional space in which

words with the same meaning have similar representations. Word embedding is a technique for encoding words and texts using a dense vector representation as shown in Table 5. Popular word embedding algorithms include GloVe (Global Vectors), Word2Vec, and Keras Word Embedding. Consider an example

V=10000

Apply One Hot Encoding

Boy	2000
Girl	5000
King	6000
Queen	9000
Apple	1000
Mango	7000

Table 5: Feature Representation

	2000	5000	6000	9000	1000	7000
	Boy	Girl	King	Queen	Apple	Mango
Gender	-1	1	-0.92	0.93	0	0.01
Royal	0.01	0.02	0.95	0.96	0.02	0.01
Age	0.03	0.02	0.07	0.06	0.95	0.92
Food	0.01	0.02	0.04	0.05	0.98	0.96
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
300 dim						

III. Literature Review

Ghaleb et.al [6] proposed the CTI-MURLD model. This study uses datasets collected from the sources such as Kaggle, Phishtank, and ISCX-URL2016. Two distinct categories of URLs, malicious and benign, were identified and included in the dataset. Malicious URLs could be drive-by download links, spam, phishing, malicious links, etc. There are seven stages in CTI-MURLD. The random forest (RF) technique was used in the first six stages to build 3 ensemble-learning based predictors.

Different feature sets were utilized to train each RF-based predictor. Each RF classifier produced either a 0-malicious URL or a 1-benign URL, depending on the probabilistic model. The probabilistic results obtained from three different RF classifiers were utilized in the training process for the ANN classifier that was used to make the final decision. The N-gram approach was utilized to obtain features, and the TF-IDF method was utilized to represent those features. The two-stage classification of CTI-based features outperforms other detection algorithms.

Lakshmanarao *et al.* [10] proposed a machine learning-oriented approach to the problem of identifying malicious websites. An extensive Kaggle dataset with a lot of URLs was used to conduct the experiments. Three text feature extraction techniques (count vectorizer, hashing vectorizer-IDF vectorizer) and a phishing website detection model with four ML classifiers (LR, kNN, DT, RF) were used. The accuracy of the RF model and with the hash vectorizer was 97.5%.

According to Ebubekir *et al.* [11], a phishing detection system uses machine learning algorithms and natural language processing to identify this kind of attack. PhishTank and the Yandex Search API were utilized in order to compile a list of URLs. There are three modules in it for the detection procedure. The brand name or other important terms are first retrieved from the words to be studied, and then the Random Word Detection Module examines a word to determine whether or not it contains random characters. The words that were discovered are taken out of the list of words to be studied and saved in a random word list. Word Decomposer Module (WDM) has been used to attempt to break up words that are longer than 7 characters into their component words. In the event that the target term is composed of two or more words, WDM will separate the words. If the term does not contain two or more words, WDM will return the raw version of the word. The words that were shorter than seven characters and the words that were produced via the Word Decomposition Module were both assessed, and then those results were provided to the Maliciousness Analysis Module (MAM). One or more brand names or keywords could be among the words obtained by WDM. As a result, the brand name and keyword lists in this module are used to double-check the words that will be examined. After that, the degree of similarity between the analyzed word and the other terms on both the brand name and keyword lists is determined by conducting a word-by-word comparison. In order to determine degrees of similarity, the Levenshtein Distance algorithm is applied. The proposed system has been put through a number of tests, and the results revealed that the Random Forest algorithm performs quite well, with a success rate of 97.2%.

In order to detect phishing URLs, Eint et.al [12] used two different techniques of segmentation (Keras embedding and ELMo embedding). The experiment prepares a dataset for each of the two cases: imbalanced (D2) and balanced (D1) training, using data from DMOZ and PhishTank. In order to adapt in embedding, domain-level word and path-level character features are used. In the domain level word part, tokenization turns URLs into a list of words, since the most meaningful words and brand names are in the domain part.

Compared to the domain section, the characters in the path part occur more randomly and with less significance. Concatenated domain-level words and path-level characters are input into the Keras embedding layer, whereas domain-level words alone are input into the ELMo embedding layer.

The dropout layer, which has a value of 0.2, comes after the embedding layer. The system then employs LSTMs with a 0.2 recurrent dropout and densely applies them prior to the concatenating layer. Concatenation is followed by batch normalization, which is then densely repeated into a 128 output size, before the output layer. With the usage of mean evaluation metrics and 10 fold cross-validation, the proposed system has achieved approx. 93 percent and 97 percent in D1 and D2 datasets, respectively.

A lexical approach to classifying URLs was proposed by Quan *et al.* [13]. This technique makes use of machine learning algorithms that pattern the URLs. The proposed algorithm employs word vector representation and n-gram models for the words on the blacklist. By employing this method, classifiers will be better able to distinguish between benign and malicious URLs.

The data for the experiment was gathered from a variety of sources, including the DMOZ Project, the Malware Domain List, Malc0de, and CleanMX.

The results of the experiment demonstrate that SVM obtains a high level of accuracy of 97.1%.

Zhao *et al.* [14] propose a lexical analysis and feature quantification-based method for identifying malicious domain names. The approach consists of two phases for effective and precise detection. The initial phase compares a domain name seen with a blacklist of known malicious uniform resource locators. Based on how closely the observed domain name edits match the domain names on the blacklist, it is categorized as being either certainly malicious or potentially malicious. A suspected malicious domain name is further assessed in the second step using its reputation value, which indicates its lexical feature and is determined using an N-gram model. The N-gram approach is used to extract a set of whitelist substrings from the top 100,000 regular Alexa domain names. Each domain name, except the top-level domain, is divided into substrings of lengths 3, 4, 5, 6, and 7. According to their frequency in the whitelist substring set, the weighted values of the substrings are determined. By using the N-gram approach to segment a possibly hazardous domain name, its reputation value is determined based on the weighted values of its substrings. Finally, the reputation value of the prospective malicious domain name is used to assess if it is malicious or legitimate.

In order to identify algorithmically created domains, Jose *et al.* [15] introduced a machine learning method utilizing Random Forest that only considers lexical aspects of the domain names. Specifically, masked N-grams and other information were extracted from the domain name. Additionally, the dataset was created for experimentation and included regular and algorithmically produced domain names from various malware families (taken from Alexa & Bader repo extended). According to the results, masked N-grams offer detection accuracy that is comparable to that of other current approaches while also offering significantly superior performance.

TABLE 6: SUMMARY OF THE RELATED WORK

Authors	Dataset	NLP Methods	Rules/ ML / DL Model
Ghaleb *et al.* [6]	Kaggle, Phishtank ISCX-URL2016	N-gram technique	RF Algorithm, ANN Classifier.
Lakshmanarao *et al.* [10]	Kaggle	Count vectorizer, Hashing vectorizer-TF-IDF vectorizer,	Logistic Regression, K-NN, Decision Tree, Random Forest
Ebubekir *et al.* [11]	PhishTank, Yandex Search API	Word Decomposer	Random Forest
Eint *et al.* [12]	DMOZ, PhishTank	Domain-level word and path-level character features to adapt in embedding	Keras embedding, ELMo embedding, LSTM
Quan *et al.* [13]	DMOZ Project, Malware Domain List, Malc0de, CleanMX	word vector representation, n-gram models	SVM
Zhao *et al.* [14]	Alexa	N-gram	Reputation value based rules
Jose *et al.* [15]	Collected from Alexa & Bader repo extended	N-gram	Random Forest

IV. RESEARCH ISSUES

NLP decreases the effort of feature engineering in classic ML-based models by processing URL strings and web page content. But it also faces some difficulties

- Based on the context the meaning of a word, may vary.
- Most of the models are processing text in English only. Processing specific language may also possess challenges.
- Processing overhead may incur when the word vector size is more.

Contextual processing of the content of the webpage may improve the classification of URLs.

V. CONCLUSION

A malicious website or link poses a significant threat to the digital world. Attackers can effortlessly play with their victims and take their private data. It takes vigilance and a methodical approach to identify and block malicious Web pages in order to stop these kinds of attacks. This paper discusses the significance of detecting malicious links or pages and the support of NLP-based detection techniques. The most recent research in this field is also presented. Finally, incorporating language processing research difficulties to identify harmful pages or links may aid future research.

References

[1] Report Defense Cyberthreat 2020. https://cyber-edge.com/wp-content/uploads/2020/03/CyberEdge-2020-CDR-Report-v1.0.pdf

[2] Apoorva Joshi, Levi Lloyd, Paul Westin, Srini Seethapathy. Using Lexical Features for Malicious URL Detection - A Machine Learning Approach,ArXiv, 2019.

[3] Patgiri R., Katari H., Kumar R., Sharma D, Empirical Study on Malicious URL Detection Using Machine Learning. International Conference on Distributed Computing and Internet Technology (ICDCIT), Lecture Notes in Computer Science, vol 11319. Springer, Cham 2019. https://doi.org/10.1007/978-3-030-05366-6_31.

[4] Mamun M.S.I., Rathore M.A., Lashkari A.H., Stakhanova N., Ghorbani A.A. Detecting Malicious URLs Using Lexical Analysis. International Conference on Network and System Security. Lecture Notes in Computer Science, vol 9955. Springer 2016 https://doi.org/10.1007/978-3-319-46298-1_30.

[5] https://machinelearningmastery.com/natural-language-processing/

[6] Ghaleb, Alsaedi, Saeed, Ahmad, Alasli, M. Cyber Threat Intelligence-Based Malicious URL Detection Model Using Ensemble Learning. Sensors , 22, 3373, 2022.

[7] Shreyas, Nisha, N-Gram Assisted Youtube Spam Comment Detection, Procedia Computer Science, vol 132, pp.174-182, 2018.

[8] Chan, NtMalDetect: A Machine Learning Approach to Malware Detection Using Native API System Calls, arXiv:1802.05412v2 [cs. CR], 2018.

[9] Crișan, Florea, Halasz, Lemnaru, Oprisa, Detecting Malicious URLs Based on Machine Learning Algorithms and Word Embeddings, *IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP),* pp. 187-193, 2020.

[10] Lakshmanarao, Raja, Bala, Malicious URL Detection using NLP, Machine Learning and FLASK, International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), pp. 1-4, 2021.

[11] Ebubekir,Banu,Ozgur, NLP Based Phishing Attack Detection from URLs, AISC, pp. 608–618, Springer International Publishing AG, 2018.

[12] Eint, Hayato, Phishing URL Detection using Information-rich Domain and Path Features, DEIM, 2021.

[13] Quan, Seong, Detection of malicious URLs based on word vector representation and ngram, Journal of Intelligent & Fuzzy Systems, vol. 35, no. 6, pp. 5889-5900, 2018.

[14] Zhao, Chang, Wang, Zeng, Malicious Domain Names Detection Algorithm Based on Lexical Analysis and Feature Quantification, IEEE Access, vol. 7, pp. 128990-128999, 2019.

[15] Jose, Ricardo, Emilio, Detection of Algorithmically Generated Malicious Domain Names using Masked N- Grams, Expert Systems with Applications, vol. 124, pp. 156-1.