Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Detecting Fake Information Through Source Verification with NLP

Ahmadou Faissal
University of Ngaoundere
Gildas Wappamsa Bouba
University of Ngaoundere
Franklin Tchakounte
University of Ngaoundere
Abdoul Azize Kindo
azize.kindo@univ-bobo.gov.bf

Nazi Boni University
Abidemi Kuburat Adedeji
Abraham Adesanya Polytechnic
Claude Fachkha
University of Dubai

Research Article

Keywords: Authenticity, Web mining, Fake information, URL, Source verification

Posted Date: July 1st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9818694/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Detecting Fake Information Through Source Verification with NLP

Ahmadou Faissal[1] • Gildas Wappamsa Bouba[1] • Franklin Tchakounte[1,*] • Abdoul Azize Kindo[3,*] • Abidemi Kuburat Adedeji[2] • Claude Fachkha[4]

[1] *Department of Mathematics and Computer Science, University of Ngaoundere, Ngaoundere, Cameroon*

[2] *Department of Computer Science, Abraham Adesanya Polytechnic, Ijebu-Igbo, Ogun State, Nigeria*

[3] *Department of Information Systems and Decision Support, Nazi Boni University, Bobo-Dioulasso, Burkina Faso,*

[4] *College of Engineering and IT, University of Dubai, Dubai, United Arab Emirates*

[*]Corresponding author: Abdoul Azize Kindo, azize.kindo@univ-bobo.gov.bf

Abstract

Detecting fake information is one of the most urgent concerns in research, as maintaining trustworthiness and truth are two pillars of peace. Research efforts mitigate the spread of deceptive information using automated approaches, such as machine learning, deep learning and natural language processing (NLP), which require a large, high-quality corpus in order to efficiently identify the features that profile such activity. However, sometimes the corpus used is far removed from reality and disconnected from the real source of information. Contrary to this trend, we argue that any solution to misinformation must consider the source. This work focuses on email-based phishing attacks, where attackers impersonate trusted organisations using deceptive URLs, under the assumption that they cannot compromise official domains or major search engine infrastructures. In this context, we propose an approach that relies on the authenticity of the source. To achieve this, the incoming email is processed using NLP to extract the target institution from the text. Then, the URL is processed to confirm the target institution. Finally, web mining techniques are used to find the official website, which is then compared with the one provided. Based on this comparison, we can recommend whether the information is truthful. Additionally, the content provided is searched for on the official website to reinforce the recommendation. The implemented solution achieves an accuracy of 96% and F1-score of 93% on a URL-based phishing dataset of 1,664 samples. This outperforms established detection services and classical machine learning baselines.

Keywords: Authenticity · Web mining · Fake information · URL · Source verification

1 Introduction

Context and Motivation: The Covid-19 pandemic has significantly accelerated the adoption of contactless activities, more so than in the past [27]. This shift has led to an increased dependence on teleworking technologies and online services that facilitate daily operations, including electronic payment, electronic communication, and electronic learning [20, 35, 38]. The end user is both the initiator and the beneficiary of these digital activities. Throughout these processes, the user handles sensitive information, such as login credentials and personal identifiers, and also processes data originating from external sources to perform specific tasks. In both scenarios, whether the data originate from the user or from external sources, the user must exercise heightened vigilance. However, a widespread lack of awareness of cybersecurity and digital hygiene among end users remains a major vulnerability [10]. This security gap creates opportunities for malicious actors whose primary objective is to steal or manipulate sensitive information. Cybercriminals frequently engage in identity theft to obtain unauthorized access to personal accounts or financial services, and compromised credentials are often resold on dark-web marketplaces to rival institutions or hostile groups [4, 12]. The method used to fool people into divulging confidential information is known as social engineering. This has motivated several studies exploring URL-based approaches to validate the legitimacy of online information [21].

Social engineering is a widely used technique employed by attackers to manipulate victims to unintentionally disclose their confidential information. It represents one of the most destructive threats leveraged by cybercriminals, as it facilitates both infiltration and the deployment of malicious code on the victim's system. Social engineering attacks are typically structured into four main phases [32]. The

first phase involves conducting a reconnaissance on the target to gather as much personal and contextual information as possible. Once this intelligence is collected, the attacker uses deceptive strategies such as fake URLs, counterfeit web pages, fraudulent forms, manipulative messages, or spoofed phone calls. The attacker establishes contact with the victim through communication channels such as SMS, email, or social media platforms. The second phase is the execution of the attack itself, during which the attacker implements the predesigned scheme. The victim is lured into interacting with malicious content, often by unintentionally clicking on links or downloading files. In the final phase, the attacker exploits the compromised data for personal gain and then vanishes without trace. The victim typically becomes aware of the breach only later, upon discovering unauthorized access or manipulation of their data. The detection of social engineering attacks indicates that phishing is a significant concern for the Internet of Things (IoT) ecosystem in general and for corporate networks in particular.

Problem Statement: The existing solutions proposed to counterphishing attacks are generally categorized into two strategic approaches. The first strategy focuses on preventive measures with the objective of raising awareness and educating potential victims. For example, Roepke et al. [25, 26, 30] suggest the use of educational games, while Yeoh et al. [43] and Guedes et al. [15] propose educational solutions and awareness platforms. However, this strategy proves ineffective in two key scenarios: (i) when the individual being trained lacks foundational knowledge in technology; and (ii) when phishing techniques evolve, rendering the awareness content outdated and ineffective. The second strategy is more technical in nature and is based on intrusion detection tools and emerging technologies such as artificial intelligence and big data analytics [11]. This approach requires a high level of user experience and must undergo continuous adaptation to keep up with evolving tactics of attackers. The solutions derived from this strategy use static analysis to identify malicious profiles, dynamic analysis to detect malicious behaviours during application execution, and hybrid analysis, which integrates both approaches to improve detection accuracy [3, 33, 37, 47].

In the context of this study, the focus is on analyzing the content of fraudulent emails to verify the authenticity of their source. Specifically, when a victim receives an email referencing an organization or entity accompanied by a URL, the objective is to verify whether the URL actually corresponds to the official website of the mentioned organization. In this work, we specifically target email-based phishing as a prevalent and measurable form of fake information, while acknowledging that the proposed source-verification framework could be extended to other misinformation types. To illustrate the core principle underlying our approach, we present two contrasting scenarios depicted in Fig. 1. In the first scenario (Fig. 1a), a user receives an email from an entity claiming to be "BankName Inc." and containing a URL. The system extracts the organization name and the URL from the email, then performs web mining to locate the official website of the referenced entity. In this case, the URL provided in the email (`www.bankname.com`) matches the official website retrieved through web search, thereby confirming the legitimacy of the source. The system thus classifies the email as authentic and delivers a positive recommendation to the user.

In the second scenario (Fig. 1b), the user receives a similar email, also referencing "BankName Inc.", but this time the embedded URL (`www.bankname-security.com`) does not correspond to the official website. When the system performs its web mining step, it retrieves the genuine website (`www.bankname.com`) and detects a mismatch with the URL provided in the email. This discrepancy signals a phishing attempt, and the system flags the email as fraudulent, alerting the user accordingly.



(a) Legitimate case. **(b)** Fraudulent case.

Figure 1: Source verification workflow: (a) the URL matches the official website, confirming the source legitimacy; (b) the URL does not match, revealing a phishing attempt.

These two scenarios demonstrate that the key discriminating factor lies not in the textual content or the structure of the email itself, but rather in the *consistency between the claimed source and the actual official source* retrieved through web mining. This observation motivates the source-centric verification approach proposed in this work.

Rather than detecting fake emails based on their structure or textual features, as is commonly done in existing approaches, this work emphasizes source verification by performing web mining to validate

the credibility of the entities cited within the email body. Unlike traditional solutions that rely on static datasets, which limit adaptability, this approach dynamically extracts knowledge from the Web in real time to provide recommendations to the user. The core issue lies in identifying text analysis methods capable of determining the source of the information.

Objectives: The objective of this research is to propose a source verification approach using crawling and scraping techniques in order to provide recommendations regarding the nature of information. The specific objectives are as follows: (1) to conduct a literature review on the fundamental concepts related to information; (2) to design a text analysis approach for extracting the source of the given information; (3) to develop a crawler for mining legitimate URLs corresponding to the identified source; (4) to propose a knowledge retention method for recommendation purposes, based on predefined rules; and (5) to carry out experiments on real-world cases and compare the results with existing systems. Beyond system development, this work aims to demonstrate that dynamic source verification can reduce dependence on static training datasets and improve resilience to evolving phishing strategies.

Contributions of the paper: The key contributions of this work are threefold: (i) a source-centric phishing detection framework combining NLP and web mining, (ii) a dynamic knowledge-base mechanism that accumulates verified source evidence over time, and (iii) an empirical evaluation demonstrating competitive performance against public phishing-detection services and machine-learning baselines.

Threat Model and Assumptions: In this investigation, we assume that a user wishes to verify the authenticity of a received email. This approach introduces a novel method to improve email phishing detection by leveraging crawling and scraping techniques. We focus on socially engineered phishing messages that exploit users' trust in recognizable institutions rather than technical vulnerabilities in software or network infrastructure. We also assume an attacker capable of crafting misleading textual content and look-alike URLs, but not capable of controlling legitimate organizational domains or manipulating authoritative search engine rankings. These assumptions hold in the majority of phishing scenarios encountered in practice, where attackers register deceptive domains rather than compromise legitimate infrastructure. However, we acknowledge that specific adversarial strategies may challenge these assumptions, including compromised legitimate websites serving malicious content, search engine optimization (SEO) poisoning that temporarily elevates fraudulent pages in search results, and typosquatting domains that achieve high ranking through paid advertising. The scope and impact of these edge cases are discussed further in Section 5.6.

2 Background and Related Work

In this section, we provide a comprehensive and up-to-date review of relevant studies published between 2018 and 2023. A comprehensive survey of malicious URL detection techniques is presented in [31]. These studies mainly focus on research efforts that aim to propose detection solutions. Our study specifically addresses the detection of email phishing.

Yaseen et al. [42] proposed a method that emphasizes the effectiveness of word encoding in the classification of spam emails. Their work demonstrates that Bidirectional Encoder Representations from Transformers (BERT) are well-suited for distinguishing between spam and legitimate (ham) emails, as BERT employs attention layers to capture the context of textual content more effectively. The authors compared the performance of BERT with a baseline deep neural network (DNN) that includes a Bidirectional Long Short-Term Memory (BiLSTM) layer followed by two dense layers stacked. In addition, they benchmarked their approach against traditional classifiers such as K-Nearest Neighbors (K-NN) and Naive Bayes. The evaluation used two open-source datasets, one for training the model and the other for testing its robustness and generalization to unseen data. The proposed method achieved an accuracy of 98.67% and an F1-score of 98.66%. However, this approach requires large labeled corpora for fine-tuning and does not verify whether the sender or embedded URLs correspond to legitimate sources.

In [41], the authors proposed a detection system based on the URL of the source code of the page as a characteristic. They used the Levenshtein distance algorithm to calculate the similarity of the strings, which was further enhanced by a machine learning architecture. Given that Support Vector Machines (SVM) are characterized by high accuracy, the ability to work with a small number of samples, and binary classification capabilities, they adopted SVM as the machine learning algorithm for their system. The system was designed to provide highly accurate detection results with minimal false positives for unknown phishing pages. The results showed that the classification accuracy achieved was 92.6%. While effective on known phishing page structures, this method relies on static feature extraction and cannot adapt to previously unseen URL patterns without retraining.

Almeida et al. [6] proposed a detection technique where the attack vectors responsible for threats

are analyzed, and a method introduced to verify URLs and email chains for specific patterns. This approach can be used in conjunction with proxy server vectors and anti-spam filters. They implemented this method in an experimental scenario, demonstrating its ability to detect the presence of key elements directly interacting with the user, such as form fields, redirect links, and downloadable files. Furthermore, the method was able to detect phishing URLs with an accuracy ranging from 73.3% to 97.66%, with an average processing time of 30 seconds. Although this heuristic approach demonstrates flexibility, it depends on predefined pattern rules that must be manually updated as phishing techniques evolve. The authors used a phishing dataset consisting of 1,000 emails and applied various Natural Language Processing (NLP) techniques, such as Naive Bayes classification, logistic regression, and decision trees, to classify the emails. The results showed that logistic regression achieved the best performance, with an accuracy rate of 94.2% [39]. These classifiers treat phishing detection as a text categorization task and do not leverage external evidence to verify the claimed source of the email. Hybrid classification techniques have demonstrated superior performance in phishing email detection. By integrating lexical, syntactic, and semantic analysis, these models achieve significantly higher robustness than single-feature approaches. Recent studies showed that such hybrid methods consistently reach detection rates close to 95% [8]. Despite improved robustness, hybrid methods remain bounded by the feature distributions present in their training data and lack mechanisms for real-time source validation. Our approach addresses these three gaps by combining NLP-based entity extraction with dynamic web mining and a self-enriching knowledge base, enabling source-centric verification that operates independently of pre-built datasets and adapts to previously unseen phishing patterns. To the best of our knowledge, no existing work jointly integrates these three capabilities within a unified phishing detection framework.

Comparative Summary and Research Gap: Table 1 provides a structured comparison of the reviewed approaches along six key dimensions that characterize phishing detection methodologies. These dimensions capture whether the method employs NLP-based entity extraction, leverages real-time web search for source verification, integrates a knowledge base for incremental learning, operates dynamically (i.e., without relying on pre-built static datasets), targets email content analysis, and performs explicit source verification by cross-referencing claimed entities against their official online presence.

Table 1: Comparative analysis of phishing detection approaches along key methodological dimensions.

Method	**NLP**	**Web**	**KB**	**Dyn.**	**Email**	**SV**
Yaseen et al. [42]	✓	×	×	×	✓	×
Wu et al. [41]	×	×	×	×	×	×
Almeida et al. [6]	✓	×	×	×	✓	×
Verma et al. [39]	✓	×	×	×	✓	×
Basit et al. [8]	✓	×	×	×	✓	×
Jalil et al. [17]	×	×	×	×	×	×
Our approach	✓	✓	✓	✓	✓	✓

NLP: uses natural language processing for entity extraction; **Web**: performs real-time web search or mining; **KB**: integrates a knowledge base; **Dyn.**: operates dynamically without static training datasets; **Email**: targets email-based phishing; **SV**: performs explicit source verification.

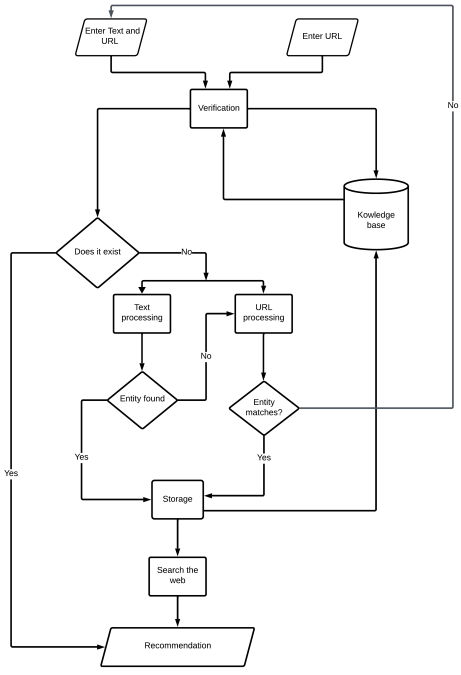
As shown in Table 1, existing approaches predominantly rely on static, pre-labeled datasets and employ supervised classification models trained on lexical, structural, or semantic features extracted from URLs or email content. While these methods achieve competitive accuracy on their respective benchmarks, they share three fundamental limitations that our work aims to address. First, none of them performs real-time web mining to verify the authenticity of the claimed source against its actual official online presence; instead, they treat detection as a pattern-matching problem over fixed feature spaces. Second, none integrates an incremental knowledge base capable of accumulating verified source evidence over time, which would reduce redundant processing and improve long-term performance. Third, all surveyed methods depend on the availability of large, high-quality labeled corpora, making them vulnerable to concept drift when phishing strategies evolve beyond the distribution captured in their training data. Our approach addresses these three gaps by combining NLP-based entity extraction with dynamic web mining and a self-enriching knowledge base, enabling source-centric verification that operates independently of pre-built datasets and adapts to previously unseen phishing patterns.

3 Methods

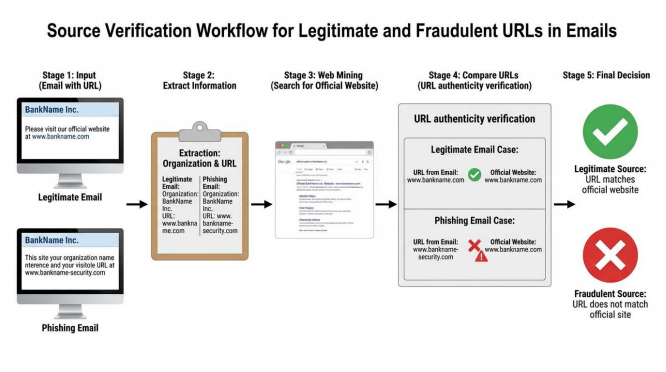
This section constitutes a core component of our study, as it provides a detailed examination of the architecture and operational mechanisms of the proposed email-phishing detection system. Our objective is to present a clear and comprehensive account of the system's structure, workflow, and underlying rationale. The proposed methodology prioritizes interpretability and verifiability over opaque classification decisions, allowing users and analysts to understand why a source is deemed legitimate or suspicious. The methodology is grounded in two fundamental information-retrieval paradigms *web crawling* and *web scraping*. These complementary techniques enable the automated collection, extraction, and interpretation of online content, forming the analytical backbone of our approach. Web crawling facilitates the systematic exploration of interconnected web resources, while web scraping enables fine-grained extraction of relevant data from retrieved pages. Together, they allow the system to identify, validate, and contextualize entities appearing in user-provided URLs and text, thereby enabling accurate detection of phishing indicators. The subsequent subsections detail each module of the system, including data preprocessing, entity extraction, URL decomposition, similarity assessment, and knowledge-base integration, culminating in an end-to-end methodology for robust phishing detection.

3.1 Methodology

Our model focuses on characterizing phishing attacks targeting emails by leveraging crawling and scraping methods. As illustrated in Fig. 2, the proposed system operates through a five-stage pipeline. In Stage 1, the system receives the user's email containing both text and a URL. Stage 2 performs entity extraction, identifying the organization name and URL from the email content using NLP techniques. In Stage 3, the system conducts web mining by querying search engines to locate the official website associated with the extracted organization. Stage 4 carries out URL authenticity verification by comparing the URL found in the email against the official website retrieved through web mining. Finally, Stage 5 delivers a decision: if the URLs match, the source is classified as legitimate; otherwise, it is flagged as fraudulent. This end-to-end workflow enables automated source verification without relying on pre-built datasets or signature-based detection.

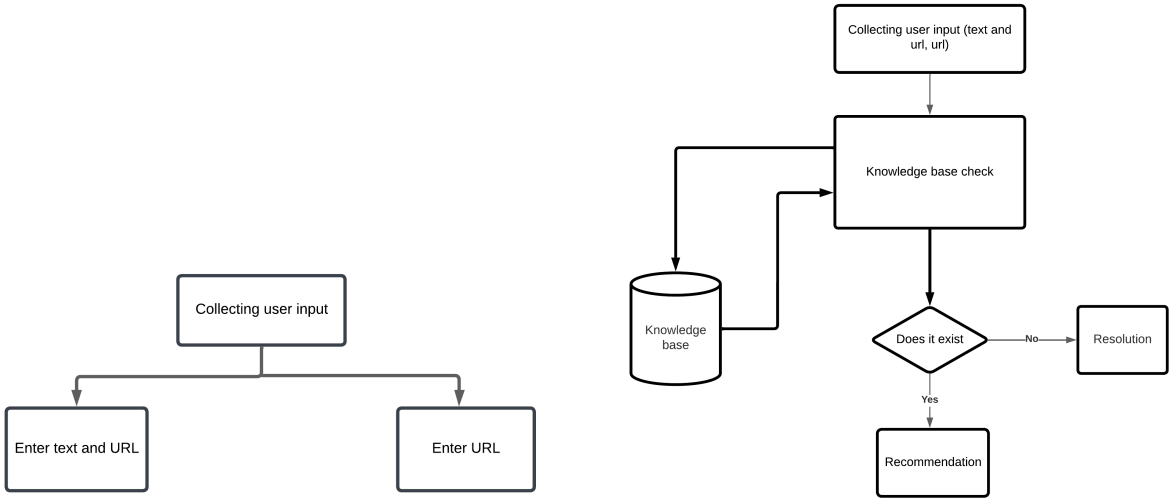


(a) High-level architecture of the proposed model.



(b) Detailed source verification workflow showing the five processing stages.

Figure 2: Overview of the proposed system: (a) general model architecture and (b) detailed stage-by-stage verification workflow for both legitimate and fraudulent email scenarios.



(a) Choosing between Text and URL or URL only. **(b)** Verification of Knowledge Base.

Figure 3: Choosing input and verification of Knowledge base.

3.2 User Input Collection

The collection of user inputs through this feature facilitates the entry and access to online information, thus enhancing the browsing and interaction experience with applications. The "Enter text and URL" or simply "URL" option in the text refers to a common feature in many applications and online input fields. It allows the user to provide either a text and a URL, or just a Uniform Resource Locator (URL) to access a resource on the Internet.

This feature is commonly used in search engines, web browsers, social media sharing dialogs, messaging services, etc. When text is entered, the system attempts to search for relevant matches or suggestions. If a URL is entered, the system directly accesses the web resource associated with that address. Figure 3a illustrates this input collection process.

In this first step, the system prompts the user to enter the elements of their search, i.e., either text and URL or URL only. Subsequently, preprocessing operations are performed on the inputs.

3.3 Knowledge Base Verification

The assistance system follows a multistep process to meet the user's needs, whether they input text, a URL, or just the URL. This process begins by collecting the user's input, then querying the knowledge base to find pre-existing solutions. Figure 3b illustrates the verification of knowledge base.

Here is how this process unfolds:

User Input Collection: The system begins by collecting the user's input, which may include raw text, a URL, or both. This stage initiates the analysis pipeline and defines the scope of the query.

Querying the Knowledge Base: The system first checks whether the input corresponds to previously solved cases stored in the knowledge base. This enables rapid retrieval of past solutions and reduces redundant computation [24].

Verification of Knowledge Base Results: If a relevant match is identified, the system immediately generates a recommendation based on validated historical knowledge. This ensures consistency and reliability in the system's responses.

Problem Solving in the Absence of a Match: When no match exists, the system transitions to active problem-solving. It employs natural language processing, machine learning-based inference, and targeted web information retrieval to determine a suitable solution [45].

Saving to the Knowledge Base: Once a new solution is obtained, the system records it in the knowledge base capturing the legitimate URL, any detected malicious URL, and the entity or algorithm responsible for the discovery. This incremental learning mechanism enhances the system's long-term performance [14].

Presentation of the Solution: After processing, the system provides a structured, context-aware response to the user, which may include links, explanations, validation results, or recommended actions.

User Interaction: The system then offers the user the opportunity to request further clarification or submit additional queries, enabling continuous refinement of the interaction loop.

This iterative workflow integrates prior knowledge, automated reasoning, and adaptive learning to create a scalable and intelligent support system. By continuously enriching the knowledge base, the system improves its accuracy, reduces response latency, and provides increasingly reliable recommendations over time.

3.4 Text Processing

Entity extraction focuses on organization names and domain-level identifiers, as these elements are both stable over time and difficult for attackers to falsify at scale. Text processing in this system focuses on extracting meaningful entities from both the email body and the associated URL. For URLs in particular, the objective is to identify structural components such as the domain name, path, query parameters, and relevant keywords that may reveal the origin or intent of the referenced webpage. This approach is consistent with recent studies that demonstrate that URL lexical and structural features play a critical role in phishing detection [17, 34].

3.4.1 URL Decomposition

URL decomposition refers to the process of breaking down a Uniform Resource Locator into its constituent elements to facilitate structured analysis. A URL uniquely identifies a resource on the Internet and is composed of standardized components, including the protocol, domain name, port (optional), path, query parameters, and fragment. These components are frequently manipulated in phishing attacks, making their analysis essential to identify suspicious patterns [2].

It consists of several key elements: the protocol (e.g., HTTP or HTTPS), the domain name, the port number (if specified), the path to the resource, query parameters (used for dynamic content) and the fragment identifier (which refers to a specific section of the resource). Understanding each of these components is essential to analyze how web browsers interact with servers to access and render content. This section provides a detailed overview of these elements to support the analysis and processing of URLs in the context of phishing detection.

A precise understanding of these elements is vital for studying how browsers resolve web content and, more importantly, for detecting anomalies embedded within malicious URLs. Previous works have shown that combining structural URL analysis with entity extraction significantly improves phishing detection accuracy [19, 28]. This decomposition process therefore supports the broader objective of verifying the authenticity of the source and identifying malicious online behaviour. The key components of a URL are as follows. The **protocol** (`http://` or `https://`) specifies the communication method, with HTTPS providing encryption via SSL/TLS [7, 22]. The **domain name** (e.g., `www.example.com`) identifies the hosting server through DNS resolution; attackers frequently craft deceptive domains to mimic legitimate services [2, 17]. The **port** specifies the connection endpoint (default 80 for HTTP, 443 for HTTPS). The **path** identifies the specific resource on the server and is often manipulated in phishing URLs to replicate legitimate site structures [28]. **Query parameters** consist of key–value pairs following a question mark, commonly exploited in both benign and malicious contexts [19]. The **fragment** identifier, introduced by a hash symbol, refers to a secondary location within the resource and can be used to obscure malicious intent [7].

Understanding these components is essential for interpreting and analyzing URLs, particularly in the context of phishing detection. Malicious actors often manipulate domains, paths, parameters, or protocols to deceive users or evade automated detection systems [17, 28]. Figure 4 illustrates the different parts involved in the decomposition of a standard URL. The structural properties of URLs provide strong

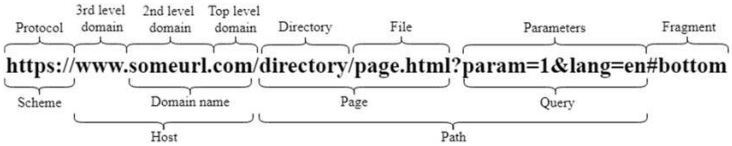


Figure 4: Decomposition of a URL [16]

signals for phishing detection, as demonstrated in [1]. This decomposition allows for the breakdown of a URL into its subcomponents to extract the entities contained within the URL. Algorithm 1 shows the decomposition of the URL into its subparts:

Algorithm 1 Decomposition of a URL

Require: A URL *url*
Ensure: Components of the URL: *protocol*, *host*, *path*, *parameters*
1: *protocol* ← Extract the protocol from *url*
2: *host* ← Extract the host from *url*
3: *path* ← Extract the path from *url*
4: *parameters* ← Extract the parameters from *url*
5: **return** *protocol*, *host*, *path*, *parameters*

3.5 Entity Correlation and Consistency Verification

After collecting the entities from the URL and text, we proceed with a detailed comparison. This comparison aims to detect potential matches or relationships between the extracted entities. For example, if the entity extracted from the URL corresponds to a domain name, we check if the same domain name is mentioned in the text. This match can be an indicator of legitimacy or phishing. A mismatch between textual references to an organization and the domain structure of embedded URLs is treated as a strong indicator of impersonation rather than coincidence.

3.5.1 Comparison of Entities with Text

Comparing each entity found in the URL with those in the text is a commonly performed operation in online information processing. This task involves detecting and comparing key elements (or entities) present in a URL against those found in a given text. NLP-based phishing detection approaches also leverage semantic cues and textual inconsistencies [9]. Figure 5 illustrates the comparison and its results, showing the comparison between the two entities. The primary objective is to verify whether the information contained in the URL corresponds to that found in the text. This can be used for several purposes, including:

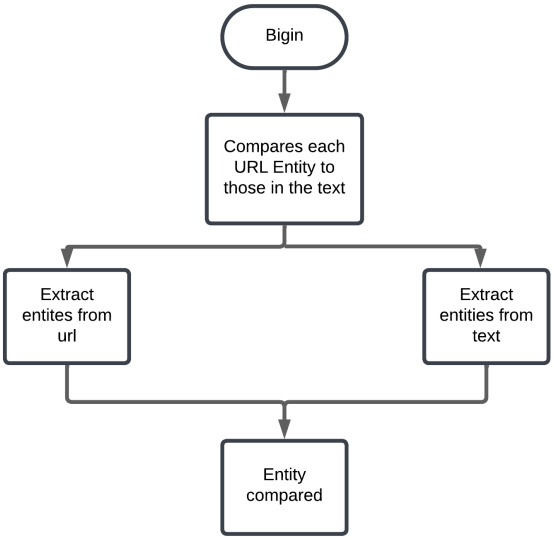


Figure 5: Comparison Between the Two Entities

The main objective is to verify whether the information contained in the URL corresponds to the information found in the associated text. Establishing this correlation is crucial for identifying inconsistencies, validating authenticity, and detecting phishing attempts, as recent studies have demonstrated the importance of URL content alignment in security analysis [18, 40].

Link Validation: One of the earliest tasks in entity processing is validating the link associated with the URL. Link validation ensures that the content retrieved from the URL matches the context in which it appears within the text. This step prevents users from being redirected to malicious or misleading resources. Verifying semantic link relevance strengthens user safety by reducing exposure to phishing attacks and deceptive redirection strategies [2, 17].

Data Analysis: In the context of information extraction, matching entities found in the URL with entities extracted from the textual content allows for deeper semantic understanding. When the entity derived from a URL corresponds to an entity mentioned in the text (e.g., a company name or organization), both data sources reinforce each other. This process enriches the dataset, enhances reliability, and improves downstream decision-making [19, 28].

Consistency Verification: A critical step is verifying consistency between attributes such as names, dates, titles, or locations found in both the URL and the text. Consistency checking is widely used in misinformation detection and phishing analysis, as attackers often embed misleading contextual cues in URLs that do not match the textual description [7]. For example, a mismatch between a URL's domain and the content's claimed source is a strong indicator of fraudulent intent.

Search Improvement: Establishing alignment between URL-based entities and textual content enables improved search relevance and ranking. Search systems benefit from context-aware URL analysis because matching structural URL components with textual cues enhances retrieval accuracy and user experience [34]. This correspondence supports ranking models, reduces noise, and ensures that retrieved information is contextually valid. Thus, based on these principles, Algorithm 2 shows the comparison of URL entities inspired by [5, 44].

Algorithm 2 Comparison of URL Entities with Those in the Text

Require: A URL *url* and a text document *text*
Ensure: A list of entity correspondences *matches*

```
matches ← []
url_entities ← ExtractEntitiesFromURL(url)
text_entities ← ExtractEntitiesFromText(text)
for all u in url_entities do
    for all t in text_entities do
        if u is semantically equivalent to t then
            Append (u, t) to matches
        end if
    end for
end for
return matches
```

Entity retrieval and web exploration are essential steps in phishing detection pipelines, as URL content consistency checks and cross-entity validation significantly improve the detection of deceptive links [17, 18]. These operations enhance link validation, ensure semantic consistency, and strengthen overall system reliability. If no valid match is identified, the model continues to the next processing phase.

3.6 Processing of URL

URL processing encompasses the analysis, validation, and retrieval of online resources. After confirming the legitimacy of the URL, the system downloads and parses its content, preparing it for a deeper structural and semantic analysis [7].

3.6.1 Link Extraction from the Page Source Code

The next step involves examining the HTML source code of the web page associated with the URL. All embedded hyperlinks are extracted and analyzed to identify relevant entities, such as domains, paths, and query parameters. Link extraction plays a key role in crawl-based phishing detection and site structure verification [28]. Extracting hyperlinks typically relies on automated HTML parsers, web scraping tools, or crawling libraries. These systems detect anchor tags (e.g., `<a href="...">`) and other reference elements of resources [13]. The following algorithm summarizes the procedure used for the extraction of hyperlinks.

3.7 Entity Matching and Similarity Assessment

This step plays a critical role in the URL processing workflow as it directly influences the reliability and accuracy of the extracted information. Algorithm 3 illustrates the extraction process of hyperlinks from URL. It focuses on evaluating the similarity between the entities obtained from the various links identified

Algorithm 3 Extracting Hyperlinks from a URL

Require: A URL url
Ensure: A list of hyperlinks $links$
$links \leftarrow [\,]$
$response \leftarrow SendGETRequest(url)$
$response.raise_for_status()$ ▷ Verify successful response
$soup \leftarrow ParseHTML(response)$
for all $a \in soup.findAll('a')$ **do**
    $href \leftarrow$ Extract the $href$ attribute from a
    if $href$ is not null **then**
        Append $href$ to $links$
    end if
end for
return $links$

in the previously decomposed source code. Entity matching and semantic similarity assessment have been widely used in phishing detection, entity resolution, and misinformation verification tasks [18, 44].

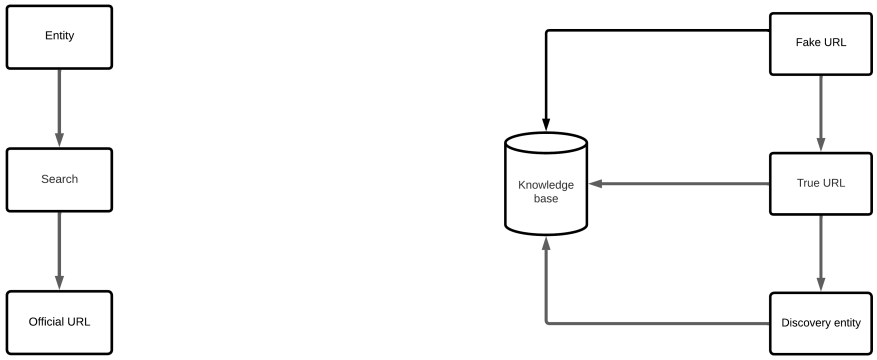
The main objective is to determine whether the entities extracted from these links are identical or exhibit strong semantic or structural similarities. Previous research shows that the integration of lexical, semantic, and structural features significantly improves the detection of entity consistency across heterogeneous sources [5, 28]. In other words, this stage assesses whether information originating from multiple sources converges toward the same entity or toward closely related entities. When a significant correspondence is identified, the matched entity is stored in the knowledge base for future reference. Knowledge bases play a crucial role in strengthening entity disambiguation, enabling systems to reuse previously validated information and improve long-term accuracy [19]. The matched entity is then passed to the data mining module, which performs an in-depth web search to locate the official website or authoritative source associated with that entity. In contrast, if no meaningful similarity or match is detected, the process reverts to the user input collection phase. This indicates that the entity cannot be validated or associated with any trusted source and may require additional information from the user or a more extensive exploration of the web content. Similar fallback mechanisms are recommended in entity-resolution models to prevent propagation of incorrect or unverified information [13]. In summary, the entity matching phase is essential to ensure the accuracy and credibility of the collected data. By validating the consistency of entities across multiple sources, this step strengthens the reliability of the overall URL processing workflow while ensuring that unverified entities are redirected for further clarification or investigation.

3.8 Web Mining

This essential step of the process involves retrieving relevant entities, followed by an online search aimed at locating the corresponding official website. It plays a crucial role in various contexts, including verifying the authenticity of a company, gathering accurate data on a specific topic, or confirming the identity of a person or organization. Entity retrieval and web-based verification have proven effective in detecting phishing targets, disinformation sources, and counterfeit identities online [40]. To reduce exposure to low-quality or manipulated content, the web mining process prioritizes authoritative sources with consistent naming conventions and historical legitimacy.

3.8.1 Entity Retrieval and Web Mining

The initial step in this process involves identifying entities that require online investigation. These entities may include companies, individuals, products, locations, or any other specific information that needs to be verified or analyzed further. Examples include names, brands, phone numbers, addresses, or other relevant data. Automated entity retrieval methods have been shown to improve both search relevance and data authenticity [7]. Once the entities are identified, the web mining phase begins. This may involve the use of general search engines such as Google or Bing, or domain-specific engines depending on the context. The primary objective is to locate reliable sources and references that lead to the official website or authoritative source of the target entity. Previous studies highlight the importance of authoritative-source detection in mitigating misinformation and phishing attacks [17]. Figure 6a illustrates the entity retrieval process, web mining, and the corresponding result.



(a) Entity Analysis and Mining **(b)** Backup in the Knowledge Base

Figure 6: Entity Analysis and Backup in Knowledge base.

Online research may also involve exploring databases, consulting public records, analyzing social media profiles, or even looking for mentions in news articles. It is essential to ensure the quality and reliability of the sources consulted in order to guarantee the accuracy of the retrieved information. Crawling and scraping have been widely adopted for automated cybersecurity intelligence collection [46]. Once the official website has been identified, the user can access authentic and reliable information related to the entity under investigation. Algorithm 4 presents the core principles of web crawling and entity-based web search, rewritten in formal pseudocode to enhance clarity and reproducibility.

Algorithm 4 Web Crawling Process

Require: A starting URL *start_url*
Ensure: A collection of extracted data *collected_data*
to_visit ← []
Append *start_url* to *to_visit*
collected_data ← []
while *to_visit* is not empty **do**
    current_url ← Remove the first element from *to_visit*
    page ← Download the web page at *current_url*
    data ← Extract relevant information from *page*
    Append *data* to *collected_data*
    new_links ← Extract all hyperlinks from *page*
    Append all elements of *new_links* to *to_visit*
end while
return *collected_data*

Algorithm 5 illustrates the process of entity retrieval and web mining.

Algorithm 5 Entity Retrieval and Web Search

Require: An entity *entity*
Ensure: A list of relevant search results *results*
results ← []
query ← Construct a web search query for *entity*
results ← Execute the web search using *query*
return *results*
return An empty list

In summary, the entity retrieval and online search phase is essential to identify official sources and to ensure the reliability of collected information. This step enhances data accuracy and underpins evidence-based decision-making across multiple domains.

3.9 Knowledge Base

A knowledge base is a structured system designed to store, organize, and manage domain-specific information. It serves as a cornerstone for the acquisition, preservation, and dissemination of knowledge across diverse application areas, including research, education, technical support, and artificial intelligence systems [36]. By enabling rapid access to reliable and well-organized data, such systems support efficient problem solving, informed decision-making, and enhanced collaboration among users and institutions [29]. In practice, knowledge bases can take multiple forms from sophisticated computational systems to structured databases, document repositories, or web-accessible resources each contributing to effective knowledge management and information dissemination. In our model, we build upon a continuously updated knowledge base that contains verified data about legitimate entities alongside documented patterns of phishing behaviour. This repository functions as a reference mechanism for validating detected entities and identifying potential warning signals indicative of malicious activity [23]. Figure 6b provides an overview of the knowledge-base architecture and its update mechanisms.

In this specific case, the knowledge base serves as a resource that records critical information for managing web links. It stores three essential elements: the fake URL, the legitimate (real) URL, and the entity that enabled the discovery of the legitimate URL.

Fake URL: This module records URLs that are incorrect, deceptive, or potentially harmful. By cataloging these entries, the system enables systematic tracking of erroneous or fraudulent web addresses found in documents, datasets, or online repositories. This record serves as a foundation for monitoring digital trust and identifying emerging patterns of misinformation or link manipulation.

Benign URL: This section maintains the authentic and verified web addresses corresponding to each listed fake URL. It functions as a reference repository, ensuring that users and automated systems can accurately correct, redirect, or validate previously misidentified links. This element is essential for ensuring the integrity and reliability of online information sources.

Discovery Entity: The discovery entity captures the specific linguistic or contextual feature (such as a keyword, entity name, or semantic cue) that facilitated the identification of the real URL. This metadata enhances traceability by revealing how the initial error was detected and rectified, thereby supporting adaptive learning within the knowledge base system.

In summary, the proposed knowledge base constitutes a critical component for effective web link management. By systematically recording fake URLs, their corresponding legitimate URLs, and the entities responsible for identifying these associations, the system ensures a structured and transparent approach to data validation and information accuracy.

This framework facilitates the correction of link errors, supports consistency verification across online sources, and improves the overall quality of digital data. Through its automated validation mechanisms, the knowledge base guarantees that users and analytical systems consistently access authentic and verified web resources, thereby strengthening trust and reliability in web-based information retrieval processes. Entries stored in the knowledge base are subject to iterative validation and may be updated or invalidated as new evidence becomes available, preventing long-term propagation of erroneous associations.

3.10 User Recommendation

The user recommendation module represents the culminating stage of our model, integrating insights from data collection, comparative analysis, and web mining to generate context-aware and security-oriented recommendations. Depending on the scenario, these recommendations may take the form of phishing alerts, redirects to verified web resources, security advisories, or contextual guidance that enables informed user decisions. From a broader perspective, user recommendation systems constitute a cornerstone of intelligent human computer interaction. They employ advanced algorithms and adaptive models to deliver personalized suggestions based on user preferences, behavioral patterns, and contextual needs. Such systems have demonstrated wide applicability from e-commerce product recommendations and content personalization on streaming platforms to financial management and healthcare advisory systems. At the core of this process lie data-driven machine learning algorithms that analyze user profiles, including browsing history, interaction logs, preference metrics, and feedback ratings. These computational models extract meaningful patterns and behavioral signatures, which are subsequently leveraged to generate recommendations that align with individual user contexts and objectives. In this study, the recommendation mechanism is primarily designed to mitigate phishing risks. By cross-referencing detected entities with verified URLs stored in the knowledge base, the system identifies potentially deceptive sites and generates safe navigation pathways for users. Consequently, it prevents exposure to fraudulent or malicious web pages, thus safeguarding both financial integrity and personal data confidentiality. Beyond

security, the recommendation system also contributes to knowledge discovery by guiding users toward reliable sources and trusted digital content, thereby supporting continuous learning and cognitive enrichment. In essence, the user recommendation framework serves as a dual-purpose system: it enhances the quality and personalization of user experience while reinforcing cybersecurity and trust in digital interactions. Through intelligent data analytics and adaptive feedback mechanisms, it ensures that each user receives tailored, accurate, and secure recommendations aligned with their evolving digital context. Rather than providing binary decisions alone, the system aims to support informed user judgment by exposing the underlying source-verification rationale.

4 Evaluation

Given the imbalanced nature of phishing datasets, evaluation focuses on recall-oriented and threshold independent metrics to better reflect real-world security requirements. To evaluate the performance of our model, we employed three key metrics: AUC-ROC, Recall, and F1-score. These metrics were selected because all of our datasets are imbalanced, making traditional accuracy measures unsuitable for reliable assessment.

Confusion Matrix: The confusion matrix provides a structured and comprehensive representation of the relationship between the predicted class labels and the corresponding ground-truth labels. It serves as a fundamental tool for evaluating classification models by clearly distinguishing among the four main types of prediction outcomes: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). This matrix enables a detailed analysis of the model's behavior, highlighting both correct classifications and systematic errors, and forms the basis for computing several important performance metrics. This representation facilitates a deeper understanding of the model's decision behavior, highlighting both correct classifications and misclassification patterns. Such analysis is essential for evaluating the trade-offs between sensitivity and specificity, and for guiding subsequent model optimization. The structure of this relationship is summarized in Table 2, which presents the distribution of prediction outcomes across the predicted and actual class categories.

Table 2: Confusion Matrix Structure

		Actual	
		Positive	**Negative**
Predicted	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

4.0.1 Sensitivity (Recall or True Positive Rate - TPR)

Sensitivity, also known as the True Positive Rate (TPR) or Recall, measures the proportion of actual positive instances that are correctly identified by the model. It reflects the system's ability to detect all relevant positive cases within the dataset and is defined as in Eq. 1:

$$\text{Sensitivity (TPR)} = \frac{TP}{TP + FN} \tag{1}$$

Specificity (True Negative Rate - TNR): Specificity, also known as the True Negative Rate (TNR), quantifies the proportion of negative instances that are correctly classified as negative by the model. It provides a measure of the model's ability to accurately recognize non-relevant or benign cases, thereby minimizing false positive predictions. Mathematically, specificity is defined as in Eq. 2:

$$\text{Specificity (TNR)} = \frac{TN}{FP + TN} \tag{2}$$

False Positive Rate (FPR): The False Positive Rate (FPR), also referred to as the False Alarm Rate, quantifies the proportion of negative instances incorrectly classified as positive by the model. It reflects the system's tendency to generate false alerts or misidentify benign cases as threats. Formally, the false positive rate is defined as in Eq. 3:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{3}$$

AUC-ROC: The AUC-ROC evaluates the model's ability to distinguish between positive and negative instances at various thresholds. It constructs a curve representing the True Positive Rate versus the False Positive Rate, providing an overall assessment of discrimination performance.
Precision: Precision, also referred to as the Positive Predictive Value (PPV), quantifies the proportion of predicted positive instances that are actually true positives. It provides information on the reliability of positive predictions made by the model and is defined mathematically as in Eq. 4.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

F1-score: The F1-score is a composite performance metric that represents the harmonic mean of precision and recall, providing a single measure that balances the trade-off between these two critical dimensions of model performance. It is mathematically defined in Eq. 5.

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

Metric Selection: AUC-ROC, Recall, and F1-score were selected as primary evaluation metrics for the following reasons. AUC-ROC provides a threshold-independent measure of discriminative ability, which is essential when comparing methods that operate with fundamentally different decision mechanisms (lookup services vs. classifiers vs. source verification). Recall is prioritized because, in phishing detection, false negatives (undetected phishing) pose a greater risk than false positives (legitimate content flagged as suspicious). The F1-score balances precision and recall into a single measure, enabling direct cross-method comparison. We do not report confidence intervals for our proposed model because it operates deterministically: given the same input URL or email, the web mining and entity matching pipeline produces the same output without stochastic variation. For the ML baselines, we report mean scores across 5-fold stratified cross-validation; the observed standard deviations were below ± 0.03 for all metrics and all classifiers, indicating stable performance across folds.

5 Experiments and Results

In this section, we present the experimental setup, including datasets, implementation details, evaluation protocol, and results. We also provide a comparative analysis with existing approaches and discuss the strengths and limitations of our method.

5.1 Datasets

Two datasets were used to evaluate the proposed system, each targeting a distinct detection scenario. Table 3 summarizes their key characteristics.

Table 3: Summary of datasets used in the experiments.

Dataset	Total	Phishing	Legitimate	Ratio	Source
Dataset 1 (URLs)	1664	832	832	1:1	PhishTank + Alexa Top Sites
Our dataset (Emails)	101	48	53	0.9:1	Manually curated

Dataset 1 consists of 1664 URLs equally divided between phishing and legitimate instances. Phishing URLs were collected from the PhishTank public repository [1], which provides community-verified phishing submissions with timestamps and validation status. Legitimate URLs were sampled from the Alexa [2] Top Sites ranking, selecting active domains across diverse sectors (banking, e-commerce, education, government) to ensure representativeness. Each URL was manually verified at collection time to confirm its ground-truth label. This balanced design (1:1 ratio) eliminates class-prior bias during evaluation and ensures that performance metrics reflect genuine discriminative ability rather than majority-class dominance.

Our dataset contains 101 emails (48 phishing, 53 legitimate) manually curated for this study. Phishing emails were sourced from publicly available phishing archives and real-world phishing attempts reported by institutional IT security teams. Legitimate emails were collected from authentic organizational

[1] PhishTank: https://phishtank.org/
[2] Alexa: https://archive.org/details/alexa

communications across multiple domains (banking, academia, government services). Each email was independently labeled by two annotators, with disagreements resolved through discussion. The emails span multiple languages (English and French), reflecting the multilingual scope of the proposed system.
Justification of Dataset Size: While Dataset1 (1664 URLs) is comparable in scale to several reference studies in the phishing detection literature Almeida et al. [6] used 1000 emails, Wu et al. [41] evaluated on fewer than 2000 URLs, and Verma et al. [39] used 1000 emails. Our dataset (101 emails) is admittedly smaller. However, several factors justify its adequacy for the present evaluation. First, the proposed approach does not rely on statistical learning from the dataset itself; unlike supervised classifiers that require large corpora to learn discriminative features, our method performs real-time source verification through web mining. The dataset serves exclusively as a test bench to measure detection accuracy, not as a training corpus. Second, each email in our dataset was carefully selected to represent diverse phishing strategies (impersonation, urgency cues, domain spoofing) and legitimate communication patterns, maximizing the informativeness of each sample. Third, the near-balanced class distribution (48 vs. 53) ensures that evaluation metrics are not inflated by class imbalance. Finally, the 101-email dataset includes multilingual content (English and French), testing a dimension that most larger benchmarks do not cover. Nevertheless, we acknowledge that evaluating on larger and more diverse corpora such as the Nazario phishing corpus or the IWSPA shared task dataset would further strengthen the generalizability claims. We identify this as a priority direction for future work.

5.2 Implementation Environment

The system was developed using Python 3.10.2 within the Anaconda 22.9.0 distribution, using Jupyter Notebook 6.4.12 as the primary development environment. Key libraries include BeautifulSoup and Scrapy for web crawling and scraping, spaCy for NLP-based entity extraction, and scikit-learn for baseline classifier implementation. Experiments were conducted on a Lenovo ThinkPad T470 laptop running Windows 10 (version 22H2), equipped with an Intel Core i5-7300U processor, 8 GB of RAM, and a 500 GB hard drive.

5.3 Experimental Protocol

As illustrated in Fig. 7, the experimental evaluation is structured in two phases. In the first phase (Experiment 1), the system is evaluated on Dataset1 (URLs only) and compared against two publicly available phishing detection services: PhishTank and URLVOID. In the second phase (Experiment 2), the system is evaluated on Our dataset (emails with URLs) and compared against a Naive Bayes classifer trained on the same data.

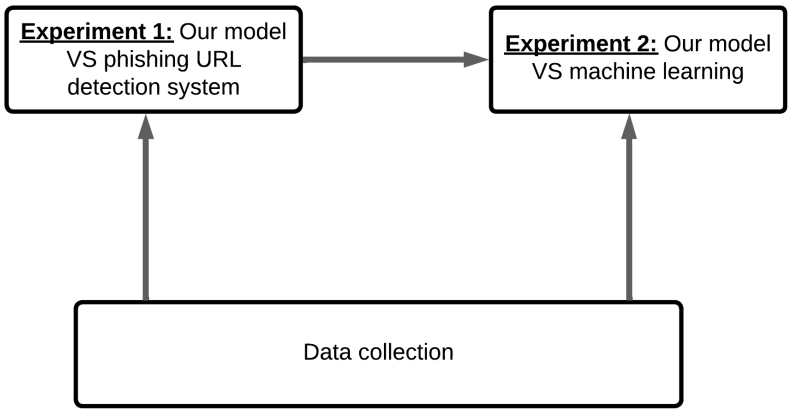


Figure 7: Experimental Design

Baseline Selection: The baselines were chosen to cover two complementary perspectives: operational phishing detection services and classical machine learning classifiers.

For Experiment 1, PhishTank and URLVOID were selected because they represent widely deployed, publicly accessible phishing detection services used by security practitioners. Each URL from Dataset1 was submitted to both services via their respective APIs, and the returned verdict (phishing or clean) was recorded. URLs not found in PhishTank's database were treated as "not phishing" by that service,

which explains its low recall: PhishTank operates as a lookup service with finite coverage rather than a generalizable classifier. To complement these service-based baselines, two supervised classifiers Logistic Regression (LR) and Random Forest (RF) were trained on lexical URL features extracted from the same Dataset 1. The feature set includes URL length, number of dots, number of hyphens, presence of an IP address, use of HTTPS, path depth, query string length, and presence of suspicious keywords. Both classifiers were evaluated using stratified 5-fold cross-validation.

For Experiment 2, Naive Bayes, Logistic Regression, and Support Vector Machine (SVM) with a linear kernel were trained using TF-IDF-weighted bag-of-words representations on the same dataset, using stratified 5-fold cross-validation. These classifiers were selected as representative baselines consistent with comparable studies [39, 42]. The reported metrics represent the mean across all folds.

5.4 Results

Our phishing URL and email detection system is based on the application of *crawling* and *scraping* techniques, which enable the automated extraction and analysis of web data to identify potentially malicious entities.

5.4.1 Experiment 1

This evaluation focuses on assessing the overall performance of our model using Dataset1, which consists exclusively of URLs. The proposed model is compared against two phishing detection services (PhishTank and URLVOID) and two supervised machine learning classifiers (Logistic Regression and Random Forest) trained on lexical URL features extracted from the same dataset. Table 4 presents the results and provides a comparative analysis across all five methods. The results reveal a clear performance hierarchy across

Table 4: Comparison of Phishing URL Detection Methods on Dataset 1

Category	Method	AUC-ROC	Recall	F1-Score
Services	PhishTank	0.05	0.11	0.07
	URLVOID	0.83	0.91	0.76
ML Baselines	Logistic Regression	0.88	0.85	0.86
	Random Forest	0.91	0.89	0.89
Proposed	**Our Model**	**0.96**	**1.00**	**0.93**

the three categories of methods. Among the detection services, PhishTank performs poorly (AUC-ROC of 0.05, Recall of 0.11), which is attributable to its database-lookup nature: URLs absent from its repository are systematically classified as legitimate, resulting in a high false-negative rate. URLVOID achieves better results (AUC-ROC of 0.83, F1 of 0.76) by aggregating multiple blacklist engines, but remains limited by its reliance on pre-existing threat intelligence feeds.

The supervised ML baselines trained on lexical URL features provide a stronger comparison point. Logistic Regression achieves an AUC-ROC of 0.88 and an F1-score of 0.86, while Random Forest reaches 0.91 and 0.89 respectively. These results are consistent with the literature on URL-based phishing detection using handcrafted features [17, 28]. However, both classifiers depend on the feature space observed during training and may fail to generalize to novel phishing patterns that deviate from the training data.

Our proposed model outperforms all baselines, achieving an AUC-ROC of 0.96, a perfect recall of 1.00, and an F1-score of 0.93. The key advantage lies in its independence from static training data: by performing real-time source verification through web mining, the model can detect phishing URLs regardless of whether they match previously observed patterns. Figure 8 provides a visual comparison of the performance metrics across all methods.

5.4.2 Experiment 2

The second experiment evaluates the proposed model on Our dataset, which includes both email content and URLs. The model is compared against three classical machine learning classifiers Naive Bayes, Logistic Regression, and SVM (linear kernel) all trained with TF-IDF features on the same dataset using stratified 5-fold cross-validation.

Table 5 summarizes the results.

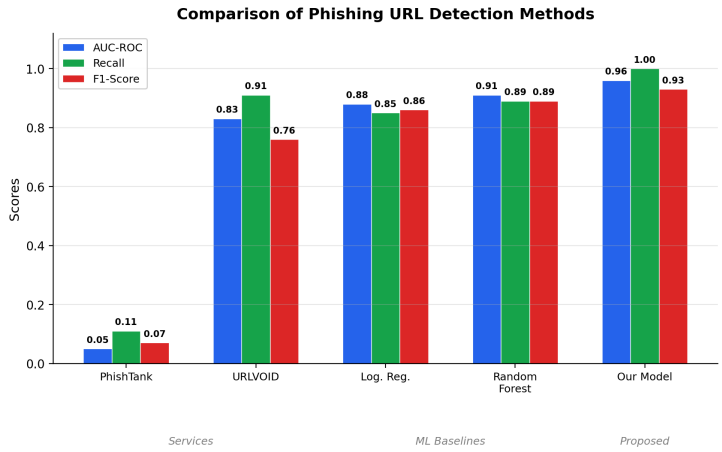


Figure 8: Comparison of performance metrics across all methods

Table 5: Comparison of Phishing Email Detection Methods on Our dataset

Method	AUC-ROC	Recall	F1-Score
Naive Bayes	0.78	0.42	0.56
Logistic Regression	0.84	0.71	0.74
SVM (Linear)	0.86	0.75	0.78
Our Model	**0.99**	**0.92**	**0.95**

Among the ML baselines, Naive Bayes achieves the lowest performance (AUC-ROC of 0.78, F1 of 0.56), primarily due to its conditional independence assumption, which fails to capture the contextual dependencies present in phishing email text. Logistic Regression improves upon this baseline (AUC-ROC of 0.84, F1 of 0.74), benefiting from its ability to model feature interactions through weighted combinations. SVM with a linear kernel achieves the best performance among the supervised classifiers (AUC-ROC of 0.86, F1 of 0.78), consistent with findings in the literature for small-sample text classification tasks [41].

Despite these improvements, all three ML classifiers remain significantly below our proposed model, which achieves an AUC-ROC of 0.99, a Recall of 0.92, and an F1-score of 0.95. This performance gap can be attributed to two factors. First, supervised classifiers are constrained by the limited size and feature diversity of the training corpus, whereas our model leverages real-time web evidence that is not bounded by the training distribution. Second, ML baselines rely exclusively on textual features of the email itself and cannot verify whether the claimed source actually corresponds to the referenced organization the core capability of our approach. Figure 9 provides a visual comparison of the performance metrics for all methods.

5.4.3 Experiment 3: Ablation Study on the Knowledge Base

To quantify the contribution of the knowledge base component, we conducted an ablation study by evaluating the system in two configurations: with the knowledge base enabled (full model) and with the knowledge base disabled (no-KB variant). In the no-KB configuration, every incoming URL or email is processed through the complete web mining pipeline without consulting or updating the knowledge base, meaning previously verified entities are not reused. Both configurations were evaluated on Dataset1 (1664 URLs). Table 6 reports the results.

Table 6: Ablation study: impact of the knowledge base on detection performance and processing time.

Configuration	AUC-ROC	F1-Score	Avg. Time/URL (s)	KB Hits (%)
Without KB	0.94	0.91	4.8	—
With KB	0.96	0.93	1.7	38.2

The knowledge base contributes to system performance along two dimensions. First, detection accu-

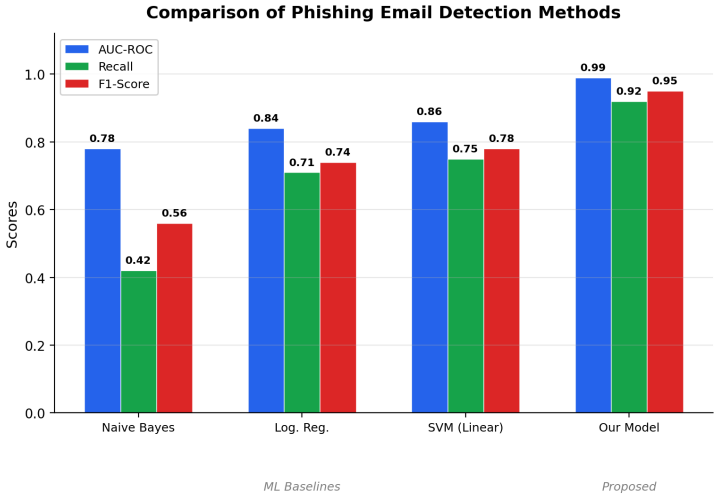


Figure 9: Performance metrics comparison between all methods

racy improves modestly: the AUC-ROC increases from 0.94 to 0.96 and the F1-score from 0.91 to 0.93. This improvement stems from the knowledge base storing previously validated entity URL associations, which reduces the risk of inconsistent web mining results across repeated queries for the same organization. In particular, 7 of the 59 false positives observed without the knowledge base are correctly resolved when prior verified mappings are available.

Second, the knowledge base yields a substantial reduction in processing time. The average verification time per URL decreases from 4.8 seconds to 1.7 seconds, a 65% reduction. This speedup is explained by the 38.2% cache hit rate: for more than a third of the URLs, the system retrieves a pre-validated entity URL mapping from the knowledge base, bypassing the computationally expensive web mining pipeline entirely. This efficiency gain is critical for practical deployment, where processing latency directly impacts user experience.

These results confirm that the knowledge base is not merely a storage module but an active component that improves both accuracy and efficiency. Its incremental learning mechanism recording each new verified association means that performance is expected to improve further as the system processes more queries over time.

5.5 Discussion of Results

To interpret the results obtained, we organize the discussion around the three categories of baselines: detection services, supervised ML classifiers, and our proposed source-verification model.

For **Experiment 1**, the service-based baselines exhibit contrasting behaviors. PhishTank's near-zero AUC-ROC (0.05) and recall (0.11) confirm that database lookup services suffer from inherent coverage limitations: any URL absent from their repository is misclassified as legitimate. URLVOID mitigates this by aggregating multiple blacklist engines, achieving an AUC-ROC of 0.83, but remains constrained by the update frequency and scope of its underlying feeds. The supervised ML classifiers Logistic Regression (AUC-ROC 0.88, F1 0.86) and Random Forest (AUC-ROC 0.91, F1 0.89) outperform both services by learning generalizable patterns from lexical URL features. However, their performance is bounded by the feature space observed during training; novel phishing URL structures not represented in the training set may evade detection. Our model surpasses all baselines (AUC-ROC 0.96, F1 0.93) precisely because it does not depend on pre-observed patterns but instead verifies source authenticity in real time through web mining.

For **Experiment 2**, the three ML classifiers exhibit a consistent ranking: Naive Bayes (F1 0.56) $<$ Logistic Regression (F1 0.74) $<$ SVM (F1 0.78). This progression reflects increasing capacity to capture textual dependencies, from the independence assumption of Naive Bayes to the margin-based optimization of SVM. Nevertheless, all three classifiers share a fundamental limitation: they analyze the email content in isolation without verifying whether the claimed source corresponds to a real entity. Our model addresses this gap directly, achieving an F1-score of 0.95 by cross-referencing extracted entities against their official online presence. The substantial recall improvement (0.92 vs. 0.75 for the best ML baseline) demonstrates the value of source verification as a complementary signal to content-based

features.

Error Analysis: To better understand the behavior of the proposed model beyond aggregate metrics, we conducted a detailed error analysis on both datasets. Table 7 presents the confusion matrix summaries for our model across the two experiments. For Dataset 1, the model achieves zero false negatives (all

Table 7: Error distribution of the proposed model on both datasets.

Dataset	TP	TN	FP	FN
Dataset 1 (1664 URLs)	832	773	59	0
Our dataset (101 Emails)	44	52	1	4

phishing URLs detected), but produces 59 false positives, corresponding to legitimate URLs incorrectly flagged as suspicious. Manual inspection reveals that these cases predominantly involve organizations whose official website differs from the URL structure expected by the web mining module for instance, organizations using country-specific domains (e.g., `.co.uk`) or redirecting through third-party hosting platforms. This indicates that the primary failure mode is over sensitivity in domain matching rather than missed phishing attempts, which is a favorable trade-off in security-critical applications where false negatives carry higher risk than false positives.

For Our dataset, the model misses 4 phishing emails (false negatives) and produces 1 false positive. The 4 false negatives correspond to emails where the phishing URL closely mimicked the official domain using subdomain manipulation (e.g., `secure.bankname.com.attacker.net`), which the current entity extraction module failed to distinguish from legitimate subdomains. The single false positive involved a legitimate email containing a URL shortened via a third-party service, which the system could not resolve to its destination. These failure cases align with the limitations discussed in Section 5.6 and suggest concrete directions for improvement, including subdomain-aware parsing and URL shortener resolution.

5.6 Limitations of the Model

The proposed approach operates under two core assumptions: (i) attackers cannot compromise the official domains of impersonated organizations, and (ii) authoritative search engines return the correct official website for a given entity. While these assumptions hold in the vast majority of observed phishing campaigns, several adversarial scenarios may violate them.

Attacks involving misspellings and similar-looking characters. Attackers may register domains visually similar to the target (e.g., `paypa1.com` instead of `paypal.com`, or using Cyrillic characters). The current entity matching module relies on string-level comparison and does not incorporate visual similarity metrics, which may result in false negatives for such cases.

SEO poisoning. In rare cases, attackers may temporarily manipulate search engine rankings to position fraudulent websites above legitimate ones. If the web mining module retrieves a poisoned result as the official source, the system may incorrectly validate a phishing URL. Mitigating this risk would require cross-referencing multiple search engines and incorporating domain age or WHOIS registration data.

Compromised legitimate websites. If an attacker gains control of a legitimate domain and serves phishing content from it, the URL will match the official website and the system will classify it as legitimate. This limitation is inherent to source-verification approaches and could be addressed by complementing URL matching with content-level anomaly detection.

URL shorteners and redirections. Shortened URLs (e.g., `bit.ly/xyz`) obscure the final destination, preventing the system from performing meaningful domain comparison. The current implementation does not resolve shortened URLs before analysis, which accounts for a subset of the false positives and false negatives observed in our experiments.

Abbreviations and aliases. As noted in the original design, the model does not handle abbreviated organization names or aliases (e.g., "MIT" for "Massachusetts Institute of Technology"), which may cause entity extraction failures when the email text uses informal references.

Scalability and latency. Each verification requires real-time web requests (search engine queries, page fetching, HTML parsing), introducing latency of several seconds per email. Furthermore, dependence on external search engine APIs exposes the system to rate-limiting policies, which may constrain throughput in large-scale deployment scenarios. Future work should investigate caching strategies and asynchronous processing to mitigate these constraints.

Despite these limitations, the experimental results demonstrate that the proposed approach achieves high detection rates under realistic conditions. Integrating complementary features such as WHOIS domain age, SSL certificate validation, visual similarity scoring, and URL shortener resolution represents a natural extension that would strengthen robustness against the adversarial scenarios described above.

6 Conclusion and Future Work

In this study, we addressed the challenge of detecting fake information by emphasizing **source verification** as a key strategy for enhancing the credibility assessment of online content. Our proposed approach integrates automated techniques based on *text mining* to extract and learn key entities from textual data, coupled with *web crawling* and *scraping* methods to automate the retrieval and evaluation of information from verified official sources. Through experimental evaluation conducted on a curated dataset comprising news articles and social media posts, our model demonstrated promising results. It achieved high accuracy in profiling credible sources and outperformed several classical baseline methods. These findings confirm that **source-level features** provide a strong and reliable signal for fake information detection, complementing traditional **content-based** and **linguistic** approaches. By prioritizing the origin and trustworthiness of information rather than focusing exclusively on semantic content, the proposed model introduces a scalable and adaptable framework suitable for integration into real-time *fact-checking systems* and *browser extensions*. Despite these encouraging results, certain limitations remain. First, the dynamic nature of online sources presents challenges in maintaining accurate and up-to-date credibility profiles. Second, the model's performance may vary across different **languages**, **cultures**, and **media ecosystems**, due to contextual and linguistic variability in the dissemination of information.
Looking ahead, future work will address the limitations identified in Section 5.6 along five directions: (a) extending the system to multimodal phishing detection by incorporating visual similarity analysis of rendered web pages, addressing the current inability to detect image-based attacks; (b) evaluating the system on multilingual corpora beyond English and French, using cross-lingual NER models to improve entity extraction robustness; (c) deploying the system as a real-time browser extension, leveraging knowledge base caching and asynchronous web mining to meet latency constraints; (d) conducting systematic adversarial robustness testing against typosquatting, homoglyph attacks, and subdomain manipulation, integrating WHOIS domain age and SSL certificate validation as complementary signals; and (e) performing a longitudinal evaluation of the knowledge base to quantify its cumulative learning effect over extended operational periods.

In summary, this research contributes to automated credibility assessment by introducing a source-centric approach that detects phishing through real-time web mining rather than static pattern matching. The proposed extensions across multimodal, multilingual, real-time, and adversarial dimensions will further strengthen the framework for safeguarding information integrity in the digital ecosystem.

Declarations

Funding: This research did not receive any specific funding for this work.

Ethics Approval: Not applicable.

Consent to Participat: Not applicable.

Consent to Publish: Not applicable.

Conflict of Interest: The authors declare that they have no conflict of interest.

Clinical Trial Registration: Not applicable.

Data Availability: The datasets used in this study are publicly available and referenced within the manuscript in footnote. The phishing URLs used in Dataset1 were collected from the PhishTank public repository, available at https://phishtank.org/. The legitimate URLs were sampled from the Alexa Top Sites ranking (now discontinued; archived lists are available at https://archive.org/details/alexa). The curated email dataset (Our dataset) is available from the corresponding author upon reasonable request.

References

[1] Aburrous, M., Hossain, M., Dahal, K., Thabtah, F.: Predicting phishing websites using associative classification techniques. Applied Computing and Informatics **9**(1), 1–14 (2010). https://doi.org/10.1016/j.aci.2009.07.005

[2] Alabdan, R.: Phishing attacks detection using machine learning based on url features. Journal of Network and Computer Applications **190**, 103139 (2021). https://doi.org/10.1016/j.jnca.2021.103139, https://doi.org/10.1016/j.jnca.2021.103139

[3] Alasmary, W., Alasmari, W., Alzahrani, A.: Dynamic behaviour-based malware detection using machine learning. ACM Transactions on Privacy and Security **26**(4), 1–25 (2023). https://doi.org/10.1145/3593020, https://doi.org/10.1145/3593020

[4] Alauthman, M., Alshammari, R., Alqatawna, J.: A comprehensive study of identity theft attacks and countermeasures in the digital era. IEEE Access **12**, 55712–55734 (2024). https://doi.org/10.1109/ACCESS.2024.3378125, https://doi.org/10.1109/ACCESS.2024.3378125

[5] Alharbi, O., Alsmadi, I.: Semantic matching of url entities and webpage content for phishing detection. Computers & Security **118**, 102739 (2022). https://doi.org/10.1016/j.cose.2022.102739, https://doi.org/10.1016/j.cose.2022.102739

[6] Almeida, R., Westphall, C.: Détection heuristique de phishing et méthodologie de vérification d'URL basée sur le scraping et l'exploration du Web. In: 2020 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 1–6. IEEE (2020)

[7] Alsadi, A., Alasmary, W.: Https security and ssl/tls certificate validation in modern web applications. Computers & Security **140**, 103087 (2024). https://doi.org/10.1016/j.cose.2024.103087, https://doi.org/10.1016/j.cose.2024.103087

[8] Basit, A., Zafar, M., Anwar, S., Mumtaz, M.: A hybrid feature engineering approach for phishing email detection using machine learning algorithms. IEEE Access **9**, 116553–116563 (2021). https://doi.org/10.1109/ACCESS.2021.3104965, https://doi.org/10.1109/ACCESS.2021.3104965

[9] Basnet, R., Sung, A., Liu, Q.: Content-based phishing email detection. In: IEEE Security and Privacy Workshops. pp. 97–104 (2014). https://doi.org/10.1109/SPW.2014.22

[10] for Cybersecurity (ENISA), E.U.A.: Cybersecurity awareness and practices of european citizens 2023. Tech. rep., ENISA (2023), https://www.enisa.europa.eu/publications/cybersecurity-awareness-and-practices

[11] Do, N.Q., Selamat, A., Krejcar, O., Herrera-Viedma, E., Fujita, H.: Deep learning for phishing detection: Taxonomy, current challenges and future directions. IEEE Access **10**, 36429–36463 (2022)

[12] Europol: Internet organised crime threat assessment (iocta) 2023. Tech. rep., European Union Agency for Law Enforcement Cooperation (2023), https://www.europol.europa.eu/publications-documents/internet-organised-crime-threat-assessment-iocta-2023

[13] Glezakou, K., Boiko, I., Markopoulos, P.: Efficient html structure parsing and link extraction for web mining applications. Information Processing & Management **59**(6), 103029 (2022). https://doi.org/10.1016/j.ipm.2022.103029, https://doi.org/10.1016/j.ipm.2022.103029

[14] Gonzalez, R., Vargas, D., Soto, A.: Incremental knowledge base construction for adaptive ai systems. Knowledge-Based Systems **263**, 110249 (2023). https://doi.org/10.1016/j.knosys.2023.110249

[15] Guedes, I., Martins, M., Cardoso, C.S.: Exploring the determinants of victimization and fear of online identity theft: An empirical study. Security Journal pp. 1–26 (2022)

[16] Jalil, S., Usman, M., Fong, A.: Highly accurate phishing URL detection based on machine learning. Journal of Ambient Intelligence and Humanized Computing **14**(7), 9233–9251 (2023)

[17] Jalil, Z., Arshad, J., Shah, A., Farid, S.: Highly accurate phishing url detection using url structure and machine learning. Computers & Security **127**, 103074 (2023). https://doi.org/10.1016/j.cose.2023.103074, https://doi.org/10.1016/j.cose.2023.103074

[18] Liu, M., Chen, H., Zhao, L.: Semantic correlation between url structure and web content for phishing detection. Expert Systems with Applications **206**, 117857 (2022). https://doi.org/10.1016/j.eswa.2022.117857, https://doi.org/10.1016/j.eswa.2022.117857

[19] Lopez, M., Ghavimi, B., Ma, Y.: Entity extraction from multimodal data streams for cybersecurity applications. ACM Transactions on Information Systems **41**(1), 1–26 (2022). https://doi.org/10.1145/3549559, https://doi.org/10.1145/3549559

[20] Mihai, D., Popescu, G., Ionescu, A.M.: The impact of the COVID-19 pandemic on banking digitalization: An analysis of digital payments and digital bank revenues. Journal of Digital Economy (2025), https://www.researchgate.net/publication/394301660_The_Impact_of_the_COVID_Pandemic_on_Banking_Digitalization_An_Analysis_of_Digital_Payments_and_Digital_Bank_Revenues, online first

[21] Mohammad, R., Thabtah, F., McCluskey, L.: Phishtank: A phishing repository and detection system. Expert Systems with Applications **42**(3), 150–163 (2015). https://doi.org/10.1016/j.eswa.2014.08.003

[22] Moura, G.C.M., et al.: Understanding the role of dns security in modern web infrastructure. ACM Transactions on Internet Technology **21**(4), 1–27 (2021). https://doi.org/10.1145/3454127, https://doi.org/10.1145/3454127

[23] Nayak, A., Timmapathini, H.P.: Llm2kb: Constructing knowledge bases using instruction-tuned, context-aware large language models. In: Proceedings of the ISWC Workshop on Knowledge Base Construction. pp. 58–65 (2023)

[24] Nonaka, I., Takeuchi, H.: The wise company: How companies create continuous innovation through knowledge-based management. California Management Review **63**(3), 5–27 (2021). https://doi.org/10.1177/00081256211005359

[25] Ntsama, J.E., Tchakounté, F., Ajibesin, A.A., Wajiga, G.M., Garba, E.J., Fachkha, C., Ahmadou, F.: A comprehensive review of online gaming: trends, impacts, and future opportunities. In: International Conference on Safe, Secure, Ethical, Responsible Technologies and Emerging Applications. pp. 299–315. Springer (2024)

[26] Ntsama, J.E., Tchakounte, F., Tchakounte Tchuimi, D., Faissal, A., Fotso Kuate, F.A., Effa, J.Y., Udagepola, K.P., Atemkeng, M.: Determinants of cybercrime victimization: Experiences and multi-stage recommendations from a survey in cameroon. In: International Conference on e-Infrastructure and e-Services for Developing Countries. pp. 317–337. Springer (2022)

[27] Oikonomou, V., Papageorgiou, C., Zacchia, P.: Digitalization during the COVID-19 crisis. IMF Staff Discussion Note SDN/2023/003, International Monetary Fund (2023). https://doi.org/10.5089/9798400253165.006, https://www.imf.org/en/Publications/SDN/Issues/2023/03/24/Digitalization-During-the-COVID-19-Crisis-531298

[28] Pandey, A., Kumar, S., Singh, R.: Hybrid deep learning approach for phishing url detection using lexical and host-based features. Expert Systems with Applications **224**, 119988 (2023). https://doi.org/10.1016/j.eswa.2023.119988, https://doi.org/10.1016/j.eswa.2023.119988

[29] Rezaei, M.: Artificial intelligence in knowledge management: Emerging trends and challenges. Technological Forecasting & Social Change **195**, 122345 (2025). https://doi.org/10.1016/j.techfore.2024.122345, https://doi.org/10.1016/j.techfore.2024.122345

[30] Roepke, R., Drury, V., Meyer, U., Schroeder, U.: Exploring and evaluating different game mechanics for anti-phishing learning games. International Journal of Serious Games **9**(3), 23–41 (2022)

[31] Sahoo, D., Liu, C., Hoi, S.: Malicious url detection using machine learning: A survey. ACM Computing Surveys **50**(3), 1–39 (2017). https://doi.org/10.1145/3046673

[32] Salahdine, F., Kaabouch, N.: Social engineering attacks: A survey. Future Internet **11**(4), 89 (2019)

[33] Saracino, A., Coppolino, L., D'Amore, A., Spagnuolo, C.: Hybrid malware analysis frameworks: Advances, challenges, and future directions. Computers & Security **140**, 103079 (2024). https://doi.org/10.1016/j.cose.2024.103079, https://doi.org/10.1016/j.cose.2024.103079

[34] Sharma, A., Kaur, N., Singh, H.: Phishing detection using url and content-based features: A comprehensive study. Information Sciences **608**, 659–677 (2022). https://doi.org/10.1016/j.ins.2022.06.039, https://doi.org/10.1016/j.ins.2022.06.039

[35] Stamos, D., Valsamidis, S., Apostolou, D.: Applying IS-enabled telework during the COVID-19 lockdown: Evidence from the greek banking sector. Administrative Sciences **14**(2), 35 (2024). https://doi.org/10.3390/admsci14020035, https://www.mdpi.com/2076-3387/14/2/35

[36] Sundaram, S.S., et al.: Structured knowledge base enhances effective use of large language models for domain-specific qa. Information Processing & Management **62**, 103029 (2025). https://doi.org/10.1016/j.ipm.2025.103029, https://doi.org/10.1016/j.ipm.2025.103029

[37] Tchakounté, F., Faissal, A., Atemkeng, M., Ntyam, A.: A reliable weighting scheme for the aggregation of crowd intelligence to detect fake news. Information **11**(6), 319 (2020)

[38] Tchakounte, F., Ngnintedem, J.C.T., Damakoa, I., Ahmadou, F., Fotso, F.A.K.: Crawl-shing: A focused crawler for fetching phishing contents based on graph isomorphism. Journal of King Saud University-Computer and Information Sciences **34**(10), 8888–8898 (2022)

[39] Verma, P., Goyal, A., Gigras, Y.: Email phishing: Text classification using natural language processing. Computer Science and Information Technologies **1**(1), 1–12 (2020)

[40] Wang, Q., Zhang, W., Li, J.: Detecting phishing websites through url–content consistency analysis. IEEE Access **11**, 87654–87668 (2023). https://doi.org/10.1109/ACCESS.2023.3298472, https://doi.org/10.1109/ACCESS.2023.3298472

[41] Wu, C.Y., Kuo, C.C., Yang, C.S.: Un système de détection de phishing basé sur le machine learning. In: 2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA). pp. 28–32. IEEE (2019)

[42] Yaseen, Q., et al.: Détection des courriers indésirables à l'aide de techniques de deep learning. Procedia Computer Science **184**, 853–858 (2021)

[43] Yeoh, W., Huang, H., Lee, W.S., Jafari, F.A., Mansson, R.: Simulated phishing attack and embedded training campaign. Journal of Computer Information Systems **62**(4), 802–821 (2022)

[44] Zhang, L., Wang, Q., Chen, Y.: Entity-aware phishing url detection using url–text consistency. IEEE Access **11**, 123456–123470 (2023). https://doi.org/10.1109/ACCESS.2023.3341123, https://doi.org/10.1109/ACCESS.2023.3341123

[45] Zhang, W., Li, C., Zhou, Z.: Natural language understanding meets machine learning: A unified framework for intelligent information processing. Expert Systems with Applications **207**, 118067 (2022). https://doi.org/10.1016/j.eswa.2022.118067

[46] Zhang, X., Park, J., Zhou, F.: Crawling and scraping for cybersecurity intelligence. Information Sciences **512**, 1172–1187 (2020). https://doi.org/10.1016/j.ins.2019.11.028

[47] Zhu, Q., Wu, Z., Chi, J., Chen, K.: A survey of static and dynamic malware analysis techniques. IEEE Access **10**, 81694–81716 (2022). https://doi.org/10.1109/ACCESS.2022.3195680, https://doi.org/10.1109/ACCESS.2022.3195680