A Neural Network-Based Detection of Phishing URLs Using Embeddings

By Amey Bharambe

Abstract

Phishing attacks threaten online security, deceiving users into divulging sensitive information to malicious websites. Currently, as traditional blacklist methods struggle to keep pace with the constantly evolving phishing techniques, machine learning offers an adaptive solution. In this study, we propose creating a neural network-based model to identify phishing URLs using features directly extracted from the URL string. Instead of relying on raw URL tokens or external methods such as website content, our method extracts features related to characteristics of the URL string, including key phrases, and transforms them into dense vector embeddings. These embeddings serve as the input to the neural network, which then teaches complex relationships among feature combinations. This allows the model to generalize beyond external patterns and resist overfitting. Evaluated using the most up-to-date dataset, this method achieved 99 percent accuracy in detecting phishing URLs, outperforming other classical machine learning models.

Introduction

Since the dot-com boom, the widespread use of online websites has introduced new opportunities for cyber-criminals to exploit users. In 2024, phishing attacks emerged as the most significant cyber threat [1]. By disguising malicious websites as legitimate platforms, attackers trick victims into revealing sensitive data such as their credit card information, which can then be misused by the attacker. According to a study by the Anti-Phishing Working Group (APWG), over one million phishing attacks were recorded in 2024 [2]. The high frequency of phishing attacks comes from their simplicity and adaptability, which allows attackers to improve phishing techniques that evade traditional phishing detection systems.

Traditional phishing detection techniques rely on blacklists—databases of previously identified malicious URLs [3]. While these approaches work with known phishing websites, they cannot detect newly generated URLs that are yet to appear in the database. As cyber-criminals adapt by changing URL structure and presentation, traditional phishing detection techniques fall short in offering real-time protection. This is why there is a pressing need for intelligent detection systems that work in real time.

To address this gap, our research explores a machine learning based approach for phishing detection. Specifically, we apply a supervised approach to this binary classification problem, where the model must classify the URL as either phishing or legitimate. Our method focuses on identifying patterns in textual structure from URLs and transforming them into embeddings. These embeddings serve as the input to the neural network.

From the embeddings, the neural network captures patterns in URLs that traditional models or blacklists cannot detect. The output of the classifier model is binary, meaning the

model outputs whether the given URL is phishing or legitimate. Our model is trained and tested on a dataset with modern URLs, achieving an accuracy of 99 percent, which makes it an effective solution to this problem.

Background

Several approaches have been discussed for detecting phishing URLs, each with its strengths and limitations. Early methods have relied mostly on blacklist-based detection, but as mentioned before, the adaptive nature of phishing websites makes such lists insufficient for continuous protection. To address this, modern research focuses on using machine learning to identify patterns in URL structure and webpage content.

One notable approach by Haq et al. (2024) used one-dimensional Convolutional Neural Networks (1D-CNN) to automatically extract features from URLs without the need of physical intervention [4]. 1D-CNNs are neural networks used to process one-dimensional data, essentially extending the use of CNNs for image recognition [5]. While this model has impressive accuracy, the use of a 1D-CNN requires large, labeled datasets and usually overfits if the training data is biased towards certain patterns. This limits their generality to unseen phishing attacks.

Another method proposed by Otieno et al. (2023) used a transformer-based architecture by specifically fine-tuning BERT (Bidirectional Encoder Representations from Transformers) for phishing detection [6]. BERT is a machine learning model that is trained to predict missing words of phrases in a text, and this is used for learning multiple representations of the text [7]. However, due to transformer models being computationally expensive and memory-intensive, they are not a practical solution.

A recent approach by Aslam et al. (2024) proposed AntiPhishStack. It was created using LSTM (Long Short-Term Memory) neural networks and XGBoost classifiers [8]. LSTM neural networks excel at getting output from an input that occurred many steps ago, making it ideal for this type of task. XGBoost classifiers are fast at making predictions [9], so combining it with the LSTM neural network would make it useful for real-world applications. However, creating such an application has high complexity in implementation.

While these models have impressive results, they require complex implementations or a substantial number of resources. Our approach focuses on the use of textual structure from URL strings and transforming them into embeddings. This method enables efficient and accurate phishing detection without extensive resources.

Dataset

For this study, the PhiUSIIL Phishing URL Dataset was used. It was created by the University of California, Irvine, for purposes of benchmarking phishing URL detection models. The dataset contains the latest phishing URLs, reflecting modern phishing attack patterns. There are 57 percent legitimate URLs and 43 percent phishing URLs [10], making it a balanced

dataset. The diversity in the dataset makes it good for the evaluation of machine learning models for real-world usage.

The dataset contains URLs and their corresponding labels: 0 or 1. The zero represents legitimate and the one represents phishing URLs. Unlike raw text, URLs have textual patterns that encode information about the intent of a website. The dataset also provides numerical features, but this project does not focus on these features. Instead, the research focuses on the model learning from the URL's textual structure.

To convert raw URL strings to machine-readable form, we created embeddings using the all-MiniLM-L6-v2 sentence transformers model. It is a pre-trained model that generates dense, semantically meaningful vectors from given text input. Due to it being small, fast, and accurate, it was also a practical choice as we had limited resources.

Using this model, each URL was embedded into a vector of length 384. The embeddings are used to preserve relationships between URLs and string composition, helping the neural network detect patterns without manual feature engineering. This approach also avoids biases that come when selecting features manually.

The dataset was split with an 80/20 split, meaning 80 percent of the data was used for training and 20 percent was used for testing. An 80/20 split was used to ensure enough representation in testing data, so most scenarios are considered. Below were the steps taken for preprocessing:

1. Cleaning: There were no duplicates or null values.
2. Embedding Generation: The URL string was passed through the all-MiniLM-L6-v2 model to create the embedding.
3. Normalization: As all vector outputs were unit-normed vectors, no additional scaling was required.

Methodology

To evaluate the performance of our model, we implemented two other models for comparison: Logistic Regression and Random Forest. Logistic Regression is a classical machine learning algorithm that works by computing the sum of the input features (in our case, the embeddings) and outputs a logistic value. These logistic values form a curve and based on the position of an input relative to the curve, give a probability. The higher the input relative to the curve, the more likely the model outputs one, or in this case, phishing. Random Forest algorithms are also classical machine learning algorithms that work by taking random parts of the dataset to train each tree and combine results by averaging them. To implement these algorithms, the scikit-learn library was used. Scikit-learn is an open-source machine learning library that consists of algorithms for various tasks, ranging from classification to regression.

To create the actual neural network, however, PyTorch was used. The change from scikit-learn to the Pytorch library is because scikit-learn cannot be used for end-to-end training of a model. Figure 1 shows a diagram of the neural network created for this model.

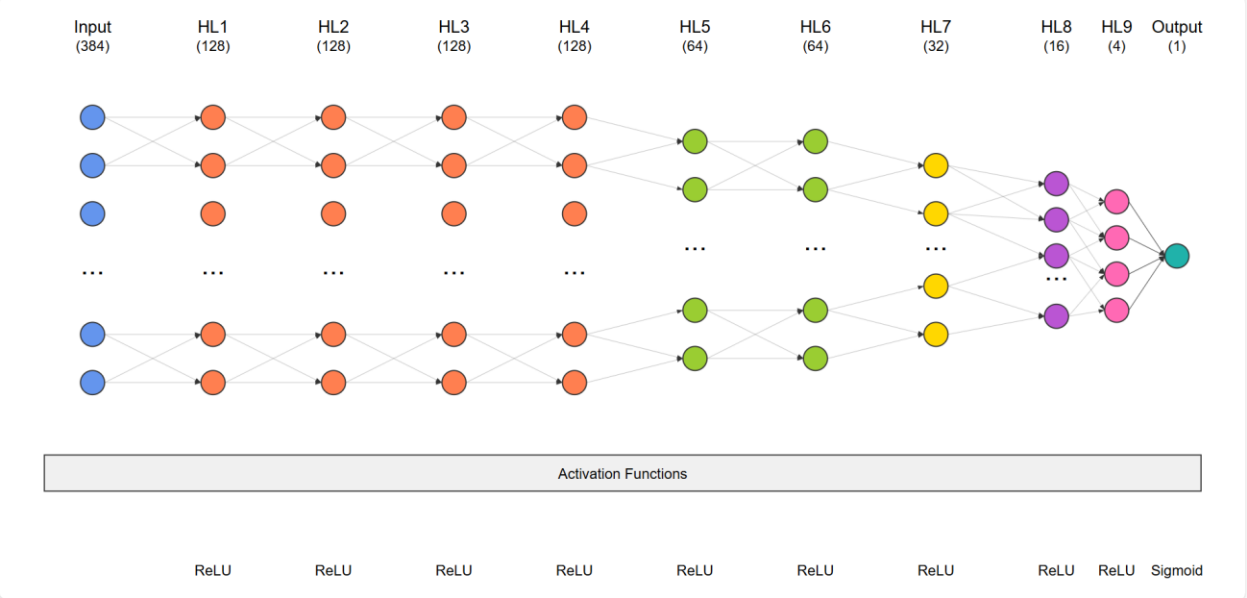

Figure 1: 11-layer neural network visualization

A sigmoid function and only one neuron are used in the output layer because the neural network outputs a probability of the URL being a phishing URL using the embedding. The input layer has 384 neurons because that is the length of the embedding put into the neural network. For the sake of faster training, some hidden layers had the same number of neurons. However, in general, the number of neurons between each layer was divided by a power of two to create a pyramid-like stack. Using a pyramid structure comes with major benefits, such as higher accuracy.

Results

The metrics used for comparison are accuracy, precision, recall, and F1 score. These metrics were chosen for the evaluation of the model because these metrics are used for binary classification tasks such as this one. Our neural network was trained using the Adam Optimizer at a learning rate of 0.001, a batch size of 1024, and binary cross-entropy as the loss function.

Table 1

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	86.6	87	87	86
Random Forest	86.3	88	86	86
Neural Network (Ours)	**99.8**	**99.1**	**98.14**	**98.6**

From Table 1, it is clear that our neural network greatly outperforms other classical machine learning algorithms across all metrics. The 99.8 percent accuracy shows the model's ability to be trusted with its output. The 99.1 percent precision shows the model is giving out a high number of correct phishing predictions. The 98.14 recall rate shows the model is capable of correctly identifying most of the phishing websites. Finally, the 98.6 F1-Score shows that the model is overall reliable with its ability to identify phishing websites.

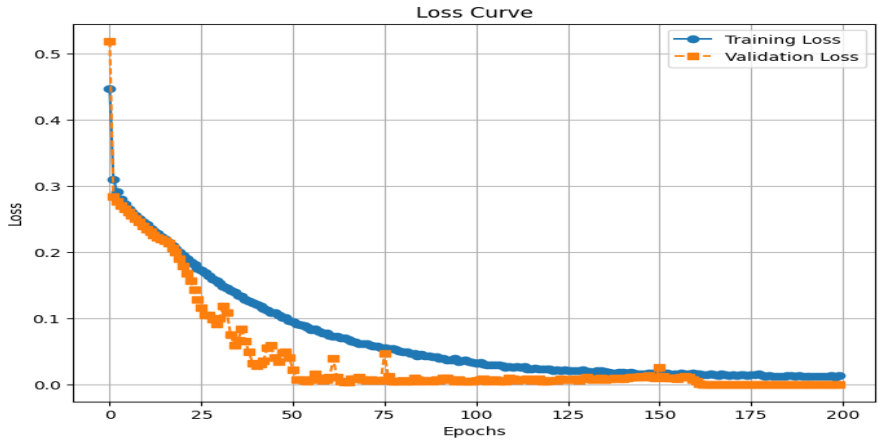

Figure 2: Loss Curve of our neural network over 200 epochs

When creating a model, if a model has a high accuracy, it is important to analyze the loss curves to make sure the model is not memorizing the training data. Figure 2, shown above, compares the loss curves of the training data and test data. As shown in Figure 2, after 200 epochs, both the validation and training loss flatten towards zero loss. The validation curve also never goes above the training curve, meaning there is no overfitting as well. This shows the model is learning from the data and not memorizing.

Finally, Figure 3 shows the confusion matrix highlighting the false positives and false negatives.

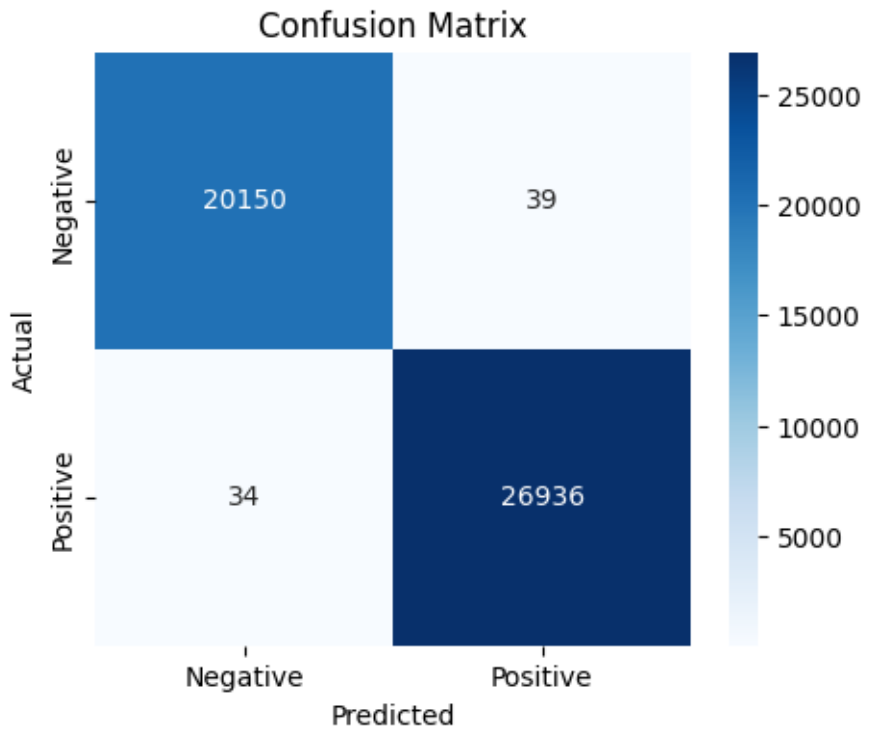

Figure 3: Confusion Matrix of our Neural Network

From Figure 3, it is clear that of the incorrectly identified, more are false positives. A false positive error is better than a false negative error because users would be more alert if a URL were identified as a phishing URL, even if it were not.

Conclusion

The primary objective of this project was to develop a neural network capable of identifying phishing URLs based on the textual structure. Given the rising sophistication in phishing techniques, this study used the PhiUSIIL Phishing URL dataset and embeddings to convert URLs into a numerical form.

Using Pytorch, a custom 11-layer neural network was created, and the performance was benchmarked against classical machine learning algorithms implemented in scikit-learn. The neural network outperformed both baseline models, demonstrating the capabilities of this technique. The pyramid structure allowed for feature compression and abstraction, leading to strong predictive abilities.

In future research, the dataset can be expanded by introducing more diverse URLs and incorporating more features to strengthen this model. Exploring larger embedding models and experimenting with ensemble learning could give better results as well. Additionally, further incorporation into real-world use cases, such as in corporate email gateways, can be done.

Acknowledgements

I would like to thank my advisor, Erick Siavichay, for guiding me through the research publication process. I would also like to thank the University of California, Irvine, for creating this dataset.

References

[1] "Top 10 Biggest Cyber Threats," *Cyber Magazine*, Feb. 5, 2024. [Online]. Available: https://cybermagazine.com/top10/top-10-biggest-cyber-threats https://docs.apwg.org/reports/apwg_trends_report_q4_2024.pdf [Accessed: Apr. 18, 2025]

[2] Anti-Phishing Working Group (APWG), *Phishing Activity Trends Report — 4th Quarter 2024*, APWG, Feb. 2025. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q4_2024.pdf [Accessed: Apr. 18, 2025]

[3] Elsevier, "Phishing Detection," *ScienceDirect Topics*, [Online]. Available: https://www.sciencedirect.com/topics/computer-science/phishing-detection. [Accessed: Apr. 18, 2025].

[4] Q. E. u. Haq, M. H. Faheem, and I. Ahmad, "Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks," *Applied Sciences*, vol. 14, no. 22, p. 10086, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/22/10086. [Accessed: Apr. 18, 2025].

[5] Imperial College London, "5. Convolutional 1D Network Classification," *ReCoDE - AI for Patents*, [Online]. Available: https://imperialcollegelondon.github.io/ReCoDE-AIForPatents/5_Convolutional_1D_Network_Classification/. [Accessed: Apr. 18, 2025].

[6] D. O. Otieno, F. Abri, A. S. Namin, and K. S. Jones, "Detecting Phishing URLs using the BERT Transformer Model," *NSF Public Access Repository*, [Online]. Available: https://par.nsf.gov/servlets/purl/10534600. [Accessed: Apr. 18, 2025].

[7] *TechTarget*, "BERT (Bidirectional Encoder Representations from Transformers) language model," *SearchEnterpriseAI*, [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model#:~:text=Using%20bidirectionality%2C%20BERT%20is%20pretrained,on%20the%20hidden%20word, s%20context. [Accessed: Apr. 18, 2025].

[8] Aslam, M., Arshad, J., Khan, Z., & Khan, I. (2024). "AntiPhishStack: A Two-Phase Stacked Generalization Model for Phishing URL Detection." *arXiv preprint arXiv:2401.08947*. https://arxiv.org/abs/2401.08947 [Accessed: Apr. 18, 2025].

[9] *IBM*, "XGBoost (eXtreme Gradient Boosting)," *IBM Think*, [Online]. Available: https://www.ibm.com/think/topics/xgboost#:~:text=XGBoost%20(eXtreme%20Gradient%20Boosting)%20is,Java%2C%20Scala%20and%20Julia1. [Accessed: Apr. 18, 2025].

[10] P. Muthii and P. Muriuki, "PhiUSIIL: Phishing URL Detection: A Comprehensive Statistical Analysis," *ResearchGate*, Nov. 2024. [Online]. Available: https://www.researchgate.net/publication/386276078_PhiUSIIL_Phishing_URL_Detection_A_Comprehensive_Statistical_Analysis. [Accessed: Apr. 18, 2025].