Phishing Detection using Neural Networks

Daniel DeFlores, Ethan Orevillo

University of Delaware

Abstract

This study compares the effectiveness of convolutional neural networks (CNN) and recurrent neural networks (RNN) in detecting phishing URLs. It examines each model's accuracy, time to process, and number of false positives/negatives on a dataset of over 800,000 URLs. To train the models, each URL was tokenized at the character level before processing. From our results we concluded that for higher accuracy, an RNN should be chosen. However, the processing time for the RNN was nearly four times that of the CNN model, with only slightly better performance. Thus, CNN may be more effective depending on the case.

Introduction

As cyber threats evolve, particularly phishing attacks, the need for advanced detection methods is crucial. This study explores the effectiveness of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in identifying phishing URLs. Phishing, a prevalent form of cybercrime, often deceives users into interacting with malicious links. Our research focuses on a dataset of over 800,000 URLS, with a significant portion being phishing threats. We aim to compare CNN and RNN models in terms of accuracy, processing time, and false positives/negatives, providing insights into the most efficient techniques for phishing detection. This work is especially relevant for those less versed in cybersecurity offering a technological shield against those sophisticated digital traps.

Purpose & Research Question

- We wanted to explore how effective different machine learning models are in recognizing malicious URLs, so that we can leverage our findings to fight cybercrime.
- How effective is a convolutional neural network compared to a recurrent neural network in detecting phishing URLs?

Methodology

Dataset:

We used a dataset of 822,010 URLs, with 394,982 being phishing links, and the rest legitimate. Phishing links were labeled with 0s and legitimate links were labeled with 1s. We padded each URL to 256 characters and tokenized them into numerical values for the models to process. We then split the dataset into training, testing, and validation sets using "train_test_split()" from the "scikit-learn" [1] Python machine learning library.

Experimentation:

We decided that a CNN or RNN would be the best choice of model since URLs don't have defining features. We used the "Keras" [2] Python neural network library for training and testing both models.

The CNN was trained with several layers. An embedding layer to turn tokenized URLs into vectors, a 1D convolution layer to recognize patterns and features of the URLs, a max pooling layer to highlight the most important features, a dense 64 neuron layer with a ReLU activation function to combine and interpret the features, a dropout layer to resist overfitting by assigning some instances a 0, and lastly a dense one neuron layer with a sigmoid function to classify the links. The convolution layer had 32 filters and spanned 3 characters. We trained the CNN for 10 epochs.

The RNN was trained similarly to the CNN. We again added an embedding layer to start. However the core was the long short-term memory (LSTM) layers. The first had 64 units and returned sequences for the subsequent LSTM layer. To prevent overfitting, we included dropout layers with a rate of 0.5 after each LSTM layer. After this, a 64 neuron dense layer with the ReLU activation function interpreted the LSTM features. The final layer was a dense layer with 1 neuron with a sigmoid function to classify the URLs. Like the CNN, we trained for 10 epochs.

Metrics:

Accuracy, processing time, and number of false positives and false negatives are the metrics we chose because they are generally the most important when it comes to a binary classifier model. Accuracy and the amount of false positives and false negatives determine how successful a model is, and give insight to how the model is fit to the data. Processing time is crucial to determine how scalable a model is. This is especially important for our idea to scale one of the models into a real life application

Results

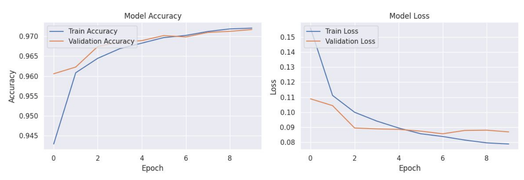

Figure 1: CNN Accuracy and Loss by Epoch

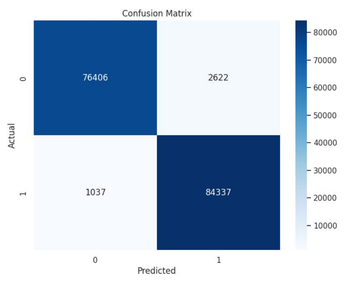

Figure 2: Confusion Matrix of CNN

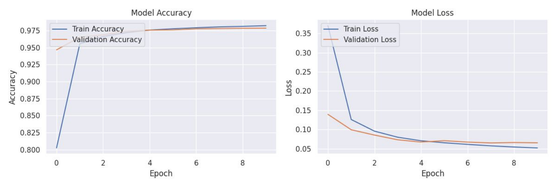

Figure 3: RNN Accuracy and Loss by Epoch

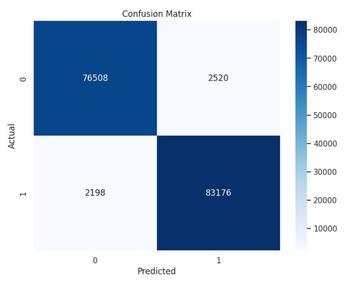

Figure 4: Confusion Matrix of RNN

Conclusions

In conclusion, the employment of machine learning techniques for the detection of phishing URLs presents a significant advancement in bolstering cyber security measures. Our study demonstrates that both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are highly effective in identifying malicious URLs, each offering distinct advantages. CNNs, with their quicker processing times, emerge as a practical choice for real-time applications, balancing efficiency with a high degree of accuracy. On the other hand, RNNs, despite their longer processing durations, provide slightly superior accuracy, making them suitable for contexts where precision is paramount.

This research opens new avenues for the application of machine learning in cybersecurity. The potential for these models to be tailored and enhanced for specific cyber threat detection and tasks is immense. Future exploration could delve into optimizing these models for speed and accuracy, potentially integrating the strengths of both CNNs and RNNs. Such advancements promise to significantly elevate the standards of online safety, offering robust defenses against the constantly evolving landscape of cyber threats.

References

[1] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. Retrieved from http://jmlr.org/papers/v12/pedregosa11a.html

[2] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras