Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051598

ARTICLE

Detecting Malicious Uniform Resource Locators Using an Applied Intelligence Framework

Simona-Vasilica Oprea* **and Adela Bâra**

Department of Economic Informatics and Cybernetics, Bucharest University of Economic Studies, Bucharest, 010572, Romania

*Corresponding Author: Simona-Vasilica Oprea. Email: simona.oprea@csie.ase.ro

Received: 10 March 2024 Accepted: 17 May 2024 Published: 20 June 2024

ABSTRACT

The potential of text analytics is revealed by Machine Learning (ML) and Natural Language Processing (NLP) techniques. In this paper, we propose an NLP framework that is applied to multiple datasets to detect malicious Uniform Resource Locators (URLs). Three categories of features, both ML and Deep Learning (DL) algorithms and a ranking schema are included in the proposed framework. We apply frequency and prediction-based embeddings, such as hash vectorizer, Term Frequency-Inverse Dense Frequency (TF-IDF) and predictors, word to vector-word2vec (continuous bag of words, skip-gram) from Google, to extract features from text. Further, we apply more state-of-the-art methods to create vectorized features, such as GloVe. Additionally, feature engineering that is specific to URL structure is deployed to detect scams and other threats. For framework assessment, four ranking indicators are weighted: computational time and performance as accuracy, F1 score and type error II. For the computational time, we propose a new metric-Feature Building Time (FBT) as the cutting-edge feature builders (like doc2vec or GloVe) require more time. By applying the proposed assessment step, the skip-gram algorithm of word2vec surpasses other feature builders in performance. Additionally, eXtreme Gradient Boost (XGB) outperforms other classifiers. With this setup, we attain an accuracy of 99.5% and an F1 score of 0.99.

KEYWORDS

Detecting malicious URL; classifiers; text to feature; deep learning; ranking algorithms; feature building time

1 Introduction

In this section, we focus on the role of Natural Language Processing (NLP) and various techniques applied for detecting malicious URL, existing practice in the field, motivation and contribution of current research.

1.1 The Role of NLP and Various Techniques for Detecting Malicious URL

Over recent decades, the majority of data has been unstructured, predominantly appearing in formats such as text, images, and videos [1]. This data can be categorized as big data due to its considerable volume, rapid update frequency, and clear diversity. Sensors and humans generate large amounts of data as they tweet, research, consume, visit websites, do online shopping, socialize, etc. Text data is analysed together with images and emoticons to understand whether the context is straightforward, or sarcasm is also present because such combination might reveal interesting findings. Usually, the text generated by humans includes considerable noise. Therefore, pre-processing text implies cleaning and transforming the initial text into a meaningful text that can be further processed with Machine Learning (ML) algorithms [2,3].

In this context, the NLP techniques become useful. It may uniform the text from the case point of view as text processing is sensitive to lower/upper text, considering "text", "tExt" and "TEXT" different although they have the same meaning. Eliminating of stopwords, punctuation signs, spelling, standardization of acronyms, normalization of text by stemming and lemmatization considering different forms of the same word root are just a few techniques to pre-process the text. The presence of stopwords across the corpus may negatively influence the output. Several libraries in Python, for instance, perform lemmatization, but they have to be tested as their performance significantly differs. Processing text has many applications as now as typing the writer may receive word suggestions. *Text classification* is one common application to detect email spam, provide recommendations, prioritize tasks, etc. Text-to-feature (t2f) transformation implies encoding text into numerical features that are further processed by the ML algorithms [4,5]. One-hot encoder, count vectorizer, Term Frequency–Inverse Document Frequency (TF-IDF), and word embedding to retain the context in which the word occurs are just a few techniques that allow the t2f transformation [6,7]. Another interesting application of text processing is *text summarization* that reveals the meaning of a long text in a brief report. *Text generation* is frequently applied to assist customers who visit websites and use the chat provided by the website to learn more about certain products. Text generation is also used to interact with users online and collect data, documents, etc. interactively. *Text regression* can provide a prediction of the time necessary to solve the damage in case of a car accident, for instance, by analysing the text inserted for a certain car event. Furthermore, it can predict illegal transactions, and churn rates, or it may analyse political speeches, transform sound into text and estimate their impact on the stock markets. *Text similarity* is used to identify plagiarism, provide content-based recommendations, suggest complementary products or alternatives [8], etc. A relevant text analysis may select the CVs that match a job description or provide a selection of journals that match the manuscript topic. Moreover, Latent Dirichlet Allocation (LDA) is a useful NLP technique applied to multiple texts in order to identify the main topics that are preponderant [9–11]. For instance, abstracts or the entire articles' text that implies NLP, ML and Artificial Intelligence (AI) can be pre-processed and then LDA applied to examine the topics approached by researchers over decades aiming to identify potential gaps in the literature. Furthermore, the impact of news media on cryptocurrencies volatility using LDA is analysed [12].

The process of malicious URL detection involves the identification and prevention of access to URLs leading to web pages linked with harmful activities like phishing, the dissemination of malware or various cyber threats. According to the most recent data from the Federal Trade Commission (FTC), in 2022, American consumers' losses due to fraud reached $8.8 billion, marking a 30 percent increase compared to 2021. The largest portion of these losses, amounting to over $3.8 billion, was attributed to investment scams, including cryptocurrency schemes, which was twice the figure reported in 2021. The FTC discovered that individuals in the 20–29 age group reported more frequent monetary losses compared to those in the 70–79 age group. However, when older adults did experience financial losses, the amounts involved were typically higher. This is attributed to the fact that many retirees possess valuable assets such as savings, pensions, life insurance policies or property, which make them attractive targets for scammers. According to the FTC, scam texts alone led to consumer losses exceeding $326 million in 2022. There is a large variety of frauds that may affect us when clicking on a URL link [13,14]. The organizations that are behind the malicious links make them look almost similar to genuine ones. They often use the identity of benign sites and scam people. Usually, the URL contains a small change that is not easily detected by users (obfuscation) [15,16]. Therefore, more efforts have to be put into identifying such issues that ultimately deceive people and take advantage of their vulnerabilities [17,18].

The most common techniques and methods for detecting malicious URLs are signature-based detection, heuristic analysis, reputation-based detection, behaviour-based detection, ML and AI, sandboxing, real-time URL scanning, URL blacklists, user education, etc. [19], emphasizing a feature-building approach [20] and feature selection [21]. Signature-based detection entails the upkeep of a database containing recognized malicious URLs and utilizing pattern matching to detect URLs that exhibit these established patterns. Signature-based detection is a common method employed by antivirus and security software to restrict access to harmful websites [22,23]. Heuristic analysis implies the examination of a URL for different suspicious attributes, such as unusual or concealed characters, lengthy sequences of random characters or recognized patterns commonly linked with phishing attempts [24]. Reputation-based detection relies on the credibility of a URL or its hosting domain. It evaluates whether a website or domain has a track record of being linked to malicious activities. Reputation databases are consistently refreshed with data concerning websites [25,26]. Behaviour-based detection examines the actions and conduct of a website or URL. This involves scrutinizing the website's content, the interactions it initiates with the user's system or the identification of established malicious scripts [27,28]. The detection of malicious URLs is progressively making use of ML and AI algorithms and methods [29,30]. These systems analyse extensive datasets to pinpoint patterns and irregularities in URLs and their activities. Sandboxing entails running a URL or website within a controlled and isolated environment to monitor its behaviour [31]. If the website displays malicious behaviour, it can be prevented from running. Usually, web security solutions offer real-time URL scanning and filtering, which means they assess and classify URLs in real time as users attempt to access them, thereby preventing access to harmful websites [32]. Maintaining a blacklist of recognized malicious URLs is also a common practice. When a user tries to access a URL on the blacklist, their access is denied [33,34]. Nevertheless, educating users about the risks associated with clicking on suspicious links and how to identify malicious URLs is an important component of any security strategy [35]. It is known that malicious actors are constantly adapting their strategies, making URL detection an ongoing challenge. Consequently, a combination of the techniques and regular updates to security systems are often required to effectively identify and block malicious URLs, thereby safeguarding against cyber threats.

1.2 Motivation and Contribution of Current Research

In this paper, we propose a framework using multiple datasets and testing three main categories of features: frequency, prediction and structure-based embeddings. Both ML and DL algorithms are applied to predict the nature of the URL, evaluating and ranking the results based on several performance criteria. In the pursuit of this research, our ***motivation*** stems from the significance of the task, driven by the fact that detecting malicious URLs holds great importance for a multitude of reasons. The main objective is to safeguard users from the perils of cyber threats. Malicious URLs expose individuals to a range of online risks, encompassing malware infiltrations, phishing assaults, identity theft, and monetary harm [36]. The identification of these URLs plays a pivotal role in averting users from succumbing to these dangers. Moreover, malicious URLs have the potential to be exploited for the theft of sensitive information. Detecting these URLs is instrumental in protecting personal and confidential data from being compromised. Additionally, malicious URLs take advantage of weaknesses in a user's device or network. Identifying them is crucial to forestall security breaches and ensuring the protection of the integrity of computer systems and networks [37].

Furthermore, organizations have a pressing need to shield their networks and systems from malicious URLs to avert data breaches, financial losses and reputational damage. Malicious URLs are frequently utilized for the dissemination of malware, which may result in catastrophic consequences. Identifying these URLs is paramount in preventing the infiltration of malware into devices and networks. Usually, malicious URLs are linked to phishing attacks, designed to deceive individuals into divulging personal information. Additionally, malicious URLs, including QR codes, lead to financial fraud, such as online scams, thus, identifying them is imperative to protect users from fraudulent activities [38].

Given that users rely extensively on the Internet for various tasks (shopping, banking, booking, payments, trading, etc.), maintaining trust is a cornerstone of the online experience. The detection and blocking of malicious URLs are pivotal in upholding trust in online services and platforms [39]. However, the malicious actors are continually adjusting their strategies and generating new malicious URLs. Detecting these evolving threats necessitates perpetual vigilance and ongoing technological enhancements. Thus, the identification of malicious URLs is significant for safeguarding individuals, institutions, and society from a wide spectrum of online threats [40,41].

In the modern digital era, Android has risen to become the most widely used mobile operating system globally, powering billions of smartphones and various connected devices. The open and adaptable nature of the Android ecosystem has not only fostered innovation and user empowerment but has also, unintentionally, made it susceptible to a continuous influx of cyber threats. Among these threats, Android malware stands out as a pervasive and ever-evolving menace. In 2022, according to the SECURELIST by Kaspersky, there was a significant upsurge in mobile malware, with over 5.7 million blocks[1]. This statistic highlights the exponential growth in mobile malware incidents in recent times. Android malware encompasses a wide range of malicious software specifically crafted to exploit vulnerabilities, pilfer sensitive information, disrupt normal operations and, in some cases, extort users. Often concealed within seemingly harmless applications, these malicious programs pose a substantial risk to personal privacy, data security and the overall integrity of the Android ecosystem. Given this looming threat, the necessity to detect and categorize malware has never been more crucial.

Our contribution consists of proposing a framework for detecting malicious URL that includes four stages, such as generalization by using multiple datasets, feature options, algorithms options and assessment. In the proposed malicious URL detection framework, we envision an exploratory data analysis stage, 3 feature builders that rely on NLP techniques that run in parallel, multiple classifiers implemented on various datasets to ensure generalization of the results and an evaluation stage. The proposed framework uniquely combines the data pre-processing stage using NLP techniques and processing stages using ML algorithms as well as a more complex evaluation that takes into account performance metrics of various classifiers as well as the time required to build features that are also significant in the case of real-time detection tools. The results are optimized by choosing the best solution that maximizes the F1 score and minimizes the computational time and other metrics of the confusion matrix such as false positive rate or type error II.

As a novelty, we propose a pipeline to investigate URLs and additionally propose an evaluation step in which the results are weighted using the computational time, confusion matrix and other metrics obtained from the classification of the URLs. Furthermore, we suggest incorporating a new metric,

[1] https://securelist.com/it-threat-evolution-q2-2023-mobile-statistics/110427/

called Feature Building Time (FBT), to evaluate the performance of a malicious URL detection tool. This metric takes into account not only the computational time required for running the ML algorithm, but also the computational time needed for building or extracting the features, which can vary significantly depending on the technique.

The remainderof the paper is structured in several sections. The most recent and relevant related works are briefly presented in Section 2, whereas in Section 3, the URL detection framework is proposed offering a flowchart that describes the pipeline. The results of implementing various feature builders and ML algorithms are showcased in Section 4, the conclusion and interesting insights in Section 5.

2 Literature Review

In this section, we review several research papers and approaches related to the detection of malicious URLs. This section consists of two subsections: (1) URL structured-based approach in which the structure of the URL is used to create features and (2) Advanced NLP techniques and ML algorithms-based approach in which various techniques of vectorization are implemented to convert text to numeric vectors that can be handled by ML and DL algorithms.

2.1 URL Structured-Based Approach

Malicious URL detection based on ML concerns numerous researchers and entrepreneurs. Both companies and freelancers provide online tools to detect malicious websites[2]. They scan the URL that is inserted by the user and may identify potential issues. For instance, URLhaus project shares malware URLs[3] to users to block them. Fake Sites List is a project that offers more than 160,000 of URL[4] of active problematic websites. Moreover, numerous benign URLs are provided by Alexa, Umbrella, Majestic and Farsight[5]. The websites' security-related communities maintain blacklists of malicious URLs based on denouncements and detection tools. However, they are sometimes overpassed because new and numerous scams appear constantly. Even if blacklists do exist, they are seldom checked when shopping, getting a prize or doing Internet searches. Human nature is enthusiastic when gaining a prize even if a lottery ticket has not been bought. Getting a rapid benefit out of a marvelous efficient investment scheme is always appealing. Usually, people that refrain from clicking on such a link did not click it using their intuition and rarely after checking the blacklists.

Furthermore, the detection algorithms must constantly improve as the scammers always improve their methods to deceive people as those who commit fraud are many times one step ahead of those who fight fraud. The interaction in social networks is largely based on texts that indicate NLP techniques to handle and pre-process large volumes of data as text. For instance, a Markov model was proposed in [42] to estimate the level of trust in social networks and how text can influence users by the level of trust. Several research studies proposed methods to identify malicious URLs using ML and DL algorithms. In [43], the authors combined big data and ML to enhance the ability to detect malicious URLs and abnormal behavior. They extracted features and behaviors, using ML and big data technologies and created a tool for detecting the problematic URLs. The proposed URL attributes and behavior improved the ability to detect malicious URL significantly. The researchers considered that even the Alexa's ranked websites as benign, there might be fraudulent URLs (or

[2] https://zeltser.com/lookup-malicious-websites/
[3] https://urlhaus.abuse.ch/api/
[4] https://db.aa419.org/fakebankslist.php?start=1
[5] https://tranco-list.eu/list/7PG6X/1000000

defacement) [44]. They provided a lexical analysis for the detection of bad-labeled URLs. Around 110,000 URLs were investigated, and the obfuscation techniques were analyzed.

A lexical analysis and an investigation of the network activity were also proposed [45]. The authors used K-Nearest Neighbors (KNN), Random Forest (RF) and a type of Decision Tree (DT) algorithm (C4.5) to predict four categories out of the 700,000 URLs: benign, defacement, malware and phishing. They built 78 lexical features, selected 15 and used one of the obfuscation methods to detect malicious URLs. Additionally, in [46], the authors proposed the extraction of lexical features. They also argued that blacklists are not often consulted, or they are a step behind the fraudsters. Using the lexical features and classification algorithms, their model named FireEye Advanced URL Detection Engine (FAUDE) provided good results in detecting malicious URLs. The lexical, content and network-based features were created to detect malicious URLs (spam, spyware, phishing and malware) combined with ML and DL models [47], using autoencoders to create features [48]. The dataset consisted of 66,506 URLs. Feature selection was applied to extract the most relevant features. A simple algorithm such as Naïve Bayes (NB) proved to find the best results, predicting the URLs with a precision of 96%. A malicious and benign dataset of URLs was built. Moreover, a dataset of around 1.5 million URLs was collected using a web crawler. It contained attributes and webpage source elements like JavaScript code as unstructured data. Both supervised and unsupervised tools were employed to classify the URLs. The labels were added using a Google service that allows safe browsing[6]. Furthermore, Google provides a database of hashed URLs within an API, and the user can check the nature of the URL by using the hash (SHA-256) of a particular URL. DL algorithms were applied to classify URLs [49].

Phishing URL detection, using ML methods, was proposed in [50]. Phishing involves stealing private data like passwords, identifiers, etc. when the users access the malicious website and take advantage of them. Usually, it has a considerable financial impact that may lead to losing the intellectual or copyright property and spoiling reputation. To identify whether a URL is malicious or not, several ML algorithms such as RF, DT, Light Gradient Boosting Machine (LGBM), Logistic Regression (LR) and Support Vector Machine (SVM) were applied. LGBM proved to be the best solution, but the researchers did not provide a clear methodology for phishing URL detection. They rather tested the ability to classify several ML algorithms using a small dataset (3,000 samples) with perfectly balanced target values. The importance of input features to detect problematic URLs was depicted in [51]. During the COVID-19 pandemic, when the people were in lockdowns, the threats intensified, and the classifiers were challenged due to the volume of data and the changing pattern of malicious URLs. The traditional features were not efficient and the correlation between them and the target showed a lower efficiency of their contribution to detecting problems. Linear and non-linear space transformation methods were proposed in [51]. Singular value decomposition and linear programming were proposed to find the optimal distance. The Nyström method was also used to include the non-linear space transformation method. The authors investigated a dataset with 331,622 URLs, extracting 62 features to validate the proposed methods for building features. Several ML algorithms such as KNN, SVM and Multi-Layer Perceptron (MLP) were trained, and they improved the detecting performance using the proposed feature engineering method. The results showed that the proposed methods considerably improved the efficiency and performance of certain classifiers, such as KNN, SVM and neural networks.

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

2.2 *Advanced NLP Techniques and ML Algorithms-Based Approach*

URL2vec is an interesting method to create features out of URLs. Both structural and lexical features were extracted from URLs and classified using a temporal convolutional network reaching a precision of 95.97% [52]. Moreover, the URLs were converted into documents or words similar to doc2vec or word2vec embeddings [53]. The authors mixed the character embeddings with the URLs components that were vectorized. The method was applied to one million-record dataset containing phishing indications. The accuracy achieved by this method was 99.69%.

To address the limitations of existing phishing blacklists, a more effective approach for identifying malicious URLs was introduced, which leverages DL through a variational autoencoder (VAE) as detailed in [54]. This framework involved the automatic extraction of intrinsic features from raw URLs using the VAE model, achieved by reconstructing the original input URLs, thus enhancing the detection of phishing URLs. For experimentation, approximately 100,000 URLs were gathered from two publicly accessible datasets: the ISCX-URL-2016 dataset and the Kaggle dataset. The experimental findings demonstrated that the proposed model achieved a remarkable peak accuracy of 97.45% and exhibited a faster response time of 1.9 s. These results outperformed other models tested in the study. Furthermore, an innovative approach called adaptive ensemble clustering with Boosting Broad Learning System (BLS)-based autoencoder (BoostAEC) was introduced [55]. Thorough experiments conducted on diverse real-world datasets showcased the exceptional performance of BoostAEC compared to contemporary ensemble clustering methods, establishing its superiority.

Moreover, a novel approach was presented in [56] for intrusion detection in the context of Industry 4.0, where the prevalence of big data and the Internet of Things (IoT) presented ongoing challenges. In this environment, cyber-physical systems play a vital role, and the task of anomaly detection remains both crucial and demanding. The researchers developed the one-class BLS (OCBLS) and extended it to the stacked OCBLS algorithms. Leveraging the efficiency of BLS, these approaches offered the advantage of streamlined training processes. Furthermore, the stacked OCBLS method allowed for the acquisition of high-level hidden features from network traffic data through a progressive encoding and decoding mechanism. To validate the efficacy of their proposed methods, extensive comparative experiments were conducted on various real-world intrusion detection tasks.

Similar frameworks were provided [57–61]. In [57], an analysis focused on evaluating adversarial attacks in a framework designed for detecting malicious advertisement URLs. Commonly used techniques involved extracting linguistic features from URL strings, such as bag-of-words, followed by the application of ML models. Traditional methods for detecting malicious URLs typically required effective manual feature engineering that handled previously unseen features and generalized well to test data. In this research, a set of lexical and web-scrapped features was extracted, and ML techniques were employed to detect fraudulent advertisement URLs. The mix of six distinct types of features effectively addressed the challenges posed by obfuscation in the classification of fraudulent URLs. The research utilized twelve datasets for various tasks, including detection, prediction and classification, and expanded the analysis to include mismatched and unlabeled datasets. The performance of four ML techniques-RF, GB, XGB and AdaBoost-was analyzed within the detection component of the framework. With their proposed approach, the researchers achieved an impressively low false negative rate of 0.0037, while maintaining a high detection accuracy of 99.63%. Additionally, they applied an unsupervised learning technique for data clustering, using the K-Means algorithm to facilitate visual analysis.

Another approach for the detection of malicious webpages, known as the heterogeneous ML ensemble framework was proposed [60]. The effectiveness of malicious URL detection was significantly influenced by the features present in the learning dataset. The overall performance of various ML models varied depending on the data features, making it less desirable to rely on a single model in any given environment. To overcome these limitations, the proposed approach utilized an ensemble strategy that combines different ML algorithms. This method surpassed the performance of single models by 6%, enabling the detection of an additional 141 malicious URLs. Furthermore, reference [60] automated repetitive tasks, which enhanced the performance of various ML models. Additionally, the framework included an advanced feature set derived from URL and web content, incorporating an optimized detection model structure. In reference [61], a 3-step framework for detecting malicious URLs was presented. Many studies have introduced methods and achieved accuracy, but there has been a lack of comprehensive research summarizing the field of malicious URL detection. Therefore, the researchers proposed a 3-step framework for detecting malicious URLs and provided an overview of 14 previous works that aligned with their framework. They found that nearly all research in the domain of malicious URL detection using ML could be categorized within the 3-step framework. The study involved the evaluation of various ML models and methods that involved context, assessing their suitability based on the 3-step framework. The results underscored the significance of the context in the detection process and highlighted that context-based embedding methods were particularly crucial. The accuracy of malicious URL detection was shown to improve with the incorporation of context-based methods.

The overall drawback of the previous methods consists in the fact that they mainly use only one dataset to train the ML algorithm and test their performance, because this approach is not able to generalize on other datasets and give confidence that the detection is accurate. Another downside is that they only assess the methods using several metrics (one or two metrics) without ranking the results. Out of the 15 briefly summarised researchers, only 5 were written as complex and reliable frameworks. Equally important is the Feature Building Time (FBT) as the newest methods (like GloVe or doc2vec) are time-consuming. Only a few works provided and compared methods to build features as most of them rely only on the URL structure-based extraction method. Therefore, in comparison with the previous studies, our contribution consists of extensive testing of each method and creating a ranking tool to assess various methods in which we include both the accuracy and the computational time. In the next section, the proposed NLP-based framework is described.

3 Detection Framework

In this section, we propose a URL detection framework, offering a flowchart that describes the pipeline designed for URL processing. We depict the URL structure and extraction of the features. Then, we propose to vectorize data, underlining on skip-gram algorithm and GloVe model, and providing several metrics to assess the results. In this framework, we train supervised classifiers to detect malicious URLs. Both ML and DL algorithms are trained.

3.1 Methodology

A robust application framework for investigating URLs is proposed in this paper. It consists of several steps that include multiple datasets to provide a detection solution that can generalize on other datasets. Initially, the data is explored searching for missing values or null. Then, the categories are graphically represented to check whether the categories are balanced or not. Three categories of features are deployed to classify URLs. For the first category, the frequency-based embeddings (features) approach consists of 5 main methods, out of which hash vectorizer and TF-IDF are applied as they are the most advanced and require less computational effort. In the second category, which focuses on prediction-based embeddings, models such as Google's word2vec, skip-gram, and CBOW are utilized [62,52]. Additionally, GloVe, which provides a global vector representation, is also applied [63,64].

The overall methodology and its flows are showcased in Fig. 1. To build numeric vectors out of text using word2vec and GloVe, the tokenization of the URLs is applied. Feature selection can be applied and tested to improve the results of the classification [65]. The third category is related to the specificity of the URL structure, extracting features from the components of the URL itself. In this case, tokenization is not necessary as the features are created and named by applying procedures and functions to extract different insights from the URLs structure. For instance, counting the digits or specific symbols such as "&" or "#", counting the total number of symbols or checking whether the *http* and *https* exist.

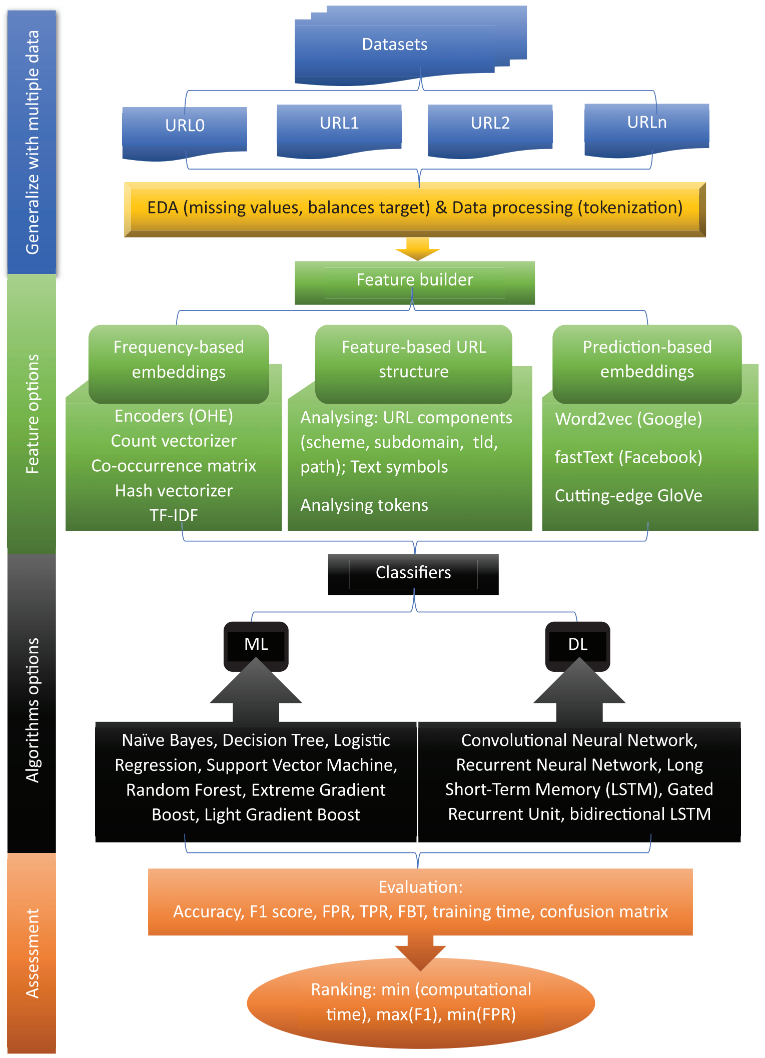
Generalize with multiple data
Feature options
Algorithms options
Assessment
Datasets
URL0
URL1
URL2
URLn
EDA (missing values, balances target) & Data processing (tokenization)
Feature builder
Frequency-based embeddings
Encoders (OHE)
Count vectorizer
Co-occurrence matrix
Hash vectorizer
TF-IDF
Feature-based URL structure
Analysing: URL components (scheme, subdomain, tld, path); Text symbols
Analysing tokens
Prediction-based embeddings
Word2vec (Google)
fastText (Facebook)
Cutting-edge GloVe
Classifiers
ML
DL
Naïve Bayes, Decision Tree, Logistic Regression, Support Vector Machine, Random Forest, Extreme Gradient Boost, Light Gradient Boost
Convolutional Neural Network, Recurrent Neural Network, Long Short-Term Memory (LSTM), Gated Recurrent Unit, bidirectional LSTM
Evaluation:
Accuracy, F1 score, FPR, TPR, FBT, training time, confusion matrix
Ranking: min (computational time), max(F1), min(FPR)

Figure 1: Methodology flowchart

Two types of supervised classifiers are trained to detect malicious URLs: ML and DL. The ML algorithms: Naïve Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and eXtreme Gradient Boost (XGB) are trained, as well as DL algorithms: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), bidirectional LSTM.

3.2 URL Structure Building the Structural Features

URLs for http (without ssl) or https (with ssl) have usually several components, such as: a) *schema* that indicates the protocol used to access the webpage as in Fig. 2. The schema is followed by a colon and two forward slashes; b) *host* name or the host for the webpage. Services are provided by a server as a hostname. Sometimes, the hostname is followed by a port number, but usually, it is omitted; c) path to the webpage or the resource on the Internet. Usually, it displays directories and starts with a single slash and contains dots, slashes and other symbols; d) parameters that are part of a query. They get a value and are concatenated using ampersands. The port number is optional, and it is placed after the hostname. When the query is specified, it starts with a question mark. Additionally, URL can be followed by a hash that can be an identifier of a subsection of a page.

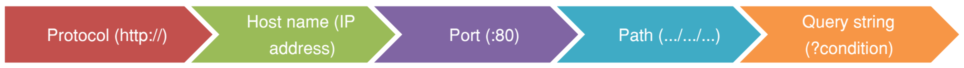

Figure 2: Graphical representation of the URL structure

The specific structure of a URL allows us to compute special features, such as total length of the URL, length of the hostname, length of the path, first directory length, top-level domain (tld) length, count special characters (including punctuation), count http, https, www, count digits, letters, number of directories, sum of the number of special characters, use of IP or not, use of shortening URL or not, etc.

3.3 Vectorized Features

There are numerous techniques to convert words into numbers and build features that allow the ML algorithms to classify texts. Converting text to features or feature engineering is one of the foundations of the NLP. There are two types of methods or techniques: 1) Methods based on frequency and hence called frequency-based embeddings or features (One Hot Encoding, Count vectorizer, N-grams, Co-occurrence matrix, Hash vectorizer, Term Frequency-Inverse Document Frequency (TF-IDF)); 2) Methods based on prediction-based embeddings, typically called word embeddings (word2vec–skip-gram and cbow from Google, fastText from Facebook).

One Hot Encoding (OHE) is the simplest text transformation into numeric values and converts characters or words into numbers (1, 0). However, the OHE does not take the frequency of the words into account. OHE can be replaced by a *count vectorizer* to avoid this drawback. The count vectorizer considers only the word itself counting its frequency and eluding the previous and the next words and missing a complete meaning to the words. Therefore, the count vectorizer can be combined with the *n-grams* technique to create numeric features and overcome this disadvantage. Additionally, the *co-occurrence matrix* is similar to the count vectorizer counting the occurrence of the words together, not individual words. However, the count vectorizer and co-occurrence matrix have a limitation: when the vocabulary becomes huge, it leads to memory and computational congestion.

Hash vectorizer solves the problem of memory being a memory-efficient technique. Hashing is used in cybersecurity to encrypt signatures, messages, etc. Instead of storing the tokens as strings, it hashes the tokens (with SHA256, for instance) encoding them as numerical indexes. Once vectorized, the features cannot be decoded that can be a significant downside. To obtain the hash value of a word i, a hash function f is applied, and the result or the hashed value is h_i.

$$w_i \rightarrow f(w_i) \rightarrow h_i. \tag{1}$$

Term Frequency-Inverse Document Frequency (TF-IDF) is a technique that shows the meaning of a number of words and avoid the drawbacks of the bag of words approach. TF is the ratio of the count of a word in a sentence to the length of the sentence, while IDF of each word is the log of the ratio of the total number of rows to the number of rows in a document that contains the word. In other words, the IDF is the logarithm of the ratio between the total number of documents and the number of documents containing term or word i. Therefore, IDF assesses the rareness of a term. TF-IDF is the product between TF and IDF, which makes predictions and information retrieval more relevant. The value of TF-IDF for a term i in a document j is:

$$TF - IDF\left(w_{i,j}\right) = tf_{i,j} \times log\frac{N}{df_i}, \tag{2}$$

where: $tf_{i,j}$–number of occurrences of i in j;

df_i–number of documents containing i;

N–total number of documents, where at least 1 document has to be processed.

Word2vec is a DL Google framework to train word embeddings. It provides the semantic links among words, the context and meaning of the words, or how frequently the words appear together. Word2vec was created for capturing the semantic relationship of the words, whereas the above-mentioned techniques fail to provide. It creates vectors that encapsulate the meaning of the words using several dimensions (such as negative/positive, concrete/abstract, etc.). Consequently, word2vec distinguishes between "glove" as a cloth and "GloVe" as a vectorizing technique. There are two algorithms provided by word2vec: skip-gram (sg) and a continuous bag of words (cbow). Cbow predicts the probability of a word based on the context or the surrounding words, while sg starts from a word that is also known as the target word and attempts to predict the context of that word. For a large vocabulary, it is better to use sg, but it is slower to train, while cbow is better for small vocabulary and is faster to train. The mathematical model underlying word2vec is based on neural networks and optimization techniques. Particularly, word2vec uses a shallow neural network architecture and the Maximum Likelihood Estimation (MLE) to compute word embeddings. Word2vec uses a *distributed* representation of a word in a text. Each word is a distribution of weights across the text. Therefore, one word contributes or is responsible for the representation of other words. By analyzing a large corpus, it learns relationships between words and generates numerical vectors.

By training the word2vec model on a large corpus of text, it learns to assign similar vector representations to words that have similar context or meaning. These embeddings can then be used to measure similarity between words, perform vector arithmetic operations or as input features for downstream NLP tasks. To summarize, the mathematical model of word2vec involves a neural network with softmax outputs and utilizes MLE as the training objective. The model parameters are optimized using optimization algorithms to learn word embeddings that capture semantic relationships between words. *fastText* is another DL framework developed by Facebook to capture context and meaning. It provides character-level embeddings.

3.3.1 Skip-Gram Algorithm

The vector for each word in the analyzed corpus is initialized randomly. In a document, each word that has a position t will be considered as the center (as in Fig. 3). At that point, the position t becomes c. Let's denote its context with o. To find out the context of the word, a sliding window size m will be defined, meaning that the model will search for context from position $t - m$ to $t + m$. $P\left(w_{t+j}|w_t\right)$ is the probability of a word w_t to be in the context w_{t+j}, where $j \in \overline{1, m}$.

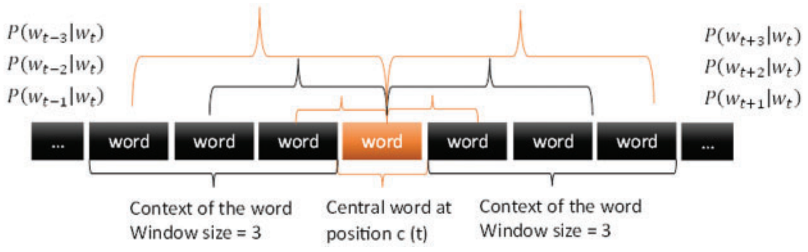

Figure 3: Sequence of words and the context of the central position

Setting the target at position t, sg will maximize the likelihood of the context words for a certain word. In other words, it computes the probability of the model for the prediction of the context words for a center word and it maximizes the probability. The maximum likelihood function $L(\theta)$, where θ represents the parameters of the model, is a product of probabilities over the words in the corpus T as defined in Eq. (3):

$$L(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m j \ne 0} P\left(w_{t+j}|w_t;\theta\right). \tag{3}$$

The logarithm operator is required to transform multiplication into summation and apply derivatives for the minimization problem. Therefore, Eq. (3) becomes:

$$J(\theta) = -\frac{1}{T} logL(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \le j \le m j \ne 0} logP\left(w_{t+j}|w_t;\theta\right). \tag{4}$$

To compute the probability of the context, each word is represented by a set of two vectors: U_w when word w is a context word o and V_w when word w is a target word c. Therefore, using the two vectors, the probability for the target word c to be in the context o is given in Eq. (5):

$$P(o|c) = \frac{exp\left(U_o^T V_c\right)}{\sum_{w \in T} exp\left(U_w^T V_c\right)}. \tag{5}$$

The numerator captures the similarity between these two vectors depicting context and target (higher the similarity, higher the probability), whereas the denominator normalizes the probability over the vocabulary. The weights vector θ consists of the elements of the two vectors U_w and V_w.

$$\theta = \begin{bmatrix} V_{w1} \\ V_{w2} \\ \ldots \\ V_{wn} \\ U_{w1} \\ U_{w2} \\ \ldots \\ U_{wn} \end{bmatrix}. \tag{6}$$

Using the SGD algorithm, the weights are updated to maximize the likelihood. Thus, the derivative of the loss function with respect to U and V is calculated. First, let's calculate the derivative of $J(\theta)$ from Eq. (4) with respect to V_c:

$$\frac{\partial J(\theta)}{\partial V_c} = \frac{\partial}{\partial V_c}\left(\log\left(\exp\left(U_o^T V_c\right)\right)\right) - \frac{\partial}{\partial V_c}\left(\log \sum_{w=1}^{V} \exp\left(U_w^T V_c\right)\right). \tag{7}$$

As log(exp(x)) is equal to x, the first part of Eq. (7) becomes:

$$\frac{\partial}{\partial V_c}\left(\log\left(\exp\left(U_o^T V_c\right)\right)\right) = \frac{\partial}{\partial V_c}\left(U_o^T V_c\right) = U_o. \tag{8}$$

Considering the derivative of log(x), the second part of Eq. (7) becomes:

$$\frac{\partial}{\partial V_c}\left(\log \sum_{w=1}^{V} \exp\left(U_w^T V_c\right)\right) = \frac{1}{\sum_{w=1}^{V} \exp\left(U_w^T V_c\right)} \sum_{x=1}^{V} \frac{\partial}{\partial V_c} \exp\left(U_x^T V_c\right). \tag{9}$$

Considering the derivative of exp(x), Eq. (9) is equivalent with:

$$\frac{1}{\sum_{w=1}^{V} \exp\left(U_w^T V_c\right)} \sum_{x=1}^{V} \frac{\partial}{\partial V_c} \exp\left(U_x^T V_c\right) = \sum_{x=1}^{V} \frac{\exp\left(U_x^T V_c\right)}{\sum_{w=1}^{V} \exp\left(U_w^T V_c\right)} \times U_x. \tag{10}$$

The result is the probability. Thus, Eq. (10) becomes:

$$\sum_{x=1}^{V} \frac{\exp\left(U_x^T V_c\right)}{\sum_{w=1}^{V} \exp\left(U_w^T V_c\right)} \times U_x = \sum_{x=1}^{V} P\left(x|c\right) \times U_x. \tag{11}$$

Combining the two terms, we obtain:

$$\frac{\partial J(\theta)}{\partial V_c} = -U_o + \sum_{x=1}^{V} P\left(x|c\right) \times U_x. \tag{12}$$

Considering that U_o is the vector representing the context word and $\sum_{x=1}^{V} P\left(x|c\right) \times U_x$ is the probability of the context, the two components are necessary to compute the vector V_c so that to maximize the likelihood. Similarly, we calculate the derivative of $J(\theta)$ with respect to U_w.

$$\begin{cases} \dfrac{\partial J(\theta)}{\partial U_w} = \sum_{x=1}^{V} P\left(x|c\right) \times V_c, & w \neq o \\ \dfrac{\partial J(\theta)}{\partial U_w} = -V_c + \sum_{x=1}^{V} P\left(x|c\right) \times V_c, & w = o \end{cases}. \tag{13}$$

Using the two partial derivatives, the weights can be updated:

$$\theta\left(V_c, U_w\right) = \theta - \alpha\frac{\partial J\left(\theta\right)}{\partial\theta}; \tag{14}$$

$$\frac{\partial J(\theta)}{\partial\theta} = \begin{bmatrix} \dfrac{\partial J(\theta)}{\partial V_c} \\ \dfrac{\partial J(\theta)}{\partial U_w} \end{bmatrix}, \tag{15}$$

where α–learning rate.

To handle the exponential function that is time consuming, the *negative sampling* method is used which maximizes the probability of real context occurring around the target. The loss function becomes:

$$J_{ns}\left(o, c, U\right) = -\log\left(\sigma\left(U_o^T V_c\right)\right) - \sum_{k=1}^{K}\log\left(\sigma\left(U_k^T V_c\right)\right), \tag{16}$$

where K-number of negative samples (ns).

Calculating the derivative of Eq. (16) with respect to the weights U_w, U_k, and V_c, we obtain:

$$\begin{aligned} \frac{\partial J_{ns}\left(\theta\right)}{\partial V_c} &= -\sigma\left(-U_o^T V_c\right) U_o + \sum_{k=1}^{K}\sigma\left(U_k^T V_c\right) U_k; \\ \frac{\partial J_{ns}(\theta)}{\partial U_o} &= -\sigma\left(-U_o^T V_c\right) V_c; \\ \frac{\partial J_{ns}(\theta)}{\partial U_k} &= \sum_{k=1}^{K}\sigma\left(U_k^T V_c\right) V_c. \end{aligned} \tag{17}$$

3.3.2 GloVe Model

GloVe is a technique proposed by Stanford University for obtaining vector representations for words using an unsupervised learning algorithm. GloVe converts each word in a corpus of text into a value in a high-dimensional space, placing similar words nearby. The authors of the GloVe concluded that GloVe considerably outperformed word2vec [66]. GloVe is based on building a co-occurrence matrix with rows representing words and columns representing the context. This matrix computes the frequency of words in a context. It may have a high dimensionality. To reduce it, the reconstruction of the co-occurrence matrix is required that is also known as reconstruction loss.

A Global Log-Bilinear Regression (GBLR) model that mixes the global matrix factorization (Latent Semantic Analysis (LSA)) and local context window (embedding the advantages of the above-mentioned sg algorithm) is employed by GloVe. LSA provides statistical information, but its vector representations are outperformed in word analogies. On the other hand, sg creates enhanced vector representations and provides good results in word analogies, but the statistical information captured from the entire corpus is not used at its maximum as sg trains only on local windows not on the global co-occurrence matrix, losing the global view. GloVe provides a vector space with meaningful substructures by training only on non-zero elements in a global co-occurrence matrix, rather than on entire sparse matrix or context windows in the larger corpus. The two techniques of the word2vec approach, namely sg and cbow, take into account only the local context, not the global context. More advanced feature techniques do exist, such as ELMo, sentence encoders (doc2vec, Sentence-BERT, Universal Encoder, InferSent), and Open-AI GPT. Thus, GloVe uses a GBLR model with a simple weighted least squares method. GloVe embeddings are based on the ratio of probabilities that indicates the meanings of words. Let's consider function F which models the ratio of probabilities relationship that is the target of the neural network model.

$$F\left(w_i, w_j, \overline{w_k}\right) = \frac{P_{ik}}{P_{jk}}, \tag{18}$$

where w_i, w_j–words in context; $\overline{w_k}$–out of context; P_{ik}–probability that word i to appear in the context of word k; P_{jk}–probability that word j to appear in the context of word k.

To balance the two sides of the Eq. (18) that consist of both vectors and scalars, aiming to train function F to encode the information regarding the ratio of the two probabilities available in the global corpus, we will consider the transpose linear difference between the two vectors or the distance between w_i, w_j multiplied by the $\overline{w_k}$. Therefore, Eq. (18) becomes:

$$F\left(\left(w_i - w_j\right)^T \overline{w_k}\right) = \frac{P_{ik}}{P_{jk}}. \tag{19}$$

As a word can be either in the context or outside the context, the left side of Eq. (19) becomes:

$$\begin{aligned} &F\left(\left(w_i - w_j\right)^T \overline{w_k}\right) = F\left(w_i^T \overline{w_k} + \left(-w_j^T \overline{w_k}\right)\right);\\ &F\left(w_i^T \overline{w_k} + \left(-w_j^T \overline{w_k}\right)\right) = F\left(w_i^T \overline{w_k}\right) \times F\left(-w_j^T \overline{w_k}\right);\\ &F\left(w_i^T \overline{w_k}\right) \times F\left(-w_j^T \overline{w_k}\right) = F\left(w_i^T \overline{w_k}\right) \times F\left(w_j^T \overline{w_k}\right)^{-1};\\ &F\left(\left(w_i - \overline{w_j}\right)^T \overline{w_k}\right) = \frac{F\left(w_i^T \overline{w_k}\right)}{F\left(w_j^T \overline{w_k}\right)}. \end{aligned} \tag{20}$$

Thus, combining Eqs. (19) and (20), we can write the numerator as:

$$F\left(w_i^T \overline{w_k}\right) = P_{ik} = \frac{X_{ik}}{X_i}, \tag{21}$$

where X_i–number of times any word appears in the context of word i; X_{ik}–number of times word k appears in the context of word i.

Replacing F with the exponential function, we obtain:

$$e^{\left(w_i^T \overline{w_k}\right)} = P_{ik} = \frac{X_{ik}}{X_i}. \tag{22}$$

Applying logarithm function on Eq. (22), we obtain:

$$w_i^T \overline{w_k} = \log\left(P_{ik}\right) = \log\left(X_{ik}\right) - \log\left(X_i\right). \tag{23}$$

As $\log\left(X_i\right)$ has no influence on k, it can be added as a bias b_i along with a bias for $\overline{w_k}$. Thus, Eq. (23) can be simplified as follows:

$$w_i^T \overline{w_k} + b_i + \overline{b_k} = \log\left(P_{ik}\right) = \log\left(X_{ik}\right). \tag{24}$$

In Eq. (24), equal weight is given to all elements in the co-occurrence matrix, which is a major drawback causing noise due to the equal importance allocation. To handle this issue, a weighting function $f\left(X_{ij}\right)$ is inserted that weighs the words based on the content. Therefore, a weighted least squares regression model is used, and the loss function J of the GloVe model is obtained.

$$J = \sum_{i,j=1}^{T} f\left(X_{ij}\right)\left(w_i^T \overline{w_j} + b_i + \overline{b_j} - \log\left(X_{ij}\right)\right)^2, \tag{25}$$

where T–the vocabulary dimension.

The weighting function can be as in Eq. (26):

$$f(x) = \begin{cases} \left(\dfrac{x}{x_{max}}\right)^{\alpha}, & x < x_{max} \\ 1, & otherwise \end{cases}. \tag{26}$$

If the two biases are eliminated to simplify the Eq. (25), the loss function of the model becomes:

$$\hat{J} = \sum_{i,j=1}^{T} f\left(X_{ij}\right)\left(w_i^T \overline{w_j} - log\left(X_{ij}\right)\right)^2. \tag{27}$$

3.4 *Assessment*

The results obtained with several feature builders and supervised ML algorithms (classifiers) are weighted considering the following criteria: Accuracy (A); F1 score (F1); Type error II (FPR); Computational time (Feature Building Time (FBT) plus Training Time (TT)), where ω–weight.

$$Rank = \omega_1 \times A + \omega_2 \times F1 + \omega_3 \times \left(1 - \frac{FPR}{\max(FPR)}\right) + \omega_4 \times \left(1 - \frac{FBT + TT}{\max(FBT) + \max(TT)}\right). \tag{28}$$

In the next section, we will apply several techniques to obtain features and classify the URLs as benign or malicious. To check the generalization capacity of the classifiers, we train and test the results using four datasets.

4 Results

In this section, we present the input datasets and the results of the simulations performed using different approaches: splitting the URL based on its structure and building features as well as various vectorization techniques. We evaluate classification algorithms using specific metrics. The results are compared, and the best solution is chosen based on the performance of the algorithms.

Four input datasets are included in simulations as in Table 1. Table 1 displays the total number of records, differentiating between those classified as benign or malicious, and lists the open data sources from which these records were obtained.

Taking the first dataset url0, 76.8% of the records are benign. Using the frequencies of the words, we notice that the most frequent word (higher the font, higher the frequency) after each URL was tokenized is “https”, then “index”, “Wikipedia”, “Youtube”, “Facebook” and so on. The benign records vary between 65% and 82% of the entire dataset. The pre-processing of the URLs (except for URL structure-based feature engineering) consists of several tasks: the special characters from each URL are removed; and the tokens are created to be inserted into the feature-building methods.

Table 1: Characteristics of the input data sources

Source	Total records	Benign	Malicious	Open source
url0	450,176	345,738 (76.8%)	104,438	https://www.kaggle.com/code/siddharthkumar25/detect-malicious-url-using-ml/input
url1	651,191	428,103 (65.7%)	Defacement 96,457 phishing 94,111 malware 32,520	https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset
url2	42,249	31,823 (75.3%)	10,426	Received from another researcher[1]
url3	420,464	344,821 (82%)	75,643	https://www.section.io/engineering-education/detecting-malicious-url-using-machine-learning/

Note: [1]https://github.com/simonavoprea/URL

4.1 Hash Vectorized Features

The results for the analyzed first case study (url0) are presented below. Table 2 shows the performance metrics when generating features with a hash vectorizer. The confusion matrices using the hash vectorized features are presented in Fig. 4. The evaluation or the assessment of the algorithm's performance is achieved by interpreting several metrics, such as accuracy, F1 score, confusion matrix investigating the first diagonal as well as type error I and type error II. Ranking the algorithms consists of calculating a weighted average among the computational time that has to be small, the performance in terms of F1 score, and the false negative or the type error II that is also known as the miss or the underestimation of the algorithm. The first and third-ranking indicators are minimized, whereas the second one is maximized.

Table 2: Performance metrics for Hash Vectorizer

Model	Accuracy	F1 score	FPR	TPR	FBT (min)	TT
LR	0.996	0.992	265	204	5.76	0:00:12.17
SVC	0.996	0.992	253	189		0:00:02.25
RF	0.997	0.995	217	90		0:07:05.08
XGB	0.997	0.994	207	70		0:01:36.85

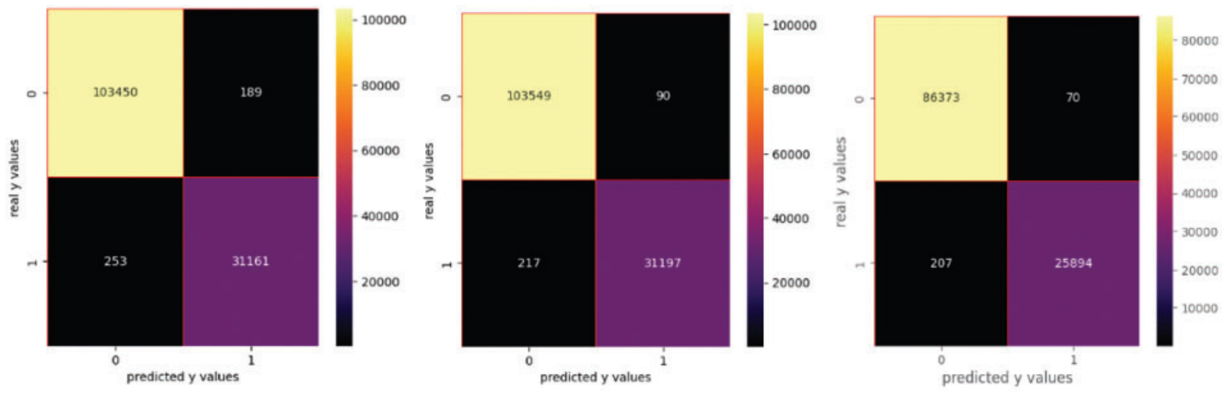

Figure 4: Confusion matrix for hash vectorizer

In this scenario, we obtain the best results using RF and XGB. XGB is faster than RF, but the F1 score is slightly smaller.

4.2 TF-IDF Vectorized Features

Table 3 shows the performance metrics when generating features with TD-IDF. The confusion matrices using the TF-IDF vectorized features are presented in Fig. 5.

Table 3: Performance metrics for TD-IDF Vectorizer

Model	Accuracy	F1 score	FPR	TPR	FBT (min)	TT
NB	0.989	0.976	880	1585	6.96	0:00:01.25
LR	0.994	0.988	1174	10		0:00:21.08
SVC	0.996	0.991	813	34		0:00:02.83
RF	0.998	0.996	211	200		0:50:26.54
XGB	0.997	0.994	273	252		0:33:54.46

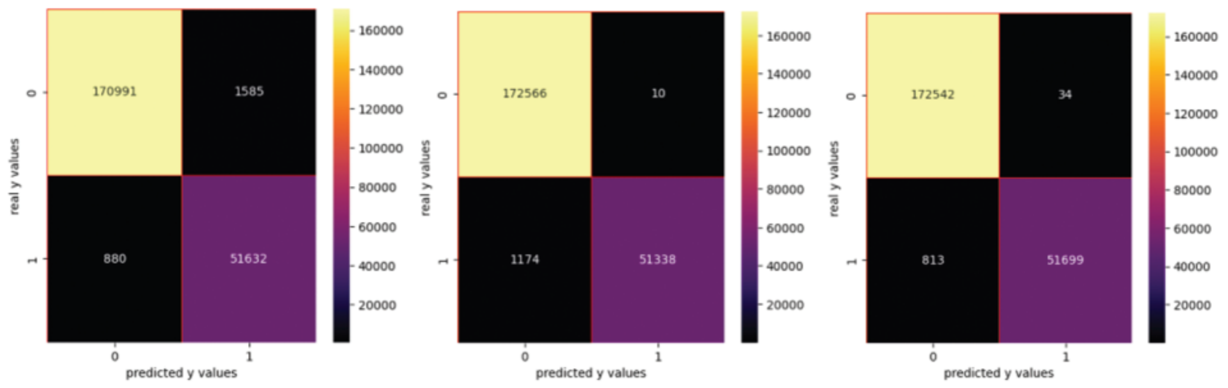

Figure 5: (Continued)

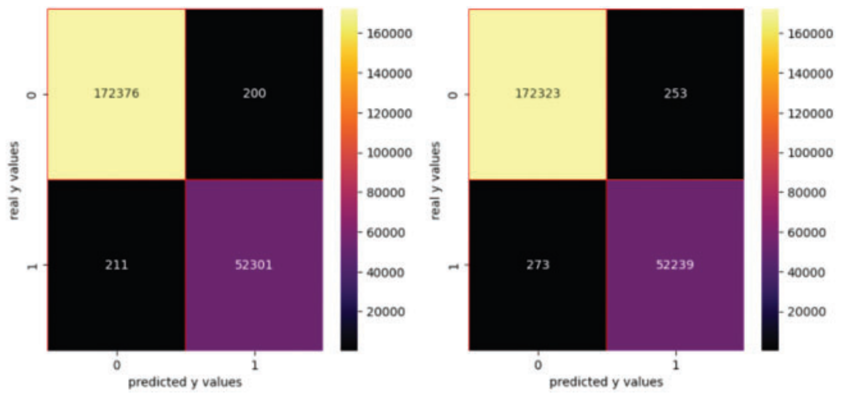

Figure 5: Confusion matrix for TD-IDF

When features are vectorized using TF-IDF, we obtain the best results with RF and XGB, but they are time consuming compared with LR and SVC that provide good results in terms of F1 score.

Comparing the two vectorization techniques, we estimate that hash vectorization technique is less time consuming. Therefore, when fast results are required, hash vectorization technique is recommended.

4.3 URL Structure-Based Features

A heatmap showing the correlations among variables can be obtained as in Fig. 6.

For a certain dataset (url0), the highest correlations with the target (result) are recorded for suspicious words (0.25), number of digits (0.18), existence of www (−0.89), existence of https (−0.95) and so on. The features are built based on the URL structure. Table 4 shows the performance metrics using feature engineering. The confusion matrices using feature engineering are presented in Fig. 7.

In this approach, we obtain the best results using RF, XGB and LGB algorithms. It can be used when fast results are expected as this approach is less time-consuming than the previous one.

4.4 Word2vec Features

Table 5 shows the performance metrics when generating features with word2vec. The confusion matrices using the word2vec vectorized features are presented in Fig. 8.

In this scenario, XGB is the best algorithm for classification, but it is also one of the slowest. Due to its higher FTB and TT, this approach is not recommended when fast results are expected. SVC algorithm also provides good results in terms of F1 score and it is much faster.

4.5 GloVe Features

As GloVe is time-consuming, a sample of 50,000 records is extracted and shuffled. Table 6 shows the performance metrics when generating features with GloVe. The confusion matrices using the GloVe vectorized features are presented in Fig. 9.

This approach implies a high FBT (187 min), therefore the time to build the features is much higher compared with other approaches. In this context, SVC, LR and XGB provide the best results.

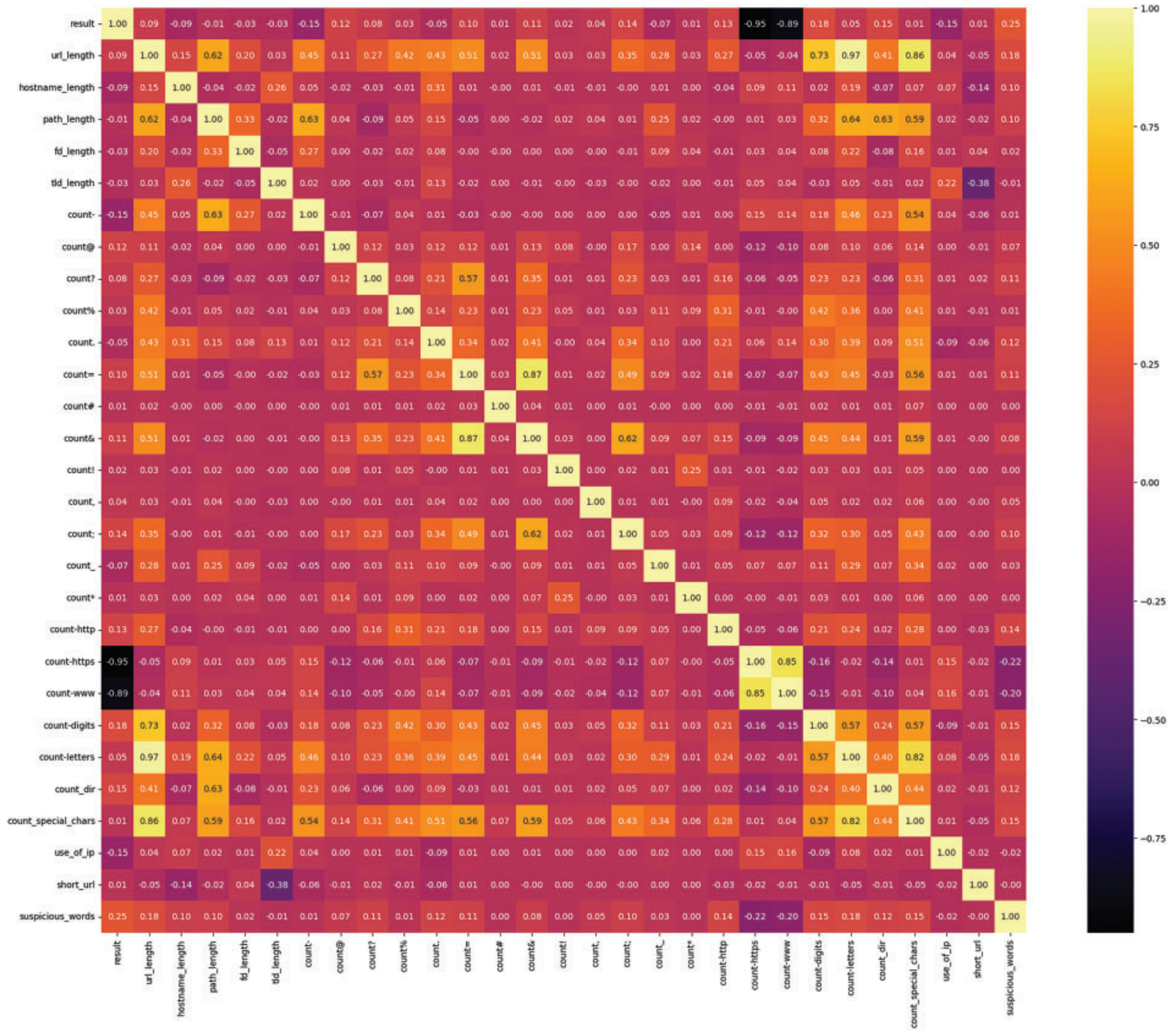

Figure 6: Heatmap of the input variables

Table 4: Performance metrics for Feature Engineering method using the URL structure

Model	Accuracy	F1 score	FPR	TPR	FBT (min)	TT
DT	0.996	0.992	197	129	8.92	0:00:18.17
LR	0.992	0.988	462	171		0:00:12.07
SVC	0.996	0.993	175	149		0:00:02.25
RF	0.997	0.995	165	55		0:07:05.08
XGB	0.997	0.994	167	50		0:01:36.85
LGB	0.997	0.993	167	62		0:11:34.60

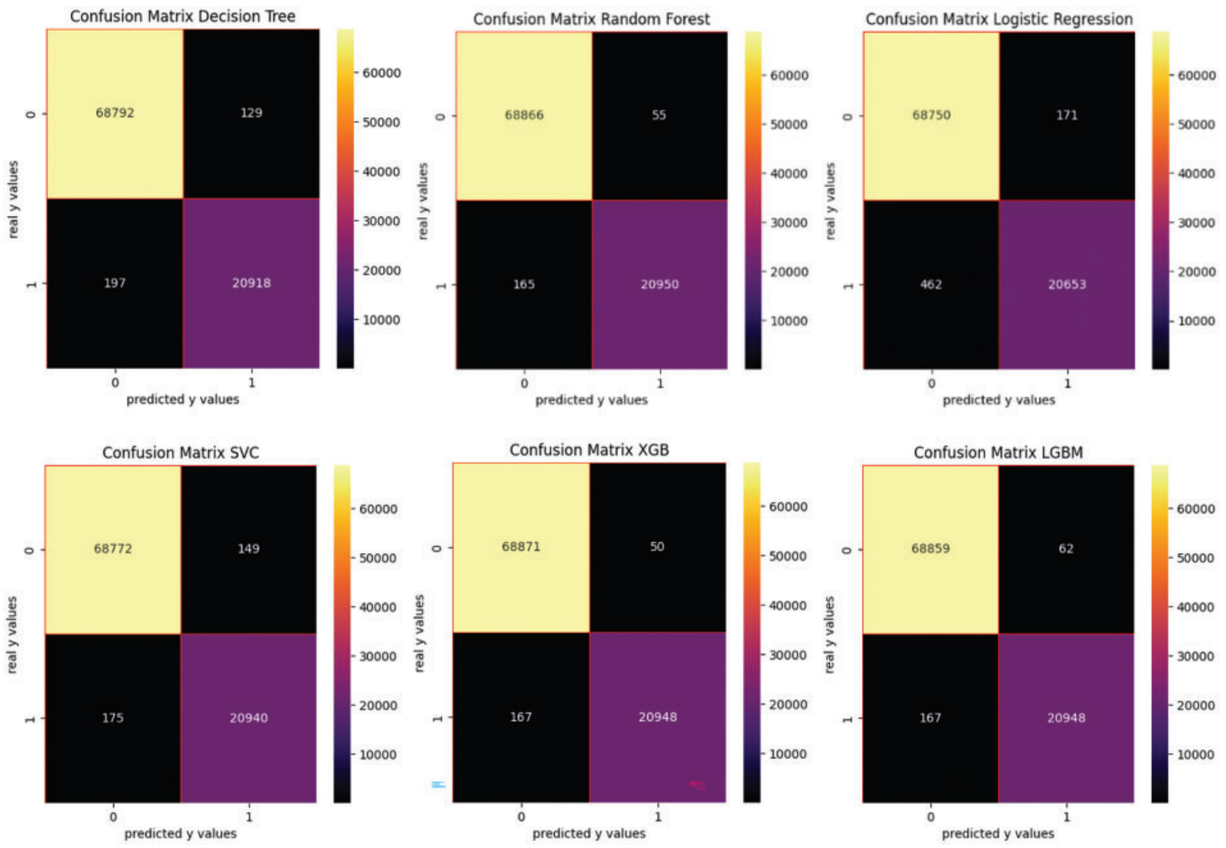

Figure 7: Confusion matrix for FE method

Table 5: Performance metrics for word2vec

Model	Accuracy	F1 score	FPR	TPR	FBT (min)	TT
LR	0.997	0.993	244	143	20.02	0:00:48.16
SVC	0.997	0.993	228	163		0:01:45.11
RF	0.997	0.988	629	56		0:38:18.95
XGB	0.995	0.990	487	76		0:26:47.10

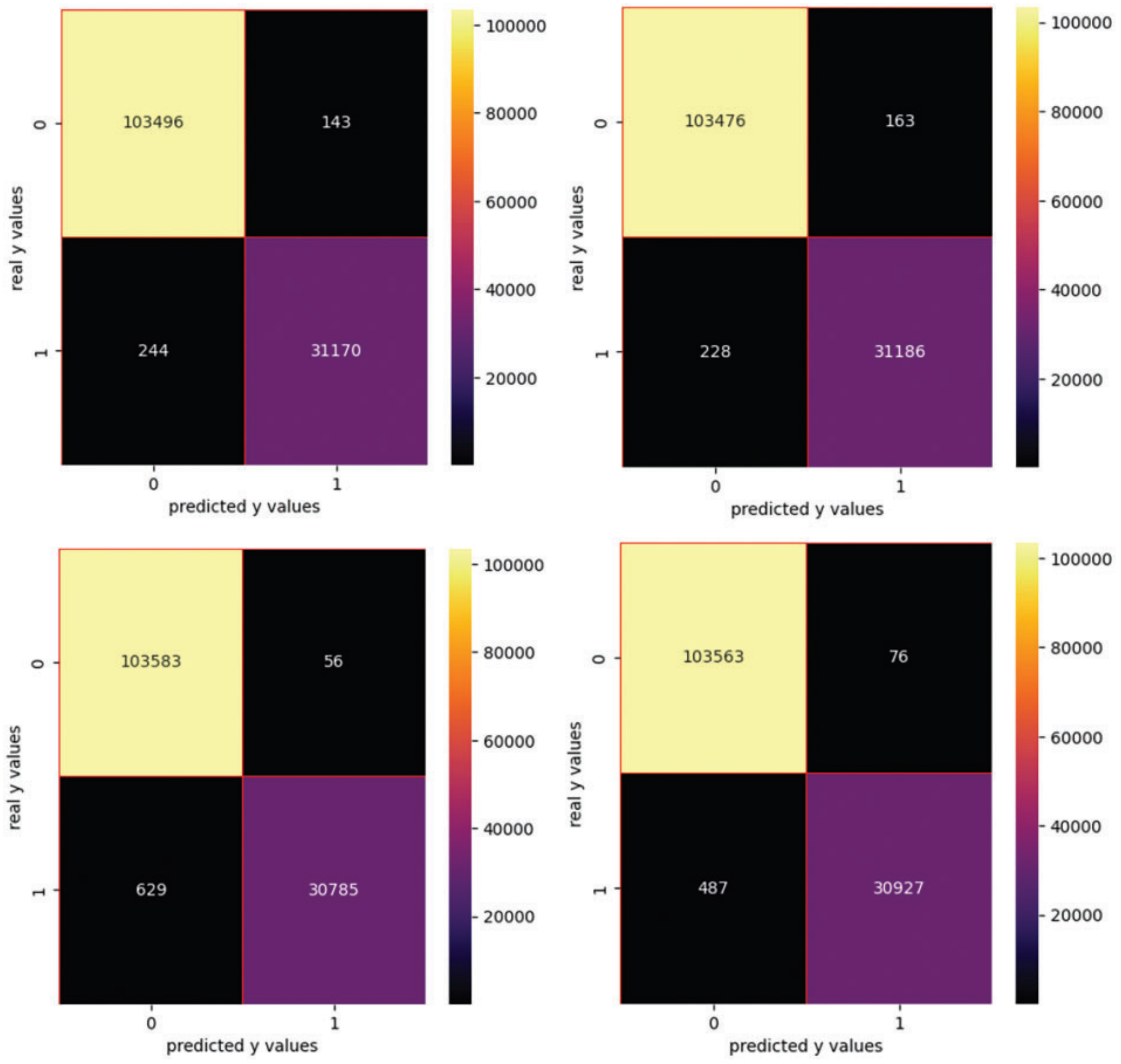

Figure 8: Confusion matrix for word2vec

Table 6: Performance metrics GloVe (50,000 records)

Model	Accuracy	F1 score	FPR	TPR	FBT (min)	TT
LR	0.991	0.981	125	42	187	0:00:01.78
SVC	0.992	0.982	111	45		0:00:02.87
RF	0.985	0.967	254	40		0:01:57.52
XGB	0.991	0.981	132	33		0:01:42.80

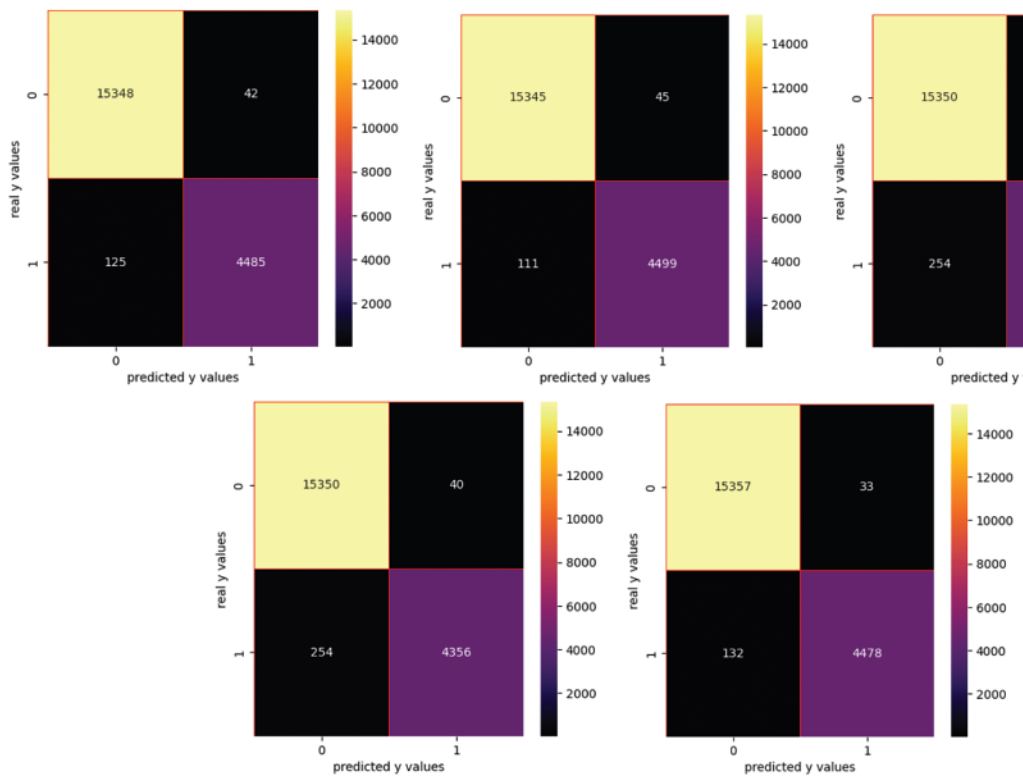

Figure 9: Confusion matrix using GloVe

5 Conclusion

In this paper, we proposed a framework for detecting malicious URLs. It consists of four stages: 1) Using multiple datasets to which exploratory data analysis and pre-processing were applied; 2) Building features with three options a) frequency-based features; b) URL structure-based features and c) prediction-based features; 3) Using several classifiers to predict whether the URL is malicious or benign; and 4) Assessment. The reason for using four datasets is related to the capacity of the framework to generalize.

Five types of features were built using hash vectorizer, TD-IDF, feature engineering based on the URL structure and more advanced: word2vec (skip-gram) and GloVe. The mathematics behind the more advanced models was described in detail. Four ML-supervised models for classification were applied for all case studies, such as LR, SVC, RF and XGB. DT, NB and LGBM were also applied for certain methods.

The assessment of the methods in the proposed framework was performed using a ranking equation that weighs the four performance criteria: accuracy, F1 score, type error II and computational time that also includes the feature building time. Moreover, by applying DL in classifying URLs (CNN, Simple RNN, LSTM, bi-LSTM), good results were obtained. Although the performance of these models is good, the computational time is much higher. However, the performance of the above-mentioned ML algorithms is also good, thus the usage of the DL algorithms for URLs detection was not necessary.

Using the ranking equation with equal weights proposed in this paper and the results obtained for the four datasets, the *word2vec*, namely the skip-gram algorithm outperformed the other feature builders. Moreover, *XGB* outperformed the other classifiers. Hence, our suggestion is to merge word2vec and XGB to identify malicious URLs. In our future endeavors, we plan to explore the potential of generative pre-trained transformer models, advanced NLP techniques and large databases to enhance the speed of detection. The computational time remains a limitation for advanced feature builders, and to mitigate this, we will consider implementing parallel processing.

Acknowledgement: This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS-UEFISCDI, Project Number PN-III-P4-PCE-2021-0334, within PNCDI III.

Funding Statement: This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS- UEFISCDI, Project Number PN-III-P4-PCE-2021-0334, within PNCDI III.

Author Contributions: Adela Bâra: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing–Original Draft, Writing–Review and Editing, Visualization, Supervision. Simona-Vasilica Oprea: Conceptualization, Validation, Formal analysis, Investigation, Writing–Original Draft, Writing–Review and Editing, Visualization, Project administration. All authors reviewed the results and approved the final version of the manuscript.

Data Availability Statement: The data will be made available upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Zhao and J. Chen, "A survey on differential privacy for unstructured data content," *ACM Comput. Surv.*, vol. 54, no. 10s, pp. 1–28, 2022. doi: 10.1145/3490237.

[2] L. Hickman, S. Thapa, L. Tay, M. Cao, and P. Srinivasan, "Text preprocessing for text mining in organizational research: Review and recommendations," *Organ. Res. Methods.*, vol. 25, no. 1, pp. 114–146, 2022. doi: 10.1177/1094428120971683.

[3] A. Kurniasih and L. P. Manik, "On the role of text preprocessing in BERT embedding-based DNNs for classifying informal texts," *Int. J. Adv. Comput. Sci. Appl.*, 2022. doi: 10.14569/issn.2156-5570.

[4] L. Che, X. Yang, and L. Wang, "Text feature extraction based on stacked variational autoencoder," *Microprocess. Microsyst.*, vol. 76, no. 1, pp. 103063, 2020. doi: 10.1016/j.micpro.2020.103063.

[5] C. Toraman, F. Şahinuç, E. H. Yilmaz, and I. B. Akkaya, "Understanding social engagements: A comparative analysis of user and text features in Twitter," *Soc. Netw. Anal. Min.*, vol. 12, no. 1, pp. 1, 2022. doi: 10.1007/s13278-022-00872-1.

[6] Z. Zhu, J. Liang, D. Li, H. Yu, and G. Liu, "Hot topic detection based on a refined TF-IDF algorithm," *IEEE Access*, vol. 7, pp. 26996–27007, 2019. doi: 10.1109/ACCESS.2019.2893980.

[7] O. I. Gifari, M. Adha, F. Freddy, and F. F. S. Durrand, "Analisis sentimen review film menggunakan TF-IDF dan support vector machine," *J. Inf. Technol.*, vol. 2, no. 1, pp. 36–40, 2022. doi: 10.46229/jifotech.v2i1.330.

[8] D. Malandrino, R. De Prisco, M. Ianulardo, and R. Zaccagnino, "An adaptive meta-heuristic for music plagiarism detection based on text similarity and clustering," *Data Min. Knowl. Discov.*, vol. 36, no. 4, pp. 1301–1334, 2022. doi: 10.1007/s10618-022-00835-2.

[9] H. Jelodar *et al.*, "Latent Dirichlet allocation (LDA) and topic modeling: Models, applications, a survey," *Multimed. Tools Appl.*, vol. 78, no. 11, pp. 15169–15211, 2019. doi: 10.1007/s11042-018-6894-4.

[10] C. Sharma, S. Sharma, and Sakshi, "Latent DIRICHLET allocation (LDA) based information modelling on BLOCKCHAIN technology: A review of trends and research patterns used in integration," *Multimed. Tools Appl.*, vol. 81, no. 25, pp. 36805–36831, 2022. doi: 10.1007/s11042-022-13500-z.
[11] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res*, vol. 3, pp. 993–1022, 2003. doi: 10.1017/9781009218245.012.
[12] K. A. Coulter, "The impact of news media on Bitcoin prices: Modelling data driven discourses in the crypto-economy with natural language processing," *R Soc. Open Sci.*, vol. 9, no. 4, pp. 1, 2022. doi: 10.1098/rsos.220276.
[13] A. Mishra and Fancy, "Efficient detection of phising hyperlinks using machine learning," *Int. J. Cybern. Informatics.*, vol. 10, no. 2, pp. 23–33, 2021. doi: 10.5121/ijci.2021.100204.
[14] F. Yahva *et al.*, "Detection of Phising websites using machine learning approaches," in *2021 Int. Conf. Data Sci. Its Appl. ICoDSA 2021*, Bandung, Indonesia, 2021. doi: 10.1109/ICoDSA53588.2021.9617482.
[15] H. Xu, Y. Zhou, J. Ming, and M. Lyu, "Layered obfuscation: A taxonomy of software obfuscation techniques for layered security," *Cybersecurity*, vol. 3, no. 1, pp. 190,901, 2020. doi: 10.1186/s42400-020-00049-3.
[16] H. D. Menéndez and G. Suárez-Tangil, "ObfSec: Measuring the security of obfuscations from a testing perspective," *Expert. Syst. Appl.*, vol. 210, no. 5, pp. 118298, 2022. doi: 10.1016/j.eswa.2022.118298.
[17] F. A. Ghaleb, M. Alsaedi, F. Saeed, J. Ahmad, and M. Alasli, "Cyber threat intelligence-based malicious URL detection model using ensemble learning," *Sensors*, vol. 22, no. 9, pp. 3373, 2022. doi: 10.3390/s22093373.
[18] Y. Liang, Q. Wang, K. Xiong, X. Zheng, Z. Yu, and D. Zeng, "Robust detection of malicious URLs with self-paced wide & deep learning," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 2, pp. 717–730, 2022. doi: 10.1109/TDSC.2021.3121388.
[19] N. T. Lam, "Developing a framework for detecting phishing URLs using machine learning," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 11, no. 11, pp. 61–67, 2021. doi: 10.46338/ijetae1121_08.
[20] S. Das Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, "Modeling hybrid feature-based phishing websites detection using machine learning techniques," *Ann. Data Sci.*, vol. 2, no. 3, pp. 217–242, 2022. doi: 10.1007/s40745-022-00379-8.
[21] M. A. El-Rashidy, "A Smart model for web phishing detection based on new proposed feature selection technique," *Menoufia J. Electron. Eng. Res.*, vol. 30, no. 1, pp. 97–104, 2021. doi: 10.21608/mjeer.2021.146286.
[22] M. SatheeshKumar, K. G. Srinivasagan, and G. UnniKrishnan, "A lightweight and proactive rule-based incremental construction approach to detect phishing scam," *Inf. Technol. Manag.*, vol. 23, no. 4, pp. 271–298, 2022. doi: 10.1007/s10799-021-00351-7.
[23] M. Aljabri *et al.*, "Detecting Malicious URLs using machine learning techniques: Review and research directions," *IEEE Access*, vol. 10, pp. 121395–121417, 2022. doi: 10.1109/ACCESS.2022.3222307.
[24] C. M. R. da Silva, E. L. Feitosa, and V. C. Garcia, "Heuristic-based strategy for phishing prediction: A survey of URL-based approach," *Comput. Secur.*, vol. 88, no. 7, pp. 101613, 2020. doi: 10.1016/j.cose.2019.101613.
[25] H. Zhao, Z. Chang, W. Wang, and X. Zeng, "Malicious domain names detection algorithm based on lexical analysis and feature quantification," *IEEE Access*, vol. 7, pp. 128990–128999, 2019. doi: 10.1109/ACCESS.2019.2940554.
[26] C. M. Chen, J. J. Huang, and Y. H. Ou, "Efficient suspicious URL filtering based on reputation," *J. Inf. Secur. Appl.*, vol. 20, no. 2, pp. 26–36, 2015. doi: 10.1016/j.jisa.2014.10.005.
[27] A. A. Orunsolu, A. S. Sodiya, and A. T. Akinwale, "A predictive model for phishing detection," *J. King Saud Univ.—Comput. Inf. Sci.*, vol. 34, no. 2, pp. 232–247, 2022. doi: 10.1016/j.jksuci.2019.12.005.
[28] S. Kim, J. Kim, and B. B. H. Kang, "Malicious URL protection based on attackers' habitual behavioral analysis," *Comput. Secur.*, vol. 77, no. 2, pp. 790–806, 2018. doi: 10.1016/j.cose.2018.01.013.
[29] D. T. Mosa, M. Y. Shams, A. A. Abohany, E. S. M. El-Kenawy, and M. Thabet, "Machine learning techniques for detecting phishing URL attacks," *Comput. Mater. Contin.*, vol. 75, no. 1, pp. 1271–1290, 2023. doi: 10.32604/cmc.2023.036422.

[30] O. V. Lee *et al.*, "A malicious URLs detection system using optimization and machine learning classifiers," *Indones J. Electr. Eng. Comput. Sci.*, vol. 17, no. 3, pp. 1210, 2020. doi: 10.11591/ijeecs.v17.i3.pp1210-1214.

[31] H. C. Kim, Y. H. Choi, and D. H. Lee, "JsSandbox: A framework for analyzing the behavior of malicious JavaScript code using internal function hooking," *KSII Trans. Internet Inf. Syst.*, 2012. doi: 10.3837/tiis.2012.02.019.

[32] Y. G. Zeng, "Identifying email threats using predictive analysis," in *2017 Int. Conf. Cyber Secur. Prot. Digit. Serv. Cyber Secur. 2017*, London, UK, 2017. doi: 10.1109/CyberSecPODS.2017.8074848.

[33] N. F. Ghalati, N. F. Ghalaty, and J. Barata, "Towards the detection of malicious URL and domain names using machine learning," in *IFIP Adv. Inf. Commun. Technol.*, Costa de Caparica, Portugal, 2020. doi: 10.1007/978-3-030-45124-0_10.

[34] B. Sun, M. Akiyama, T. Yagi, M. Hatada, and T. Mori, "Automating URL blacklist generation with similarity search approach," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 4, pp. 873–882, 2016. doi: 10.1587/transinf.2015ICP0027.

[35] C. Opara, Y. Chen, and B. Wei, "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics," *Expert. Syst. Appl.*, vol. 236, no. 12, pp. 121183, 2024. doi: 10.1016/j.eswa.2023.121183.

[36] M. Atrees, A. Ahmad, and F. Alghanim, "Enhancing detection of malicious urls using boosting and lexical features," *Intell Autom. Soft Comput.*, vol. 31, no. 3, pp. 1405–1422, 2022. doi: 10.32604/iasc.2022.020229.

[37] S. M. Nair, "Detecting malicious URL using machine learning: A survey," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 8, no. 5, pp. 2670–2677, 2020. doi: 10.22214/ijraset.2020.5447.

[38] A. S. Rafsanjani, N. B. Kamaruddin, H. M. Rusli, and M. Dabbagh, "QsecR: Secure QR code scanner according to a novel malicious URL detection framework," *IEEE Access*, vol. 11, pp. 92523–92539, 2023. doi: 10.1109/ACCESS.2023.3291811.

[39] N. A. Alfouzan and C. Narmatha, "A systematic approach for malware URL recognition," in *Proc. 2022 2nd Int. Conf. Comput. Inf. Technol. ICCIT 2022*, Tabuk, Saudi Arabia, 2022. doi: 10.1109/ICCIT52419.2022.9711614.

[40] V. K. Nadar, B. Patel, V. Devmane, and U. Bhave, "Detection of phishing websites using machine learning approach," in *2021 2nd Glob. Conf. Adv. Technol. GCAT 2021*, Bangalore, India, 2021. doi: 10.1109/GCAT52182.2021.9587682.

[41] R. Aswani, S. P. Ghrera, S. Chandra, and A. K. Kar, "Outlier detection among influencer blogs based on off-site web analytics data," in *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Delhi, India, 2017. doi: 10.1007/978-3-319-68557-1_23.

[42] A. Mohammadi and S. A. H. Golpayegani, "SenseTrust: A sentiment based trust model in social network," *J. Theor. Appl. Electron. Commer. Res.*, vol. 16, no. 6, pp. 2031–2050, 2021. doi: 10.3390/jtaer16060114.

[43] C. Do Xuan, H. D. Nguyen, and T. V. Nikolaevich, "Malicious URL detection based on machine learning," *Int. J. Adv. Comput. Sci. Appl.*, 2020. doi: 10.14569/issn.2156-5570.

[44] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious URLs using lexical analysis," in *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Taipei, Taiwan, 2016. doi: 10.1007/978-3-319-46298-1_30.

[45] A. Kumar and S. Maity, "Detecting Malicious URLs using lexical analysis and network activities," in *4th Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2022—Proc.*, Coimbatore, India, 2022. doi: 10.1109/ICIRCA54612.2022.9985586.

[46] A. Joshi, L. Lloyd, P. Westin, and S. Seethapathy, "Using lexical features for malicious URL Detection—A machine learning approach," 2019. doi: 10.48550/arXiv.1910.06277.

[47] M. Aljabri *et al.*, "An assessment of lexical, network, and content-based features for detecting malicious URLs using machine learning and deep learning models," *Comput. Intell. Neurosci.*, vol. 2022, no. 1, pp. 1–14, 2022. doi: 10.1155/2022/3241216.

[48] C. Luo, S. Su, Y. Sun, Q. Tan, M. Han and Z. Tian, "A convolution-based system for malicious URLS detection," *Comput. Mater. Contin.*, vol. 62, no. 1, pp. 399–411, 2020. doi: 10.32604/cmc.2020.06507.

[49] A. K. Singh, "Malicious and Benign webpages dataset," *Data Brief*, vol. 32, pp. 106304, 2020. doi: 10.1016/j.dib.2020.106304.

[50] S. H. Ahammad *et al.*, "Phishing URL detection using machine learning methods," *Adv. Eng. Softw.*, vol. 173, pp. 103288, 2022. doi: 10.1016/j.advengsoft.2022.103288.

[51] T. Li, G. Kou, and Y. Peng, "Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods," *Inf. Syst.*, vol. 91, no. 3, pp. 101494, 2020. doi: 10.1016/j.is.2020.101494.

[52] Y. Liang, J. Kang, Z. Yu, B. Guo, X. Zheng and S. He, "Leverage temporal convolutional network for the representation learning of URLs," in *2019 IEEE Int. Conf. Intell. Secur. Inform., ISI 2019*, Shenzhen, China, 2019. doi: 10.1109/ISI.2019.8823362.

[53] H. Yuan, Z. Yang, X. Chen, Y. Li, and W. Liu, "URL2Vec: URL modeling with character embeddings for fast and accurate phishing website detection," in *Proc.—16th IEEE Int. Symp. Parallel Distrib. Process. with Appl. 17th IEEE Int. Conf. Ubiquitous Comput. Commun. 8th IEEE Int. Conf. Big Data Cloud Comput. 11t*, Melbourne, VIC, Australia, 2019. doi: 10.1109/BDCloud.2018.00050.

[54] M. K. Prabakaran, P. M. Sundaram, and A. D. Chandrasekar, "An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders," *IET Inf. Secur.*, vol. 17, no. 3, pp. 423–440, 2023. doi: 10.1049/ise2.12106.

[55] Y. Shi, K. Yang, Z. Yu, C. L. P. Chen, and H. Zeng, "Adaptive ensemble clustering with boosting BLS-based autoencoder," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12369–12383, 2023. doi: 10.1109/TKDE.2023.3271120.

[56] K. Yang, Y. Shi, Z. Yu, Q. Yang, A. K. Sangaiah, and H. Zeng, "Stacked one-class broad learning system for intrusion detection in industry 4.0," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 251–260, 2023. doi: 10.1109/TII.2022.3157727.

[57] E. Nowroozi, M. M. Abhishek, and M. Conti, "An adversarial attack analysis on malicious advertisement URL detection framework," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 2, pp. 1332–1344, 2023. doi: 10.1109/TNSM.2022.3225217.

[58] F. Sadique, R. Kaul, S. Badsha, and S. Sengupta, "An automated framework for real-time phishing URL detection," in *2020 10th Annu. Comput. Commun. Work. Conf. CCWC 2020*, Las Vegas, NV, USA, 2020. doi: 10.1109/CCWC47524.2020.9031269.

[59] A. K. Dutta, "Detecting phishing websites using machine learning technique," *PLoS One*, vol. 16, no. 10, pp. e0258361, 2021. doi: 10.1371/journal.pone.0258361.

[60] S. S. Shin, S. G. Ji, and S. S. Hong, "A heterogeneous machine learning ensemble framework for malicious webpage detection," *Appl. Sci.*, vol. 12, no. 23, pp. 12070, 2022. doi: 10.3390/app122312070.

[61] Q. Chen and K. Omote, "A three-step framework for detecting malicious URLs," in *2022 Int. Symp. Networks, Comput. Commun. ISNCC 2022*, Shenzhen, China, 2022. doi: 10.1109/ISNCC55209.2022.9851734.

[62] J. Yuan, Y. Liu, and L. Yu, "A novel approach for malicious URL detection based on the joint model," *Secur. Commun. Netw.*, vol. 2021, no. 6, pp. 1–12, 2021. doi: 10.1155/2021/4917016.

[63] R. Bharadwaj, A. Bhatia, L. D. Chhibbar, K. Tiwari, and A. Agrawal, "Is this URL Safe: Detection of malicious URLs using global vector for word representation," in *Int. Conf. Inf. Netw.*, Jeju-si, Korea, Republic of, 2022, vol. 12, pp. 486–491. doi: 10.1109/ICOIN53446.2022.9687204.

[64] K. M. Manjunatha and M. Kempanna, "Count vectorizer model based web application vulnerability detection using artificial intelligence approach," *J. Discret. Math. Sci. Cryptogr.*, vol. 25, no. 7, pp. 2039–2048, 2022. doi: 10.1080/09720529.2022.2133243.

[65] R. Rajalakshmi and C. Aravindan, "A Naive Bayes approach for URL classification with supervised feature selection and rejection framework," *Comput. Intell.*, vol. 34, no. 1, pp. 363–396, 2018. doi: 10.1111/coin.12158.

[66] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP 2014—2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, Doha, Qatar, 2014. doi: 10.3115/v1/d14-1162.