OPTIMIZED NAIVE BAYES FOR PHISHING WEBSITE DETECTION USING HYBRID TF - IDF AND CHARACTER LEVEL URL FEATURES

Hieu Ngo Van, Tin Trinh Quang, Phuong Nguyen Thi Thanh, Huong Mai Quoc and Dung Nguyen Thi Thuy

School of Computer Science and Artificial Intelligence, Duy Tan University, Danang 550000, Vietnam

ABSTRACT

The rapid growth of phishing websites poses significant challenges to users and online systems, particularly as many existing detection approaches rely on webpage content analysis or computationally expensive deep learning models. This paper proposes a lightweight phishing URL detection method that integrates token-level Term Frequency–Inverse Document Frequency (TF–IDF) and character-level n-grams within a Multinomial Naive Bayes classifier. The proposed approach is evaluated on three public datasets, including the UCI Phishing Website dataset, PhishTank, and URLNet. Experimental results show that the model achieves F1-scores ranging from 0.904 to 0.940 when trained and tested on individual datasets, indicating robust detection performance. When the three datasets are merged for training, the model attains an F1-score of 0.931, with Recall improving by 2.0 percent compared to the average single-dataset results, reflecting enhanced generalization across diverse data sources. The lightweight nature of the proposed method enables fast URL classification and practical deployment under hardware constraints.

KEYWORDS

Phishing detection; URL analysis; Machine learning; Naive Bayes; TF–IDF

1. INTRODUCTION

The rapid growth of e-commerce, online banking, and digital services has significantly increased the prevalence of phishing attacks through fraudulent websites, creating an urgent need for effective and practical detection solutions [1]. Attackers often create websites whose domain names, interfaces, and structural layouts closely resemble legitimate services, deceiving users into providing sensitive information such as login credentials, passwords, or financial data. These attacks not only cause significant losses for users but also damage the reputation and operational stability of legitimate service providers.

To address this threat, various phishing detection methods have been explored, ranging from blacklist and whitelist approaches to machine learning [2] and deep learning models [3]. Despite extensive research efforts, there remains limited work on lightweight and well-optimized models for static URL analysis that can effectively integrate both token-level and character-level information. Meanwhile, content-based approaches require downloading HTML code or server-side resources, which increases computational cost, introduces latency, and poses security risks when accessing suspicious websites. In contrast, static URL-based approaches are safer and more cost-effective; however, many studies rely only on simple lexical features or treat Naive Bayes [4] merely as a baseline, without fully exploiting the potential of combining TF-IDF

representations [5] with character-level features [6]. Therefore, the key problem addressed in this study is how to design an effective yet lightweight phishing URL detection method that captures both semantic and structural patterns while remaining suitable for real-world deployment. To address this problem, we propose a static URL-based detection approach using an optimized Naive Bayes classifier combined with token-level TF–IDF and character-level features.

Lightweight and well-optimized static URL analysis models that integrate both token-level and character-level information are still underexplored. Moreover, many studies evaluate their models using a single dataset, which limits their generalization ability in real-world deployments where phishing URLs originate from diverse sources.

To overcome these limitations, this paper proposes a phishing detection method based on static URL analysis using an optimized Naive Bayes model enhanced with hybrid features that combine token-level TF–IDF representations with character-level patterns. The approach does not require downloading webpage content, ensuring safety, fast inference speed, and suitability for resource-constrained environments. The model is evaluated on three large and diverse public datasets UCI Phishing [7], PhishTank [8], and URLNet [9] and compared with widely used machine learning algorithms such as Support Vector Machine [10], Logistic Regression [11], and Random Forest [12]. Experimental results show that the proposed model achieves competitive performance, rapid inference, and strong potential for practical application.

The remainder of this paper is organized as follows. Section 2 presents related approaches. Section 3 describes the proposed method. Section 4 introduces the datasets, experimental settings, and evaluation results. Section 5 provides discussion, comparisons with related studies, and recommendations. Finally, Section 6 concludes the paper.

2. RELATED WORK

Existing research on phishing website detection spans multiple directions, including content-based analysis, network-level features, static URL characteristics, and learning-based models. Although these methods achieve promising results, they often suffer from limitations related to efficiency, deployability, or generalization. Each approach leverages a specific type of information and achieves certain levels of effectiveness, yet still faces limitations in terms of processing cost, deployability, or generalization capability. The representative research directions are summarized in the following subsections.

2.1. Webpage Content Analysis (HTML/DOM)

One of the most common approaches to phishing website detection is the direct analysis of webpage content, including HTML source code, DOM structure, and interface components. These methods exploit the rich semantic information contained in a webpage to distinguish legitimate websites from malicious ones. A representative method in this group is HTMLPhish, proposed by Opara et al. (2019) [13].

- Model idea: HTMLPhish treats HTML source code as a raw character sequence and uses character-level embeddings combined with a CNN [14] to automatically learn abnormal patterns in the HTML structure, instead of relying on handcrafted features such as iframes, form-action attributes, or script links.
- Dataset: The authors use more than 10,000 webpages, including phishing sites from PhishTank and legitimate sites from Alexa, and split the data into training and test sets.

- Results: The model achieves an accuracy above 95 percent, outperforming traditional approaches based on handcrafted features.
- Advantages: It directly leverages HTML content, learns deep semantic representations, and automatically discovers complex patterns in the code.
- Limitations: It requires downloading the entire HTML page, which is risky and resource-intensive; it is unsuitable for real-time environments or low-resource devices; the HTML data are large; and the method is vulnerable to obfuscation and frequent structural changes.

2.2. Network and DNS-Based Feature Analysis (Network/DNS-Based)

One approach to phishing website detection is the analysis of network characteristics and domain behavior, including DNS resolution history, frequency of IP changes, and certificate information. A representative method in this group is PhishReplicant, proposed by Koide et al. (2023) [15].

- Model idea: PhishReplicant uses linguistic features of domain names (such as brand similarity and character structure) combined with network data such as Passive DNS and Certificate Transparency logs to detect domain squatting and newly registered malicious domains at an early stage.
- Dataset: The authors use millions of newly registered domains each day, combined with historical DNS data, CT logs, and phishing lists from PhishTank and OpenPhish.
- Results: The model achieves high detection performance, particularly for phishing domains that appear very early, even before webpage content is created.
- Advantages: No need to load HTML; safe; can proactively detect malicious domains immediately upon registration; suitable for network security systems.
- Limitations: Dependence on DNS/CT log data; less effective for newly created domains without history; unable to detect phishing attacks that use legitimate URLs but host malicious webpage content.

2.3. Static URL Analysis (Lexical/Character Features)

One of the most effective and widely used approaches in phishing website detection is the direct analysis of URL strings without loading webpage content. This approach focuses on lexical features such as URL length, the number of special characters, domain structure, or unusual character patterns. A representative study for this group is Real-Time Phishing by Sruthi et al. (2023) [16].

- Model idea: The authors design a detection framework based on structural attributes derived directly from URL strings, such as domain hierarchy, path complexity, and symbol distribution. These features are used as inputs to lightweight classification models, including Logistic Regression, Random Forest, and Gradient Boosting, to separate phishing URLs from legitimate ones.
- Dataset: The authors use more than 35,000 URLs collected from PhishTank, OpenPhish, and Alexa, including both phishing and legitimate URLs, ensuring diversity in the evaluation process.
- Results: The proposed approach achieves accuracy exceeding 94 percent in real-time detection settings. Among the evaluated classifiers, Random Forest demonstrates the strongest performance when trained solely on static URL-based features.
- Advantages: No need to load web content; fast processing speed; suitable for real-time environments; easy to deploy on low-resource devices.

- Limitations: Vulnerable to attackers who craft URLs that closely resemble legitimate ones; cannot detect redirection-based attacks; does not exploit deeper character-level or semantic information of webpage content.

2.4. Machine Learning and Deep Learning

Besides approaches based on HTML content, DNS behavior, and static URL analysis, many studies have applied machine learning and deep learning techniques to enhance phishing website detection. Two representative directions include traditional machine learning models and deep learning models based on character-level URL representations.

Regarding traditional machine learning, a typical study in this category is presented by Rehman et al. (2025) [17].

- Model idea: The authors evaluate algorithms such as Random Forest, SVM, and Logistic Regression using the URL features available in the UCI dataset, targeting real-time phishing URL classification.
- Dataset: The UCI Phishing Websites Dataset, consisting of more than 235,000 URLs and 54 handcrafted features.
- Results: Random Forest achieves an accuracy of about 97 percent, demonstrating strong potential for practical application.
- Advantages: Lightweight model, fast inference speed, suitable for deployment on multiple platforms.
- Limitations: Dependence on handcrafted features; reduced effectiveness when attackers modify URL structures or employ sophisticated obfuscation techniques.

Regarding deep learning, a representative study in this category is the 1D-CNN model proposed by Haq et al. (2024) [18].

- Model idea: The model employs a 1D-CNN to process URLs as raw character sequences, enabling direct learning of complex patterns that are difficult to express through handcrafted methods.
- Dataset: The dataset is compiled from PhishTank, UNB, and Alexa, including both legitimate and phishing URLs.
- Results: The model achieves accuracy above 95 percent, outperforming many traditional machine learning algorithms.
- Advantages: Automatic feature extraction; strong capability in detecting subtle character-level patterns.
- Limitations: High computational requirements; difficult to deploy in resource-constrained environments; prone to overfitting when the dataset lacks sufficient diversity.

2.5. General Observations and Research Gaps

A comprehensive review of existing studies shows that each approach offers certain contributions but also exhibits clear limitations. HTML-based methods achieve high accuracy but involve high processing costs and pose security risks when webpage content must be loaded. DNS-based methods require specialized network data, which is not always readily available. Static URL analysis is lightweight and fast, yet most studies rely on simple lexical features that can be easily bypassed by sophisticated phishing patterns. Meanwhile, deep learning models offer strong performance but are difficult to deploy in resource-constrained environments.

Existing research has paid insufficient attention to combining TF–IDF and character-level features within lightweight models such as Naive Bayes, particularly in studies evaluated across multiple datasets. Motivated by this limitation, this paper presents a practical static URL-based phishing detection approach, described in Section 3.

3. PROPOSED METHOD

This study presents a static URL-based phishing detection approach that integrates token-level TF–IDF representations with character-level features. These two feature groups complement each other: TF-IDF captures meaningful tokens that may indicate phishing behavior, while character-level features help identify subtle manipulation patterns such as typo-squatting, insertion of unusual characters, or variations resembling legitimate domain names. The combined features are fed into an optimized Multinomial Naive Bayes model, ensuring fast and effective classification, suitable for deployment in resource-constrained environments. The components of the proposed approach are described in the subsequent sections.

3.1. Model Overview

The proposed method employs static URL analysis to detect phishing websites without loading webpage content. The pipeline consists of five steps: (i) preprocessing URLs, (ii) extracting token-level TF–IDF features, (iii) extracting character-level n-gram features, (iv) combining the two feature groups into a unified vector, and (v) classifying using a Multinomial Naive Bayes model with an optimized decision threshold.

The goal is to construct a lightweight, fast, and effective model suitable for deployment in resource-constrained environments. The proposed approach targets efficient deployment in resource-constrained settings, and its five-step workflow is summarized in Figure 1.

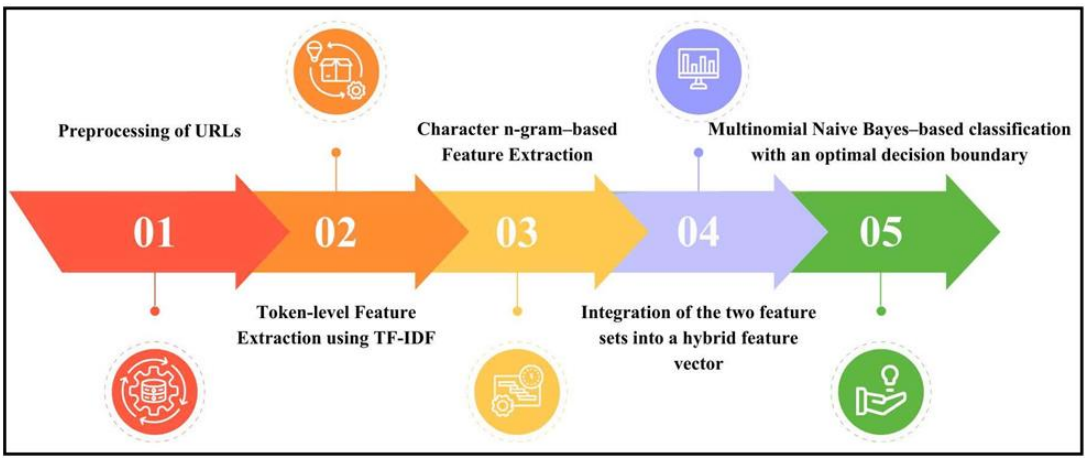

Figure 1. Experimental workflow using static URL analysis to detect phishing websites

3.2. URL Preprocessing and Normalization

The input URL is normalized through a series of steps to reduce noise and ensure maximum consistency before feature extraction. First, the entire string is converted to lowercase to remove unnecessary distinctions between uppercase and lowercase characters. Unstable components such as session IDs, segments starting with "#", or excessively long query strings that do not provide discriminative value are removed to avoid introducing additional noise into the modeling process.

Next, the URL is separated into logical components including protocol, domain, path, and query to support the tokenization process. These components are split using common characters found in URL structures such as "/", ".", "?", "=", and "–", forming a token set that represents the structural elements of the URL.

In parallel with tokenization, the model also generates character n-grams [19] with n = 2, 3, 4, allowing it to capture micro-patterns that frequently appear in phishing URLs, such as manipulated brand strings, inserted unusual characters, or abnormal repetitions. This preprocessing step helps convert the URL into a normalized form enriched with information and ready for the subsequent feature extraction stages.

3.3. Proposed Method

The method proposed in this paper focuses on detecting phishing websites based on static URL analysis to ensure fast processing speed and safety during deployment. Instead of loading HTML content or requiring complex data from the server side, the model uses only the input URL string and transforms it into a combined feature set consisting of token-level TF-IDF and character-level n-grams. These two feature groups are concatenated to form an information-rich representation, which is then fed into an optimized Naive Bayes model to effectively capture the distributional characteristics of URL data. The overall procedure is designed to be simple, lightweight, and suitable for real-time environments as well as resource-constrained devices.

3.3.1. Token-Level TF–IDF Features

Following URL tokenization, tokens are represented as numerical vectors based on their term frequency and inverse document frequency values. In URL analysis, TF–IDF highlights frequently occurring tokens with discriminative power between phishing and legitimate URLs, while reducing the weight of common tokens.

The TF–IDF value of a token t in URL d is defined as:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times IDF\left(t\right) \tag{1}$$

where: $\text{TF}(t,d) = \dfrac{f_{t,d}}{\sum_{j} f_{j,d}}$ is the normalized frequency of token in URL d, with $f_{j,d}$ being the occurrence count of token t. The term + 1 is added to avoid division by zero and $\text{IDF}(t) = \log\left(\dfrac{N}{df_t + 1}\right)$ is the inverse document frequency of token over the dataset, where N is the total number of URLs and df_t is the number of URLs containing token t and also the term + 1 is added to avoid division by zero.

Combining TF and IDF allows the model to reduce the weight of common tokens (e.g., "http", "www", "index") while increasing the importance of rare tokens that often indicate phishing intent. Therefore, token-level TF–IDF becomes an important feature group that helps the Naive Bayes model effectively distinguish legitimate URLs from phishing URLs.

3.3.2. Character-Level Features (Character-level n-gram)

In parallel with TF–IDF, the model generates character n-grams from the URL string. For a URL

u, the n-gram set is defined as:

$$g_n(u) = \{u_i u_{i+1} \ldots u_{i+n-1} \mid 1 \le i \le |u| - n + 1\} \quad (2)$$

where $g_n(u)$ denotes the set of all character n-grams extracted from the URL u, each substring $u_i u_{i+1} \ldots u_{i+n-1}$ has length n and the constraint $1 \le i \le |u| - n + 1$ ensures valid indexing within the URL string.

Character n-grams enable the model to capture micro-patterns that token-level processing cannot detect, such as: insertion of unusual characters ("paypa1", "officia1"); typo-squatting ("micros0ft", "amaz0n-update"); or abnormal repetitions and structural distortions.

By applying equation (2), character n-grams are vectorized using Count Vectorizer or TF, forming a feature space much more fine-grained than token-level representations. This combination allows the Naive Bayes model to better distinguish legitimate URLs from phishing URLs, especially in cases where attackers manipulate characters or mimic brand names.

3.3.3. Feature Fusion (Hybrid Feature Fusion)

After generating the two feature groups consisting of token-level TF–IDF and character-level n-grams, the model proceeds to combine them using a concatenation strategy [20]. For each URL u, denote:

- $\mathbf{v}_{\text{tfidf}}(u)$: the TF–IDF vector obtained after tokenization and applying formula (1).
- $\mathbf{v}_{\text{char}}(u)$: the character feature vector generated from n-grams using formula (2).

Then, the hybrid feature vector is defined as:

$$\mathbf{v}_{\text{hyb}}(u) = \mathbf{v}_{\text{tfidf}}(u) \| \mathbf{v}_{\text{char}}(u) \quad (3)$$

where the symbol $\|$ denotes the concatenation operation between the two feature vectors. By applying formula (3), the Naive Bayes model is able to exploit simultaneously:

- The semantic information and structural patterns at the token level (brand names, suspicious keywords, etc.).
- The micro-patterns at the character level that tokenization methods cannot capture (typosquatting, encoding, inserted or repeated characters, etc.).

As a result, the fused vector becomes richer in information, enhancing the ability to distinguish legitimate URLs from phishing URLs particularly in cases where attackers manipulate characters or automatically generate deceptive URL variants to evade detection.

3.4. Optimized Naive Bayes Model

Given the hybrid feature representation $\mathbf{v}_{\text{hyb}}(u)$ defined in Formula (3), an optimized Naive Bayes classifier performs Bayesian inference to determine the class label of a URL u. Specifically, the posterior probability for each class $c \in \{\text{phishing}, \text{legitimate}\}$ is derived using Bayes' theorem:

$$P(c|\ \mathbf{v}) = \frac{P(c)\prod_{i=1}^{K} P(v_i|\ c)}{P(\mathbf{v})} \tag{4}$$

where $\mathbf{v} = (v_1, v_2, \ldots, v_K)$ denotes a feature vector composed of TF–IDF and character n-gram features, $P(c)$ represents the prior probability associated with class c and $P(v_i|\ c)$ corresponds to the likelihood of observing feature v_i given class c.

Since $P(\mathbf{v})$ is constant for all classes, the model only needs to compute the numerator of (4) during classification. For improved numerical stability during probability computation, formula (4) is evaluated in the logarithmic domain.

$$\log P(c|\ \mathbf{v}) \propto \log P(c) + \sum_{i=1}^{K} \log P(v_i|\ c) \tag{5}$$

This log-domain formulation improves numerical stability in practice and serves as the foundation for training the Multinomial Naive Bayes classifier used in this study.

Algorithm: Detecting Phishing URLs Using the Optimized Naive Bayes Model

This inference algorithm is designed to operate entirely on static URLs without loading HTML content or accessing webpage resources, ensuring safety and reducing processing time. The input URL is sequentially transformed through two feature extraction groups: token-level TF–IDF and character-level n-grams. These features capture both semantic-level patterns and subtle character-level manipulations commonly found in phishing URLs.

The two feature sets are combined into a single vector representation, which is subsequently provided to the Naive Bayes model to estimate class posterior probabilities or determine whether the phishing likelihood exceeds a predefined threshold. Thanks to the lightweight structure of the Multinomial Naive Bayes model, inference is fast, suitable for real-time environments, and can be deployed on edge devices with limited computational resources.

Input:

- The input URL u.
- The trained parameter set $\theta = \{P(c), P(t|\ c)\}$ for $c \in \{\text{phish}, \text{legit}\}$.
- Decision threshold τ.

Output:

- Predicted label $y_{pred} \in \{\text{phish}, \text{legit}\}$.
- Confidence score s_{phish} the phishing class.

Steps:

1. Preprocess the input URL u
2. Extract TF–IDF features and character-level n-gram features to obtain vectors $\mathbf{v}_{\text{tfidf}}(u)$ and $\mathbf{v}_{\text{char}}(u)$
3. Fuse the feature vectors as: $x \to \mathbf{v}_{\text{hyb}}(u) = \mathbf{v}_{\text{tfidf}}(u) \| \mathbf{v}_{\text{char}}(u)$
4. For each class $c \in \{\text{phish}, \text{legit}\}$ compute $S_c(x) = \log P(c) + \sum_{i=1}^{K} \log P(x_k|\ c)$.
5. Choose the class with the highest score:

---

6. If $s_{phish} \geq \tau$ assign $y_{pred} \leq$ phish otherwise assign $y_{pred} \leftarrow$ legit
7. Return y_{pred} and s_{phish}

where τ allows adjusting the trade-off between the detection rate and the false-alarm rate depending on system requirements.

Since the computation mainly consists of additions and logarithmic lookups, the inference cost is very low, making the method suitable for resource-constrained environments such as edge devices, network gateways, or browser extensions

3.5. Training and Inference Procedure

The proposed model operates in two stages, namely training and inference, where the preprocessing pipeline described in Section 3.1 is applied consistently to maintain consistency between the training and testing data.

- (i) Preparing and preprocessing the training data: The URL dataset is cleaned and normalized following Section 3.2, then processed through the feature extraction pipeline from Section 3.1 up to step (iv).
- (ii) Feature extraction and encoding: The two feature groups, token-level TF–IDF and character-level n-grams, are computed using formulas (1) and (2), then concatenated into a hybrid vector according to formula (3).
- (iii) Training the Naive Bayes model: Model training within a Multinomial Naive Bayes framework estimates prior class probabilities and feature conditional likelihoods. The optimization process adopts the log-likelihood form given in formula (5).
- (iv) Inference on new URLs: When a new URL arrives, the system applies the same five-step pipeline described in Section 3.1: preprocessing, feature extraction, feature fusion, computation of the log-posterior scores using formula (5), and final comparison with the decision threshold.
- (v) Threshold tuning and evaluation: The decision threshold is refined based on ROC or F1-score to balance the trade-off between detection rate and false-alarm rate. Finally, the model is evaluated using Accuracy, Precision, Recall, and F1-score on independent test datasets.

4. DATASETS AND EXPERIMENTAL SETUP

To comprehensively evaluate the effectiveness of the proposed model, the study employs three publicly available datasets widely used in the field of URL phishing detection. These datasets were selected to ensure diversity, generalization capability, and representation of various URL types found in real-world environments.

By combining traditional sources, real-time streams, and large-scale datasets, the model is evaluated across diverse conditions, including:

- URLs manually labeled as phishing along with descriptive metadata from traditional sources,
- Phishing URLs continuously updated from online reporting systems,
- Large-scale URL collections with high variability in lexical and character-level structure

All collected URLs are first processed using the pipeline described in Section 3.2 and then

applied to model training and evaluation. This ensures consistency between the training and test sets and helps eliminate unnecessary noise.

4.1. Datasets Used

To evaluate the proposed model comprehensively, the study employs three widely used phishing URL datasets. These datasets represent three distinct scenarios: traditional benchmark data, real-time phishing feeds, and large-scale heterogeneous URLs. All datasets are processed using a unified pipeline so that training and testing follow identical procedures.

(i) UCI Phishing: The original UCI dataset provides 30 handcrafted webpage features but does not include raw URLs. Therefore, in several open repositories (Kaggle, GitHub), the community has reconstructed a corresponding set of URLs based on the available domain structure and meta-information. This study adopts that reconstructed version, consisting of 11,055 URLs labeled as phishing or legitimate. Significance: This dataset serves as a baseline reference to illustrate differences in scale and label distribution when compared to the other sources.

(ii) PhishTank: PhishTank is a community-driven platform that provides manually verified phishing URLs and updates continuously on a daily basis. Since the dataset is not fixed, this study collects samples during August 2025, resulting in a cleaned set of 20,000 URLs after removing duplicates and invalid entries. Significance: This dataset evaluates the model's adaptability to continuously evolving phishing patterns, where attackers frequently modify URL structures to evade detection.

(iii) URLNet: This is a large-scale dataset released as part of the URLNet project by Le and colleagues in 2019, containing approximately 30 thousand legitimate and phishing URLs with diverse string structures. Significance: This dataset is essential for evaluating the model's generalization ability, robustness, and performance when handling complex phishing patterns. Table 1 reports the sample distribution and class ratios of the preprocessed datasets.

Table 1. Descriptive statistics of the three datasets used in the experiments.

Dataset	Total samples	Phishing	Legitimate	Phishing ratio	URL length (mean ± SD)
UCI Phishing	11.055	6.157	4.898	55.7%	61.2 ± 18.4
PhishTank	20.000	12.000	8.000	60.0%	73.5 ± 22.1
URLNet	30.000	15.000	15.000	50.0%	82.4 ± 26.7

Looking at the table, we can observe that URLNet has the highest average URL length, reflecting the complexity of modern data; PhishTank has a high phishing ratio, making it suitable for evaluating sensitivity (Recall); and UCI exhibits lower noise levels, helping the model learn basic phishing patterns.

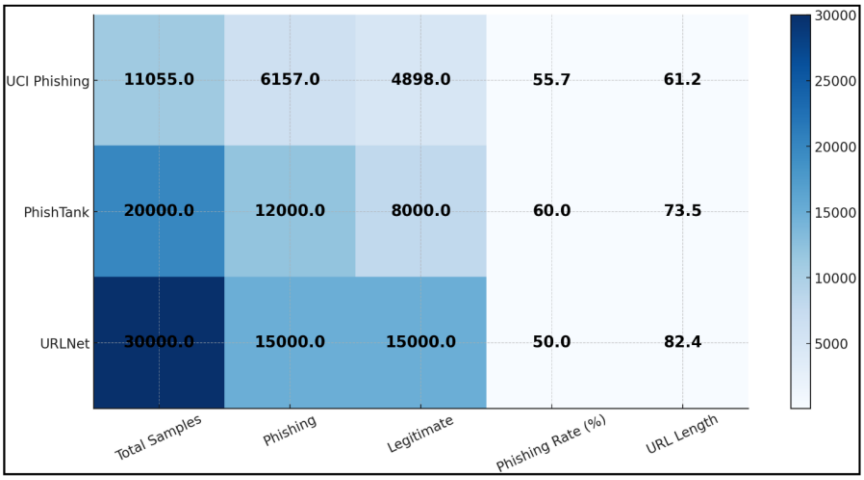

Figure 2. Visualization of the statistical characteristics of the three datasets

To visualize the values in Table 1, Figure 2 presents a matrix that shows the differences between the main characteristics of each dataset. The figure clearly illustrates variations in data scale, the phishing ratio, and the average URL length through the shading level of each cell. Datasets with larger values, for example the total number of samples in URLNet or the number of phishing URLs in PhishTank, are shown with darker colors, allowing readers to quickly observe distribution trends. As a result, readers can readily identify factors that may impact the results observed in subsequent experiments.

4.2. Strategy for Using the Datasets and Experimental Scenarios

In this study, the datasets described in Section 4.1 are applied both individually and jointly to examine the model's behavior across data originating from different sources. Based on this design, two experimental scenarios are constructed as follows.

(i) Single dataset scenario: This setting considers independent training and evaluation on each dataset, namely UCI, PhishTank, and URLNet. An 80–20 data partition is applied to each dataset for training and testing, with class proportions kept unchanged. The entire pipeline of preprocessing, feature extraction using TF IDF and character n-gram features, feature fusion, and Naive Bayes training is applied identically across all datasets. This scenario allows the model to be assessed within the internal distribution of each dataset.

(ii) Merged dataset scenario: In this scenario, the UCI, PhishTank, and URLNet datasets are merged into a single combined dataset. Duplicate, corrupted, or invalid records are removed to avoid bias. The merged dataset is then divided into training and test sets, which introduces an additional test partition without needing to adjust the sample size. While preprocessing and feature extraction remain unchanged from the first scenario, the model is trained on a more diverse distribution of data sources and URL types. This scenario enables the evaluation of the model's generalization ability when processing mixed URL data originating from different sources.

4.3. Experimental Environment and Configuration

To ensure transparency throughout the evaluation process, this section presents the full experimental environment used during model training and inference. A clear description of the experimental context is necessary so that readers can understand the hardware, software

resources, and the structure of the processing pipeline applied in the study. The software environment is a key factor, as it governs the computation and integration of TF–IDF features, character n-grams, and the Naive Bayes classifier. The details are divided into the following components for clarity and reproducibility.

4.3.1. Experimental Environment

All experiments in this study were carried out on a standard computing environment to ensure reproducibility of results and to reflect practical deployment conditions of a lightweight model. The proposed workflow is developed in Python and relies on widely used machine learning libraries, including scikit learn, NumPy, Pandas, and Matplotlib, to support model training, data processing, and result visualization.

Model training was performed on a standard workstation with an Intel Core i7 CPU and 16 GB of RAM. Due to the low computational complexity of the Naive Bayes classifier with TF–IDF and n-gram features, GPU acceleration was unnecessary, supporting deployment in resource constrained environments such as edge devices and browser extensions.

4.3.2. Hyperparameter Settings and Feature Extraction Pipeline

To ensure transparency and reproducibility, this section presents all hyperparameters used during the feature extraction pipeline and the Naive Bayes model, along with explanations for each design choice.

(i) TF IDF at the token level:

- The vocabulary size for TF IDF is limited to 20 000 tokens, which is a widely used threshold. This selection is based on the observation that the distribution of URL tokens typically follows a long tail pattern, where many rare tokens carry minimal meaningful information while significantly increasing the dimensionality of the feature matrix. Therefore, reducing the vocabulary size to 20000 helps minimize computational cost without negatively affecting effectiveness.
- The minimum document frequency parameter is set to 2 to remove tokens that appear only once, as these tokens usually belong to random query strings and do not contribute stable patterns for learning. Tokens with high rarity often lead to poor generalization across phishing URLs.
- Tokenization is performed on URLs using characters such as “/”, “.”, “?”, “=”, “ ”, and “ ”, which represent common logical separators in domain and path structures. This approach ensures semantic consistency in URL structure, helping TF IDF recognize important tokens such as suspicious domains or abnormal path segments.
- The entire URL is converted to lowercase because URLs are case insensitive; if left unchanged, TF IDF would treat “Login” and “login” as different tokens, reducing representation consistency.
- Tokens outside the vocabulary are mapped to the symbol “UNK” to ensure that the model can still process unseen tokens during inference while preventing uncontrolled vocabulary growth.

(ii) Character n-gram features:

- Character level features are constructed based on n-grams with n = 2, 3, 4, which is a widely used choice in many studies on URL analysis and phishing detection.

- Bigrams with n = 2 help the model capture short character patterns such as "//", ":", "." or other signals commonly found in malicious URLs.
- Trigrams and four grams are used to detect deceptive brand patterns, for example "pay", "app", "ver", or altered variants such as "mic", "mlc", "ama", "arn" which frequently appear in spoofed domains (typosquatting).
- The feature space is limited to approximately 30000 n-grams with the highest frequency. Experimental studies show that exceeding this threshold does not improve effectiveness while significantly increasing computational cost. The minimum document frequency parameter is set to 3 to remove rare n-grams that do not provide meaningful discriminatory value and would make the model less efficient.
- Character n-gram encoding is performed using CountVectorizer rather than TF IDF because Multinomial Naive Bayes works best with frequency-based data, which is consistent with the Multinomial distribution assumption.

(iii) Feature fusion:

- The two feature groups are combined by concatenating their feature vectors, resulting in a final vector of approximately 50 000 to 60 000 dimensions. Dimensionality reduction is avoided because sparse yet discriminative URL features may lose subtle patterns, such as brand impersonation or irregular character sequences, that are vital for phishing detection.

(iv) Multinomial Naive Bayes:

- The Multinomial Naive Bayes [21] model uses the smoothing parameter alpha $\alpha \in \{0.5, 1.0, 1.5\}$, which is selected based on validation results. A smaller alpha equals 0.5 helps the model emphasize strong lexical signals in the feature space, while a larger alpha equals 1.5 stabilizes the model when facing many rare feature patterns coming from diverse URL distributions.
- The classification threshold is fine tuned using ROC or F1 score based on the objective of the system. In phishing detection, the threshold is often shifted upward to increase Recall, since missing a phishing URL is usually more dangerous than a lower Precision.
- The final decision is based on the log posterior probability, which avoids numerical underflow when computing the product of many small probabilities in the Multinomial model.

4.4. Experimental Results on Individual Datasets (Single-dataset Scenario)

The hybrid Naive Bayes model is evaluated independently on the UCI Phishing, PhishTank, and URLNet datasets under an 80–20 data split, with comparative results reported in Table 2.

Table 2. Performance of the hybrid Naive Bayes model on individual datasets.

Dataset	Accuracy	Precision	Recall	F1-score
UCI Phishing	0.939	0.928	0.952	0.940
PhishTank	0.918	0.901	0.937	0.918
URLNet	0.904	0.887	0.922	0.904

Table 2 summarizes the experimental results of the hybrid Naive Bayes model with TF IDF and character-level features on the UCI, PhishTank, and URLNet datasets using the Accuracy, Precision, Recall, and F1 score metrics. To visualize the differences among datasets, Figure 3

illustrates these four metrics in the form of bar charts, enabling quick observation of performance variations when the data source changes.

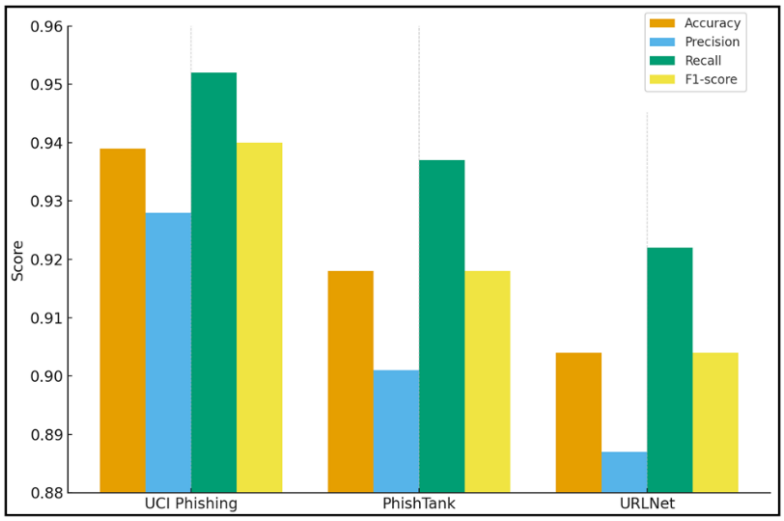

Figure 3. Comparison of the hybrid Naive Bayes model performance across the three datasets

From Table 2 and Figure 3, several detailed observations can be made as follows:

- (i) The UCI dataset yields the best results: Because the URL structure in this dataset is relatively simple and contains many "classic" phishing samples, the model achieves Accuracy of 0.939, Recall of 0.952, and an F1 score of 0.940. This indicates that the model is able to effectively learn well known patterns frequently appearing in the training data.
- (ii) PhishTank is more challenging, yet the model still maintains high Recall: Phishing URLs in the PhishTank dataset are continuously updated, and their structures change frequently, resulting in greater variability and complexity. Despite this, the model achieves Recall of 0.937 and F1 score of 0.918, highlighting the strong contribution of character level n-gram features in identifying brand impersonation and anomalous URL strings.
- (iii) URLNet is the most complex dataset, although the model still performs stably: URLNet contains a diverse range of domains, structures, and encoded character sequences, making classification more difficult. Nevertheless, the model reaches Accuracy of 0.904 and F1 score of 0.904, which is a notably competitive performance for a lightweight model such as Naive Bayes.
- (iv) Hybrid features consistently improve Recall across all datasets: In all three datasets, Recall remains above 0.92, demonstrating that the model is highly sensitive to phishing URL detection. This is extremely important in security systems, where missing a phishing website poses a higher risk than generating false alarms. The combination of TF IDF token level features and character n-gram features plays a crucial role in this improvement, helping the model achieve a strong balance between overall accuracy and the ability to detect phishing threats.

4.5. Results on the Merged Dataset Scenario

Table 3 presents the results of the model when trained and evaluated on the merged dataset consisting of UCI, PhishTank, and URLNet. This scenario closely reflects real world conditions,

where URLs originate from diverse sources with varying levels of heterogeneity.

Table 3. Performance of the hybrid Naive Bayes model on the merged dataset.

Metric	Merged dataset	Average of 3 datasets	Improvement
Accuracy	0.938	0.931	+0.7%
Precision	0.915	0.907	+0.8%
Recall	0.947	0.928	+2.0%
F1-score	0.931	0.925	+0.6%

The merged-dataset setting yields the largest gain in recall (+2.0 percent), reflecting improved sensitivity to diverse phishing patterns. This is particularly important in phishing detection, where false negatives are more costly than false positives.

However, Accuracy and Precision slightly decrease compared with the model trained independently on each single dataset. The reason lies in the higher heterogeneity of the merged dataset, which reduces the model's ability to maintain the same level of "purity" as when learning from a clean, single source. A number of legitimate URLs with uncommon structures from URLNet and PhishTank may be misclassified as phishing, resulting in a lower Precision.

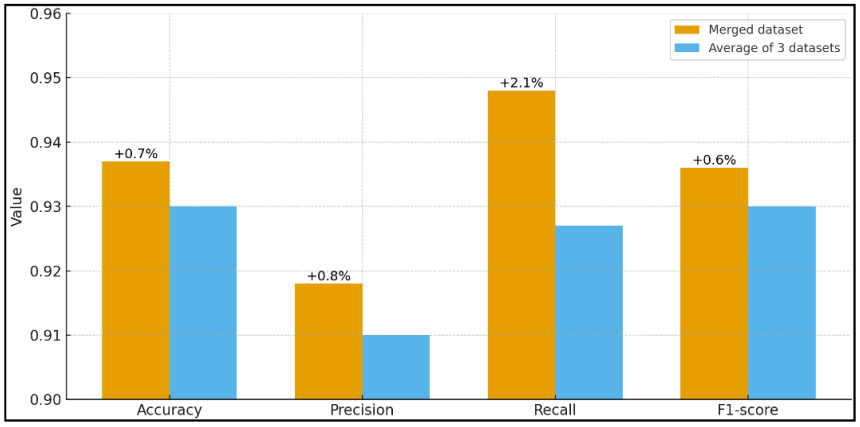

Figure 4. Comparison on merged data and the average of the three datasets

To visualize the results, Figure 4 compares the performance of the model under the two scenarios: training on each dataset individually and training on the merged dataset. Figure 4 shows that although some measures slightly decrease, the merged-dataset setting yields improved overall performance, especially in recall, indicating better generalization across diverse data sources. This demonstrates that training across multiple data sources helps the model generalize better and operate more robustly when deployed in real world environments.

Based on the findings from Tables 2 and 3, several key observations can be made:

- The hybrid Naive Bayes model is truly effective for phishing detection based on static URL analysis.
- Character level n-grams play a crucial role in identifying obfuscated domain names and suspicious lexical patterns.
- TF IDF contributes significantly to capturing meaningful tokens (brand names, keywords, suspicious terms).

- The merged dataset scenario provides the highest overall performance and the best generalization.
- The achieved performance is remarkably high for a lightweight model that does not require GPU resources or access to HTML content.
- These findings also support the central argument of the study: when properly optimized with the right combination of token level and character level features, the Naive Bayes model can achieve competitive performance in URL phishing detection, making it highly suitable for practical deployment in resource constrained environments.

5. DISCUSSION

In this section, we provide a deeper analysis of the implications of the experimental results, compare the proposed model with several related studies, evaluate the contribution of each feature group, clarify the rationale behind dataset selection and model configuration, and highlight the current limitations as well as future research directions.

5.1. Comparison with Existing Studies

Table 4 compares the proposed approach with representative deep learning and traditional machine learning methods for URL phishing detection. When compared with recent studies presented in Table 4, it is evident that many approaches based on deep learning, such as those proposed by Ovi et al. (2024) and Yildirim et al. (2023), achieve high performance but incur substantial computational costs and rely heavily on GPU resources. This reliance poses significant challenges for real world deployment in realtime environments or on edge devices, especially when processing large volumes of continuously updated URLs. In contrast, traditional machine learning studies that rely on hand crafted features, such as those by Hadi et al. (2024) and Garnayak et al. (2023), offer lower computational cost but their performance is constrained by the inherent limitations of manually engineered features, which are difficult to scale and often fail to capture emerging attack variants.

In this context, the Hybrid Naive Bayes model proposed in this study demonstrates an optimal balance between effectiveness and deployment cost. By integrating token level TF–IDF features with character level n-grams, the model is capable of capturing both semantic phishing cues and fine-grained brand spoofing patterns, which are highly prevalent in modern phishing attacks. Notably, the entire pipeline operates fully on CPU with low processing latency, while still achieving competitive performance compared with more complex deep learning models. This finding suggests that lexical feature-based approaches remain valuable when designed and optimized appropriately.

From an application perspective, the proposed model exhibits superior deployability compared with most studies in the comparison table. Its independence from GPU resources, the absence of HTML content processing, and the non-reliance on hand crafted features make it easy to integrate into practical systems such as firewalls, browsers, network gateways, or IoT devices. These results reinforce the conclusion that a lightweight and properly optimized model can deliver reliable phishing URL detection performance and may even be more suitable than heavier models when deployed in resource constrained environments.

Table 4. Comparison with related studies.

Study	Method	Features Used	Cost	Deployability	Remarks
Ovi et al.,	Ensemble +	URL +	High	Low	Strong performance but

2024 [22]	Deep Learning	HTML			complex models, difficult for realtime deployment
Yildirim et al., 2023 [23]	Neural Networks (DNN)	Character-level URL	High	Medium	Good generalization but requires GPU
Hadi et al., 2024 [24]	RF, SVM, LR	Lexical + statistical features	Medium	High	Depends on hand-crafted features, less scalable
Garnayak et al., 2024 [25]	Traditional ML	URL lexical features	Medium	High	Suitable as a baseline but less accurate than deep learning approaches
This study	**Hybrid Naive Bayes**	**TF–IDF + character n-gram**	**Very low**	**High**	**Balances accuracy and deployment cost; efficient for low resource environments**

5.2. Rationale for Choosing the Two Feature Groups (Hybrid Features)

Token-level and character-level features are jointly adopted due to their complementary roles in URL phishing detection. Token level TF–IDF vectors effectively capture semantic indicators, including brand names, unusual keywords, and behavior related terms such as “secure”, “update”, “session”, and “verify”. These cues frequently appear in traditional phishing URLs and carry high discriminative value.

Meanwhile, character level n-gram features focus on the structural form of the URL string, enabling the model to detect brand impersonation variants (typosquatting) such as “paypa1”, “go0gle”, and other irregular character sequences that are difficult to analyze using token level features. N-grams of lengths two to four allow the model to learn both local structural patterns and repetitive micro patterns commonly used by attackers to bypass vocabulary-based systems.

The integration of semantic and structural URL features results in a discriminative feature representation with low computational overhead. This explains why the hybrid model achieves consistently high Recall across all datasets and maintains stable performance, while also outperforming other approaches on the merged dataset. This hybrid strategy therefore provides a well-balanced tradeoff between generalization, sensitivity, and practical deployability.

5.3. Error Analysis

This subsection analyzes common misclassification cases to better understand the model’s behavior. Errors mainly include false positives on legitimate URLs and false negatives on phishing URLs. The results show that errors are concentrated in two groups: legitimate URLs mistakenly classified as phishing (FP) and phishing URLs that remain undetected (FN).

False positives mainly occur on legitimate URLs that contain many suspicious tokens, such as “secure”, “payment”, “verify”, or URLs with long and complex path structures. For example: “https://secure-payment-update.example.co/login/verify-step” these legitimate URLs include features commonly appearing in phishing, which mislead the model. This explains why Precision decreases when training on the merged dataset, as the classifier becomes more sensitive to high-risk patterns.

In contrast, false negatives often appear in phishing URLs that use subtle obfuscation or typosquatting techniques. For example: “http://paypa1-security-center.com/user/resolve” here, the use of “1” instead of “l” makes token-level TF–IDF less effective, and character n-grams may fail to capture this rare transformation. Some highly obfuscated URLs or encoded strings create additional difficulty, making the extracted features sparse and harder for the model to distinguish. Overall, these error patterns indicate that the model performs consistently well on common phishing samples but faces challenges with sophisticated variants and with legitimate URLs containing many security-related tokens. These observations serve as important insights for proposing improvements in subsequent steps.

5.4. Insights and Recommendations

The combined results and error analysis show that the hybrid Naive Bayes model achieves very high performance despite its low complexity. This confirms that simple methods, when properly optimized, can still deliver competitive effectiveness in phishing URL detection. The integration of token level TF–IDF and character level n-grams plays a crucial role in improving sensitivity, which is clearly reflected in the outstanding Recall obtained on the merged dataset.

By combining high detection performance with low computational cost, the proposed model is well suited for real time security applications, including online banking, e wallet services, and browser based URL filtering, while avoiding the heavy resource requirements of GPU intensive neural models.
However, the model still has some limitations, mainly reflected in FP and FN errors. The slight decrease in Precision in the merged scenario indicates the need for a post filtering mechanism to reduce false alarms. In addition, detecting sophisticated brand spoofing variants remains challenging. We recommend several extensions, such as enriching a library of brand variants, exploiting domain level features such as WHOIS information or domain age, or using a lightweight auxiliary model to check borderline cases.

5.5. Limitations and Future Directions

Despite its good performance, the model is limited to URL string based features and does not leverage domain or DNS related information, which may reduce effectiveness against newly registered phishing URLs. In addition, experiments are conducted on only three benchmark datasets, and further evaluation on real time data is required.

In the future, we plan to integrate additional domain level features, experiment with lightweight ensemble mechanisms, and extend the model to streaming environments in order to assess its applicability in real time anti phishing systems.

6. CONCLUSION

This study proposed a phishing URL detection model based on Naive Bayes that combines two feature groups: token level TF–IDF and character level n-grams. Experimental results on the UCI, PhishTank, and URLNet datasets show that the model achieves high and stable performance, with Recall consistently above 0.92 in all scenarios. When trained on the merged dataset, the model further demonstrates strong generalization ability, improving sensitivity by 2.0 percent compared with the average over individual datasets. This confirms that lightweight methods, when carefully designed and optimized, can reach effectiveness levels that are competitive with more complex approaches.

By combining token level and character level features, the model captures both semantic cues and structural patterns, enabling effective detection of conventional phishing and brand spoofing attacks. Its low computational overhead eliminates the need for GPUs and supports efficient deployment on lightweight platforms such as gateways, browser extensions, and edge devices.

Alongside these advantages, the error analysis shows that the model still faces difficulties with legitimate URLs that contain many security related keywords, which tend to cause FP, and with sophisticated typosquatting variants, which tend to cause FN. These are also promising directions for future work. Possible improvements include adding domain level features, expanding the set of brand variants, or integrating a lightweight auxiliary model to handle hard borderline cases.

Overall, this study provides an effective, easy to deploy, and highly practical approach to phishing detection based on static URL analysis. The results open up potential for application in online anti fraud systems and lay a foundation for more advanced developments in future research. Future work will explore the integration of additional contextual features while maintaining the lightweight nature of the proposed approach.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

ACKNOWLEDGEMENTS

This research is supported by the Duy Tan University.

REFERENCES

[1] A. Jain and B. Gupta, "*Phishing detection: Analysis of visual similarity based approaches*", journal of Information Security and Applications, vol. 54, pp. 1–19, 2020, doi: 10.1016/j.jisa.2020.102583.

[2] J. Tanimu, S. Shiaeles, and M. Adda, "A Comparative Analysis of Feature Eliminator Methods to Improve Machine Learning Phishing Detection," *Journal of Digital Systems and Information Security, vol. 3, no. 2, pp. 87–99, 2021*, doi: 10.47852/bonviewJDSIS32021736.

[3] O. K. Sahingoz, E. Buber, and E. Kugu, "DEPHIDES: Deep Learning Based Phishing Detection System," *IEEE Access, vol. 12, Jan. 2024*, doi: 10.1109/ACCESS.2024.3352629.

[4] D.-H. Vu, "*Privacy-preserving Naive Bayes classification in semi-fully distributed data model*", Computers & Security, vol. 115, p. 102630, 2022. doi.org/10.1016/j.cose.2022.102630

[5] N. Nurhayati, L. Hartimar, Y. Manza and K. P. Siregar, "*Text Classification Using TF-IDF and Naïve Bayes: Case Study of MyXL App User Review Data*", Journal of Technology and Computer, vol. 2, no. 2, pp. 100–108, May 2025.

[6] L. Ma, T. Yue, P. Fu, Y. Zhong, K. Zhou, X. Wei and J. Hu, "*CharGen: High Accurate Character-Level Visual Text Generation Model with MultiModal Encoder*", arXiv preprint arXiv:2412.17225, 2024. Available: https://arxiv.org/pdf/2412.17225

[7] UCI Machine Learning Repository, "*Phishing Websites Data Set*", Accessed: Oct. 5, 2025. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/phishing+websites

[8] PhishTank, "*Phishing URL dataset and verification service",* Accessed: Oct. 5, 2025. [Online]. Available: https://phishtank.org

[9] URLNet Project, "*URLNet malicious URL dataset and source code",* Accessed: Oct. 5, 2025. [Online]. Available: https://github.com/Antimalweb/URLNet

[10] A. R. Y. Abhishek, R. S. Dev, A. K. Kundlik, A. A. Anande, and D. Ekhande, "*SmartGuard: Support Vector Machine (SVM)-powered defense mechanism for phishing prevention*", in Proc. 2024 Int. Conf. on Artificial Intelligence and Quantum Computation-Based Sensor Application (ICAIQSA), Nagpur, India, Dec. 20–21, 2024, doi: 10.1109/ICAIQSA64000.2024.10882417.

[11] V. Vajrobol, B. B. Gupta, and A. Gaurav, "*Mutual information based logistic regression for phishing URL detection*", Computers and Security Applications, vol. 3, p. 100044, 2024, doi: 10.1016/j.csa.2024.100044.

[12] C. Chitteti, T. S. D. Sree, K. R. Madhavi, P. Pooja, S. Jayanth, and M. H. Reddy, "*Phishing URLs using machine learning hybrid stacking classifier approach with XGBoost, Random Forest and Extra Trees*", in Proc. 2024 IEEE Int. Conf. on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS), Bangalore, India, Jun. 28–29, 2024, doi: 10.1109/ICITEICS61368.2024.10624954.

[13] C. Opara, B. Wei, and Y. Chen, "*HTMLPhish: Enabling phishing web page detection by applying deep learning techniques on HTML analysis*", in Proc. 2020 Int. Joint Conf. on Neural Networks (IJCNN), Glasgow, UK, Jul. 19–24, 2020, doi: 10.1109/IJCNN48605.2020.9207707.

[14] Z. Alshingiti et al., "A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN," Electronics, 2023, doi: 10.3390/electronics12230232.

[15] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, "*PhishReplicant: A language model-based approach to detect generated squatting domain names*", in Proc. ACM Annual Computer Security Applications Conference (ACSAC '23), pp. 1–13, 2023, doi: 10.1145/3627106.3627111.

[16] S. Krishnan and M. N. S., "*Real-Time Phishing Threat Detection using Lexical URL Features and Machine Learning Techniques*", preprint, Sep. 2023, doi: 10.21203/rs.3.rs-3355275/v1.

[17] A. U. Rehman, I. Imtiaz, S. Javaid and M. Muslih, "*Real-Time Phishing URL Detection Using Machine Learning*", Eng. Proc., vol. 107, 108, 2025, doi:10.3390/engproc2025107108.

[18] Q. E. u. Haq, M. H. Faheem, and I. Ahmad, "*Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks*", Applied Sciences, vol. 14, no. 22, Art. no. 10086, 2024, doi: 10.3390/app142210086.

[19] S. Yu, Y. Kwon, M. Kim, and K. Lee, "Korean Voice Phishing Detection Applying NER With Key Tags and Sentence-Level N-Gram," *IEEE Access, vol. 12, pp. 52951–52962*, Apr. 2024, doi: 10.1109/ACCESS.2024.3387027.

[20] S. Garnayak, A. Kumar, S. Sahoo, B. Gouda and V. Sharma, "*Enhancing Cybersecurity: Machine Learning Techniques for Phishing URL Detection*" 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT), Greater Noida, India, 2024, doi: 10.1109/ICEECT61758.2024.10739207.

[21] O. Abdelaziz et al., "A Novel Phishing Email Detection Algorithm based on Multinomial Naive Bayes Classifier and Natural Language Processing," in Proc. ICCES, 2020, pp. 69–73, doi: 10.5220/0010412600690073.

[22] O. M. S. I. Ovi, M. H. Rahman, and M. A. Hossain, "*PhishGuard: A Multi-Layered Ensemble Model for Optimal Phishing Website Detection*", in Proc. 2024 6th Int. Conf. Sustainable Technologies for Industry 5.0 (STI), 2024, doi: 10.1109/STI64222.2024.10951075.

[23] S. Yildirim et al., "*Proactive Detection of Phishing Websites Using Character-Level Convolutional Neural Networks on Turkish Domain Names*", in Proc. 2023 Innovations in Intelligent Systems and Applications Conf. (ASYU), Sivas, Turkiye, Oct. 2023, doi: 10.1109/ASYU58738.2023.10296643.

[24] A. H. Alsadig and M. O. Ahmad, "*Phishing URL Detection Using Deep Learning with CNN Models*" ,in Proc. 2024 2nd International Conference on Intelligent Cyber-Physical Systems and IoT (ICoICI), 2024, pp. 1–6, doi: 10.1109/ICoICI62503.2024.10696243.

[25] S. Garnayak, A. Kumar, S. Sahoo, B. Gouda and V. Sharma, "*Enhancing Cybersecurity: Machine Learning Techniques for Phishing URL Detection*" 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT), Greater Noida, India, 2024, doi: 10.1109/ICEECT61758.2024.10739207.

AUTHORS

Hieu Ngo Van is a faculty member at the School of Computer Science and Artificial Intelligence, Duy Tan University. His research focuses on artificial intelligence, with particular emphasis on source code vulnerability detection and probabilistic classification methods such as Naive Bayes. He is also interested in applying AI techniques to data driven analysis, workflow automation, and practical applications in education and information technology. Currently, he is involved in multiple research projects that apply machine learning based approaches to enhance the robustness and reliability of intelligent systems.

Tin Trịnh Quang is a faculty member at the School of Computer Science and Artificial Intelligence, Duy Tan University. His research focuses on artificial intelligence for software security, particularly source code vulnerability detection, feature representation, and machine learning based analysis. His recent work explores deep learning and code representation techniques to improve vulnerability detection in modern software systems.

Phuong Nguyen Thi Thanh is a faculty member at the School of Computer Science and Artificial Intelligence, Duy Tan University. Her research focuses on artificial intelligence, particularly computer vision and sequence based learning, including convolutional and recurrent neural architectures for information extraction and recognition from image and text data.

Huong Mai Quoc is a specialist at the General Affairs Office, School of Computer Science and Artificial Intelligence. He is passionate about research directions related to intelligent information systems, data-driven decision-making models, and the application of artificial intelligence in organizational management and digital transformation. He is currently involved in projects developing AI based automation tools and predictive analytics systems to improve operational processes in the education sector and enhance the effectiveness of data driven management.

Dung Nguyen Thi Thuy (corresponding author) is a lecturer at the School of Computer Science and Artificial Intelligence, Duy Tan University. She holds a Master's degree from France, with research focused on artificial intelligence, particularly machine learning models and intelligent systems. She is also interested in generative AI applications for students, including learning support tools and AI powered solutions that enhance creativity and interaction in modern educational environments.