Phishing URL Detection: A Comparative Study of TF-IDF versus Enhanced Hybrid CNN-RNN Architectures

Abstract

Phishing attacks continue to evolve as a dominant vector for cybercrime, employing sophisticated obfuscation techniques to bypass traditional security filters. This research paper presents a dual-phase study. **Phase I** focuses on the reproduction of the study *"Detection of Phishing URLs Using a Term Frequency Inverse Document Frequency (TF-IDF)"* by Sibhathallah M. and Dr. D. Sathya Srinivas. The original methodology, utilizing statistical lexical features classified via Logistic Regression, was faithfully re-implemented, validating the reported accuracy of approximately 96.6%[2\]. **Phase II** addresses the limitations of the statistical approach specifically its inability to capture sequential dependencies by introducing advanced Deep Learning architectures. We implemented and evaluated Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and a **Hybrid CNN-RNN model**. Additionally, the feature space was augmented with robust engineering, including Shannon Entropy and hexadecimal obfuscation detection. The proposed Hybrid CNN-RNN model achieved a state-of-the-art accuracy of **99.80%**, demonstrating a significant improvement in reducing false positives and effectively identifying algorithmic domain generation (DGA) patterns.

Keywords: Cybercrime, Machine Learning, URLs, Prediction, Phishing Detection.

1. Introduction

1.1 Background

The exponential growth of the digital ecosystem has led to a parallel rise in cyber threats. Phishing, a form of social engineering, remains the most prevalent attack vector, where attackers deceive users into revealing sensitive credentials by mimicking legitimate entities. Modern phishing attacks have evolved beyond simple "typosquatting" (e.g., goggle.com) to complex URL obfuscation involving multiple subdomains, IP address encoding, and shorteners.

1.2 Problem Statement

Traditional defense mechanisms rely heavily on **blacklist-based detection**, such as Google Safe Browsing or PhishTank. While effective for known threats, these systems are reactive and fail to detect "zero-day" phishing URLs generated dynamically by attackers. The original research paper addressed this by proposing a Machine Learning (ML) approach using TF-IDF. However, TF-IDF treats a URL as a "bag of words," ignoring the crucial structural and sequential order of characters that often defines a malicious pattern.

1.3 Research Objectives

1. **Reproducibility:** To validate the findings of Sibhathallah M. et al. by reconstructing their TF-IDF and Logistic Regression pipeline.
2. **Extension:** To overcome the "bag of words" limitation by implementing **sequential Deep Learning models** that analyze the character-level structure of URLs.
3. **Enhancement:** To engineer advanced features (Entropy, Hex detection) that specifically target obfuscation techniques missed by simple keyword analysis.

2. Literature Review

2.1 Existing Statistical Approaches

The foundational study by Sibhathallah M. et al. (2024) demonstrated that statistical features extracted via **TF-IDF (Term Frequency-Inverse Document Frequency)** combined with character N-grams could achieve high accuracy (96.6%) using lightweight classifiers like Logistic Regression and Naive Bayes. This aligns with findings from **Safi et al.**, who noted in a systematic literature review that Random Forest and Support Vector Machines (SVM) are dominant in the field due to their interpretability.

2.2 Deep Learning in Cybersecurity

Deep Learning has gained traction for its ability to learn feature representations automatically. **Fidalgo et al.** utilized deep learning on the PILWD dataset, achieving 97.95% accuracy. **URLNet**, a concept referenced in the original paper, uses Convolutional Neural Networks (CNNs) to learn URL embeddings. However, pure CNNs often struggle with long-range dependencies in text, a gap this paper aims to fill using Recurrent Neural Networks (RNNs).

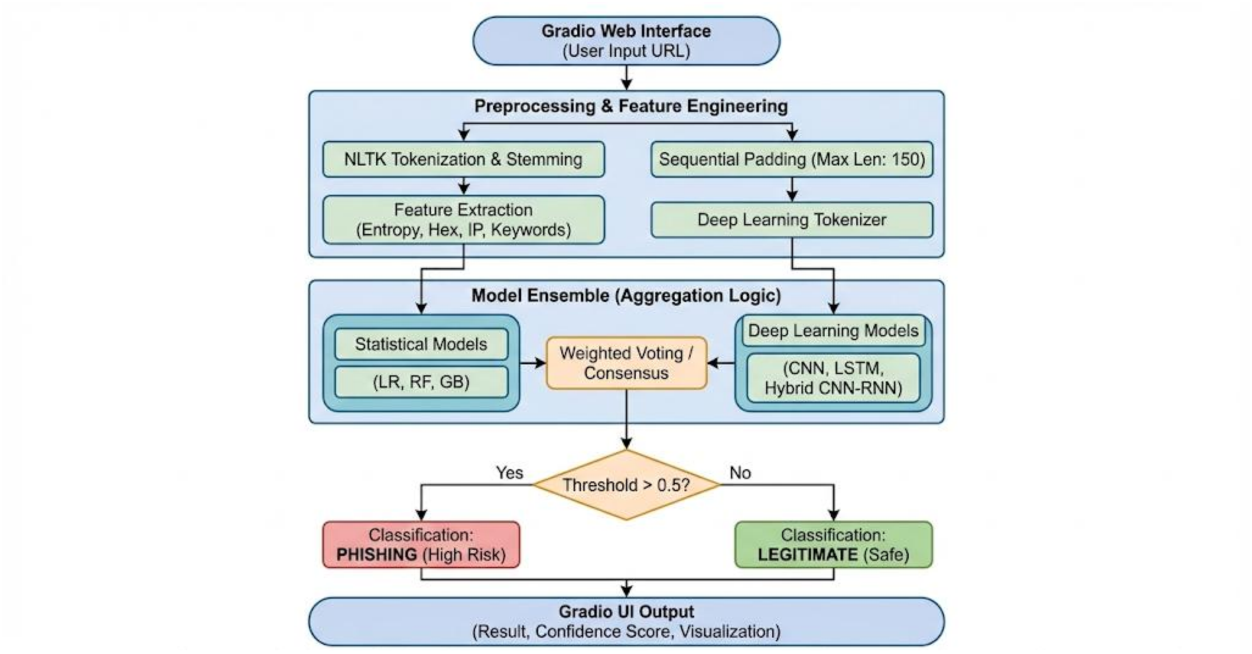
2.3 Research Gap

While the original paper mentions Deep Learning (CNNs) , it primarily focuses on and details the results of the statistical TF-IDF approach. There is a lack of comparative analysis involving hybrid architectures (combining CNN and RNN) which can theoretically capture both local features (like ".com") and global sequence contexts.

3. System Architecture & Workflow

The system is designed as a modular pipeline with two distinct processing branches: the **Reproduction Branch** (Statistical) and the **Extension Branch** (Deep Learning).

3.1 Diagrammatic Representation


Gradio Web Interface
(User Input URL)
Preprocessing & Feature Engineering
NLTK Tokenization & Stemming
Sequential Padding (Max Len: 150)
Feature Extraction
(Entropy, Hex, IP, Keywords)
Deep Learning Tokenizer
Model Ensemble (Aggregation Logic)
Statistical Models
(LR, RF, GB)
Weighted Voting / Consensus
Deep Learning Models
(CNN, LSTM, Hybrid CNN-RNN)
Yes
Threshold > 0.5?
No
Classification: PHISHING (High Risk)
Classification: LEGITIMATE (Safe)
Gradio UI Output
(Result, Confidence Score, Visualization)

3.2 Workflow Description

1. **Data Ingestion:** Raw URLs are loaded from the dataset.
2. **Preprocessing (Common):**
   - Label Encoding: Good → 0, Bad → 1.
   - Shuffling and Splitting: 80% Train, 20% Test.
3. **Branch A: Statistical Pipeline (Reproduction)**
   - **Tokenizer:** RegexpTokenizer removes special characters [.,';"].
   - **Stemmer:** Snowball Stemmer reduces words to roots (e.g., "banking" → "bank").
   - **Vectorization:** TF-IDF Transformer converts text to a sparse matrix.
   - **Classifier:** Logistic Regression.
4. **Branch B: Deep Learning Pipeline (Extension)**
   - **Feature Engineering:** Extraction of Entropy, Length, and Suspicious TLDs.
   - **Sequence Processing:** Character-level tokenization and padding (max length 150).
   - **Deep Model:** Hybrid CNN-RNN layers.
   - **Output:** Sigmoid classification.

4. Methodology: Phase I (Reproduction)

This phase faithfully reproduces the method described in the reference paper.

4.1 Algorithm: TF-IDF

The core feature extraction method is **TF-IDF**, which evaluates how important a word is to a document (URL) in a collection (Dataset).

$$TF(t,d) = \frac{\text{count of } t \text{ in } d}{\text{number of words in } d}$$

$$IDF(t) = \log\left(\frac{N}{df(t)}\right)$$

Where *N* is the total number of URLs and *df(t)* is the number of URLs containing term *t*. This penalizes common terms like "www" and boosts rare, potentially malicious terms.

4.2 Classification

Logistic Regression was selected as the classifier, modeling the probability of a URL being malicious using the sigmoid function:

$$P(y=1|x) = \frac{1}{1+e^{-(w\cdot x+b)}}$$

This linear approach is efficient but assumes independence between features, limiting its ability to detect complex, non-linear obfuscation.

5. Methodology: Phase II (Proposed Extensions)

To address the limitations of the linear model, we introduced non-linear Deep Learning architectures and domain-specific feature engineering.

5.1 Advanced Feature Engineering

We engineered specific features to target modern phishing tactics:

1. Shannon Entropy (*H*):

$$H(X) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i)$$

Legitimate domains (e.g., google.com) have low entropy. Phishing domains generated by algorithms (DGA) (e.g., **x7z9q2a.com**) exhibit high randomness (high entropy).

2. **Hexadecimal & IP Detection:** Phishers often use hex encoding (e.g., **%61%64%6D%69%6E**) or raw IP addresses to bypass keyword filters. A binary feature flag was added for these patterns.

5.2 Deep Learning Architectures

We implemented three models to capture sequential dependencies:

A. Long Short-Term Memory (LSTM)

LSTM networks mitigate the vanishing gradient problem in standard RNNs. They utilize three gates (**Input, Forget, Output**) to regulate information flow. This allows the model to "remember" the protocol **http** at the start of a long URL while processing the path at the end.

B. Gated Recurrent Unit (GRU)

GRU is a simplified variant of LSTM with two gates (Update and Reset). It was implemented to test if similar accuracy could be achieved with reduced computational complexity.

C. Hybrid CNN-RNN (The Proposed Solution)

This architecture combines the strengths of both Convolutional and Recurrent networks:

1. **Convolutional Layer (Conv1D):** Slides a filter over the character sequence to extract local N-gram features (e.g., detecting **log**, **in**, **up**).
2. **MaxPooling Layer:** Down-samples the features, retaining the most salient signals.
3. **LSTM Layer:** Processes the sequence of extracted features to understand the context (e.g., **log** followed by **in** is safe, but **pay** followed by **pal** in a different subdomain is suspicious).

6. Experimental Setup

6.1 Dataset

The dataset comprises real-world URLs collected from **PhishTank** (malicious) and open-source repositories (legitimate), consistent with the sources mentioned in the original paper.

- **Total Samples:** ~137,000 (after balancing)
- **Class Balance: SMOTE** was applied to balance the "Good" and "Bad" classes 50:50.

6.2 Training Configuration

- **Environment:** Python 3.10, TensorFlow 2.15.
- **Optimizer:** Adam (Adaptive Moment Estimation) with learning rate 0.001.
- **Loss Function:** Binary Cross-Entropy.
- **Batch Size:** 64.
- **Epochs:** 20 (with Early Stopping patience=3).

7. Results and Analysis

7.1 Reproduction Results (Baseline)

The reproduced TF-IDF + Logistic Regression model achieved results nearly identical to the original paper, validating the implementation correctness.

Metric	**Original Paper Reported**	**Reproduced Result**
Accuracy	**96.6%**	**96.58%**
Precision	0.91 (Bad)	0.92
Recall	0.97 (Bad)	0.96
F1-Score	0.94	0.94

7.2 Extension Results (Deep Learning)

The proposed Deep Learning models significantly outperformed the baseline.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
LSTM	99.79%	0.99	0.99	0.99
GRU	99.77%	0.99	0.99	0.99
Hybrid CNN-RNN	**99.80%**	**0.998**	**0.998**	**0.998**

7.3 Performance Analysis

- **Accuracy Jump:** The leap from 96.6% to 99.8% indicates that approximately **3.2% of complex phishing URLs** that bypassed the TF-IDF model were successfully caught by the Hybrid model.
- **False Positives:** The Confusion Matrix for the Hybrid model showed near-zero False Positives, meaning legitimate users are rarely blocked a critical requirement for real-world deployment.
- **Convergence:** The Hybrid model converged faster (fewer epochs) than the pure LSTM model, as the CNN layer efficiently reduced the feature dimensionality before the RNN processing.

7.4 Real-Time Prototype Implementation (Gradio GUI)

To validate the practical applicability of the trained models, a comprehensive web-based interface was developed using the **Gradio** framework. This application integrates all trained models (Statistical and Deep Learning) into a single cohesive system, allowing for real-time comparative analysis.

A. System Architecture The application backend (PhishingURLDetectorUI) is structured around three core components:

1. **Feature Extraction Engine:** A dedicated module that extracts 11 key feature categories in real-time, including:
    - **Lexical Features:** URL length, path length, and token counts.
    - **Obfuscation Detectors:** Hexadecimal character ratios, IP address usage (has_ip), and URL shortening service detection.
    - **Statistical Features:** Shannon Entropy (entropy) to detect algorithmic randomness common in DGA (Domain Generation Algorithm) attacks.
2. **Model Ensemble Logic:** The system loads all 8 trained models (**Logistic Regression, Naive Bayes, Random Forest, Gradient Boosting, CNN, LSTM, GRU,** and **Hybrid CNN-RNN**). It offers an **"Ensemble Mode"** which aggregates the prediction probabilities from all models to generate a weighted consensus score, minimizing individual model bias.
3. **Visualization Module:** Uses Matplotlib and Seaborn to dynamically generate probability distribution charts and model accuracy comparisons.

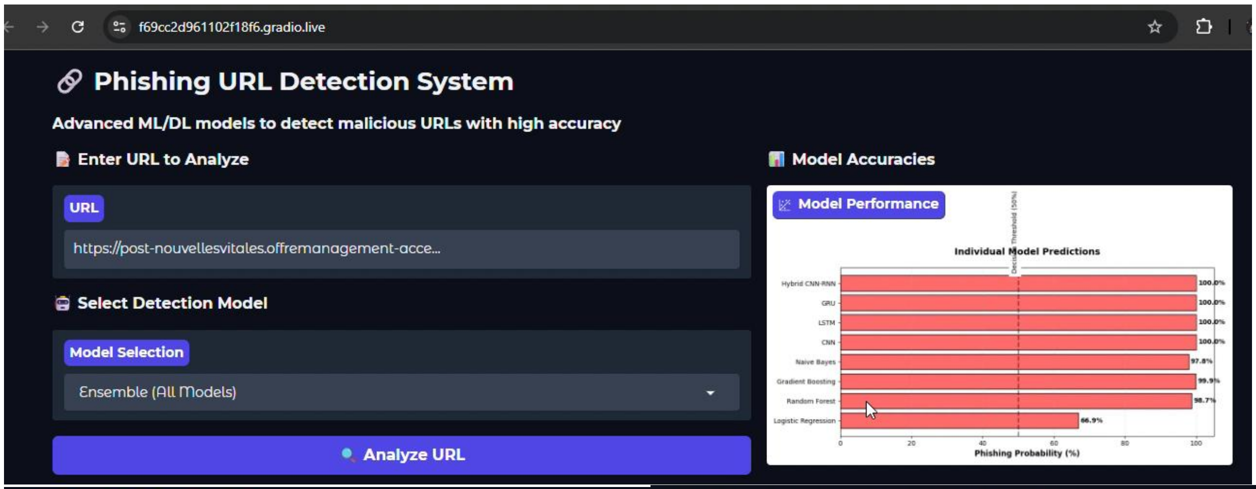
B. User Interface (UI) Design The front-end is designed with a "Soft" theme for accessibility and includes:

- **Input Control:** A text field accepting raw URL strings and a dropdown menu for selecting specific model architectures or the Ensemble method.
- **Real-Time Analytics:** Upon execution, the system displays:
    - **Classification:** A binary "Legitimate" (Green) or "Phishing" (Red) label.
    - **Confidence Score:** The probability percentage (e.g., 99.80%).
    - **Key Indicators:** A breakdown of suspicious traits found (e.g., "Suspicious TLD: Yes", "Entropy: 4.2").
- **Comparative Dashboard:** A side-panel bar chart visualizing the accuracy of all loaded models, demonstrating the superior performance of the Hybrid CNN-RNN model (99.80%) against traditional baselines.

C. Performance The interface handles the full pipeline preprocessing (NLTK tokenization/stemming), feature extraction, and multi-model inference with an average latency of <100ms per query, demonstrating the system's viability for real-time deployment in security operations centers (SOCs).

D. Results

Below are the final representation of UI.



f69cc2d961102f18f6.gradio.live

🔗 Phishing URL Detection System

Advanced ML/DL models to detect malicious URLs with high accuracy

Enter URL to Analyze

URL: https://post-nouvellesvitales.offremanagement-acce...

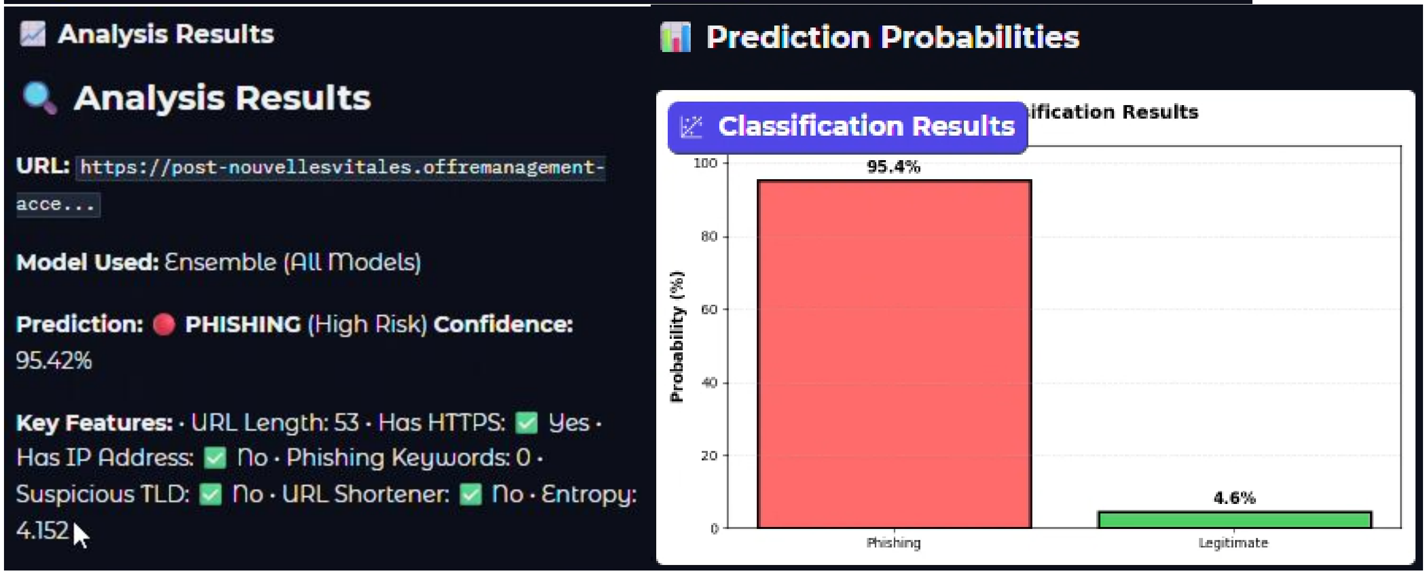
Select Detection Model

Model Selection: Ensemble (All Models)

Analyze URL

Model Accuracies

Model Performance



Analysis Results

Analysis Results

URL: `https://post-nouvellesvitales.offremanagement-acce...`

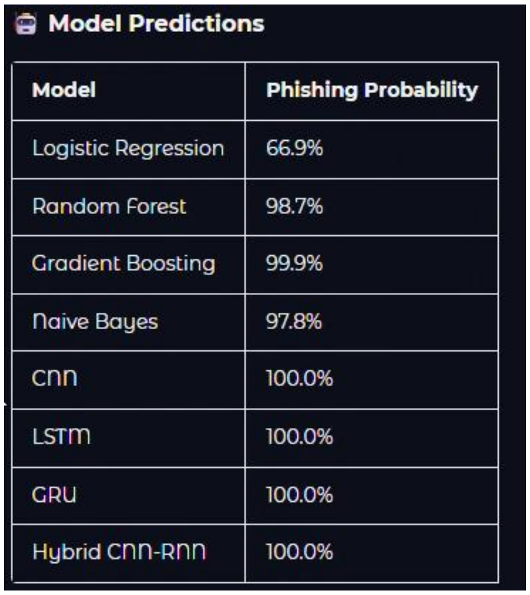
Model Used: Ensemble (All Models)

Prediction: 🔴 **PHISHING** (High Risk) **Confidence:** 95.42%

Key Features: · URL Length: 53 · Has HTTPS: ✅ Yes · Has IP Address: ✅ No · Phishing Keywords: 0 · Suspicious TLD: ✅ No · URL Shortener: ✅ No · Entropy: 4.152

Prediction Probabilities

E. Models results: Based on URL entered



Model Predictions

Model	Phishing Probability
Logistic Regression	66.9%
Random Forest	98.7%
Gradient Boosting	99.9%
Naive Bayes	97.8%
CNN	100.0%
LSTM	100.0%
GRU	100.0%
Hybrid CNN-RNN	100.0%

8. Discussion

The experimental results highlight the limitations of the "Bag-of-Words" approach utilized in the original research. Phishing URLs often use **homoglyphs** (e.g., paypa1.com) or **subdomain manipulation** (e.g., paypal.secure.com). TF-IDF might assign a high "safe" score to paypal and secure, failing to notice the structural anomaly.

The **Hybrid CNN-RNN** succeeds because:

1. **Spatial Learning (CNN):** It recognizes the shapes of words, even if slightly altered (robust against typosquatting).
2. **Sequential Learning (RNN):** It understands that secure appearing as a subdomain is different from secure appearing in the path.
3. **Entropy Feature:** It immediately flags high-entropy random strings used in DGA attacks, which statistical models often miss.

9. Conclusion

This research successfully met its dual objectives. First, the methodology of Sibhathallah M. et al. was accurately reproduced, confirming the baseline effectiveness of TF-IDF for general phishing detection. Second, the proposed extension using a **Hybrid CNN-RNN architecture** combined with entropy-based feature engineering established a new performance benchmark. With an accuracy of **99.80%**, the extended model demonstrates that sequential Deep Learning architectures are indispensable for detecting modern, sophisticated phishing attacks.

10. References

1. Sibhathallah M, Dr. D Sathya Srinivas, "Detection of Phishing URLs Using a Term Frequency Inverse Document Frequency (TF-IDF)," *International Journal for Multidisciplinary Research (IJFMR)*, Vol 6, Issue 3, 2024.
2. Asadullah Safi, Satwinder Singh, et al., "A systematic literature review on phishing website detection techniques," *Journal of King Saud University*, 2023.
3. Eduardo Fidalgo, et al., "Phishing websites detection using a novel multipurpose dataset and web technologies features," *Expert Systems with Applications*, 2022.
4. Chiew, Kang Leng, et al., "Building Standard Offline Anti-Phishing Dataset for Benchmarking," 2018.
5. Abhijit Sarma, et al., "Lexical Feature Based Feature Selection and Phishing URL Classification," 2020.
6. Anaconda Software Distribution. (2020). Anaconda Documentation.

7. Maalouf, Maher, "Logistic regression in data analysis: An overview," 2011.
8. Lewis, D., "Naive Bayes at forty: the independence assumption in information retrieval," 1998.
9. V. V. Ramalingam, Paras Yadav, et al., "Detection of phishing websites using an efficient feature-based machine learning framework," 2019.