Deep Learning Phishing URL Detection

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Austin Huang

Spring, 2024

Briana Morrison, Department of Computer Science

Deep Learning Phishing URL Detection

CS4991 Capstone Report, 2024

Austin Huang
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
alh2ggp@virginia.edu

ABSTRACT

Phishing is a dangerous form of cyberattack that many are exposed and fall susceptible to each day, leading to identity and financial theft among other troubles. During my 2023 summer internship, as part of a hackathon, a team of four other interns and I developed a model able to distinguish between valid links and fake links intended to phish users. To do this, we built a model that utilized a combination of techniques, such as Recurrent Neural Networks and Language Processing. To combine these models sufficiently, we examined the effectiveness of each individually and developed a voting mechanism to make a final decision. When trained and tested on a small dataset, our model achieved a high F1-score, but was a rudimentary solution when compared to current industry tools. Next steps for the continuation of this project would be implementing more efficient techniques for preprocessing web links and possible exploration of scalable techniques that industry tools use today.

1. INTRODUCTION

A Forbes article estimates that more than 300,000 people in the US were victims to phishing in 2022, accounting for a loss of 52 million dollars alone (Main, 2023). Phishing is the most common form of cybersecurity attack. Usually, those with malintent distribute a link by email or other way of communication to attempt to trick the end user to click on it. These links have a variety of unfortunate consequences, such as stealing user data, identity fraud, spreading viruses, or downloading malware or spyware. Sometimes a convincingly mimicked email disguises the link and at times the link itself will have a similar name to the intended link, maybe a letter off. Thus, for a human eye, detecting whether a link is malicious or not, is difficult, just as it is designed to be. Therefore, developing an automated way to detect phishing links that could supplement human detection is an essential next step to combat phishing attacks.

During the summer of 2023, I had the opportunity to intern as a Systems Engineer. From this exposure, I was exposed to a variety of projects, mainly tasked with working with the Veterans Affairs to create a tool to help with their statement of benefits. However, the project I thought was most stimulating was my experience with a team of four other interns coding a Phishing Machine Learning model from scratch as part of an in-house hackathon, competing against other intern teams doing the same. The project, an application of what I had learned in school, introduced me to relevant applications in modern-day industry.

For this hackathon project, our team wanted to leverage various Machine Learning techniques to build a model that would be capable of differentiating valid URL links from phishing URL links after training with a higher accuracy than the other intern teams competing in the hackathon.

2. RELATED WORKS

Using machine learning as an anti-phishing technique is not new and is widely explored already. Because of this, novel developments in areas like training, feature selection, and advanced machine learning algorithms have produced more complex, robust, and better-performing models than our team's solution. With our scale and expertise, we were able to use similar underlying rationales, separate from the advanced techniques in these papers, to build a simpler model fit for the scale of the project. I showcase two related works that yielded good results and reinforces our team's model selection and method of parsing data.

Gupta, et al. (2021) wanted to develop a machine learning phishing detection model with the goal of time efficiency. In their research, they looked at a dataset of almost 20,000 URLs and extracted nine lexical features to train varying types of models including Support Vector Machines (SVMs), Random Forests, Logistic Regression, and K-nearest neighbors. When comparing these models as individual classifiers, Gupta and his team found that Random Forests performed the most accurately, achieving 99.57% on their test data (2021). Gupta's feature extraction methodology using normalization and one-hot encoding to standardize potential training bias and transform qualitative data into numbers acceptable as model input informed our team about various ways to parse data (2021). In addition, with Gupta's main finding that Random Forests yielded the highest accuracy, our team moved forward, using that as a basis for the development of our Machine Learning model, expecting more accurate results from Random Forests than from the other models that Gupta's team tested.

In another study by Feng and Yue (2020), they agreed that with just traditional machine learning techniques and URL feature extraction, a high accuracy of phishing detection could be achieved. However, these researchers decided to try using Recurrent Neural Networks (RNNs), a form of deep learning, a new, computationally complex, black-boxed machine learning technique. By investigating four different types of RNNs, Feng and Yue were able to achieve higher than 99% accurate detection from a large dataset of about 1.5 million URLs (2020). Furthermore, the use of RNNs allowed Feng and Yue to pass in the entire URL as input without needing to manually perform feature extraction as needed with other models, such as Random Forests (2020). This research effectively shows that RNNs would be useful in phishing URL detection due to their ability for pattern matching.

3. PROJECT DESIGN

For our dataset, we used a company-provided phishing URL dataset. The dataset was a small 280 kilobyte dataset with 4800 URLs. The four feature columns included were "create_age," "expiry_age," "update_age," and "URL." The "create_age" was a quantitative feature specifying the age of the URL in months. The "expiry_age" was a quantitative feature specifying the time in months until the URL will expire. The "update_age" was a quantitative feature specifying the last time since the domain was updated in days. URL was the main feature, specifying the actual link the other meta-data features were detailing. Finally, each URL had a binary label specifying if the URL was phishing or not, intended to be used to train., in a process called feature extraction.

Next, we analyzed the data for more observations we could include as input into our models. With the URL alone, we were able to extract 12 additional features, extracted from human observation and our own curiosity; there was no mechanism to determine or estimate how effective the inclusion of certain features would be. These 12 extracted features were URL length, contains ".co," contains "hash," duplicate slash, contains spaces, has IP address, contains "https," number of digits, contains "www," contains suspicious words, fake TLD, and TLD popularity.

Many of these features were simple binary features that checked if the URL included a certain trait, such as a duplicate slash, or "www." We used regular expressions to implement these features. Other features required counting, such as with number of digits and URL length. With the more complex features such as contains suspicious words, TLD popularity, and fake TLD, we took an aggregate of the entire training set to identify common patterns and determined the feature based on set comparisons. We also vectorized the URL into the different sections of the URL: scheme, top level domain, secondary domain, and any subdomains or subdirectories to use as input for language processing for uncommon words and as potential input for deep learning.

Anticipating the use of deep learning models, we normalized all features to ensure the scaling was the same across each feature. For our tokenized features such as the scheme and top-level domains, because the options for those are relatively limited, we used one-hot encoding to translate those names to numbers. We split our training data with a 90-10 split with cross validation.

For our preliminary investigation, we first explored how various types of classic machine learning models performed individually, such as linear regression, random forests, support vector machines, and naïve bayes classifiers. For these classic models, we used the scikit-learn library in Python, which made the implementation of these models streamlined and simple. With these models, we were able to estimate the usefulness of our features with correlation matrices and entropy gain. Combined with a brief exploration of other clustering algorithms, like K-means, we were able to determine what features were most deterministic of a URL being phishing. We moved forward with using the best model, the Random Forest Model.

We then investigated deep learning models and their effectiveness in detecting phishing. We first selected convolutional neural networks (CNN) because of their general popularity. To use the CNN, we passed in our vectorized data from before as inputs. We built our CNN using the TensorFlow library, and from experimentation sequentially stacked an embedded layer, a convolution layer, followed by a pooling layer and encapsulating the results with three dense layers. For the method of calculating error for training the CNN, we used binary cross entropy.

Continuing our exploration with deep learning models, we stumbled upon RNNs. RNNs are a different type of neural network capable of pattern matching. Again, we implemented RNN from TensorFlow and built our model similarly to the CNN. Sequentially, we first used an embedding layer, but then followed by two bidirectional layers, then consolidating the results with 3 dense layers, using binary cross entropy to calculate error when training. To settle on the specific layer size and parameters for each layer for both RNN and CNNs, we used search functions that would test a range of parameters and select the best performing values.

Finally, to aggregate the best three results from our three models, we used weighted voting classification for the model to determine if the URL was phishing or not. The three models we used were Random Forest, CNN, and RNN, and the weight assigned to each was proportional to how well they performed on the training data. With ensemble learning, we were able to create a model that yielded the highest training accuracy. This is the model we decided to submit for the hackathon.

4. RESULTS

Our model when run against the test data yielded an F1-score of 97.13%. With this model, we were able to win 2nd place in our hackathon, only having a lower accuracy to one team that was able to use Principal Component Analysis in their feature extraction. This likely led them to find and select better features to use as input for their models.

In our individual models, we found that the best performing was Random Forest, CNN, and RNN. These models yielded 96%, 95%, and 93% accuracy, respectively, when trained and tested individually. Compared to support vector machines, naïve bayes, and logistic regression, which all yielded 90% accuracy. One interesting result was that our Random Forest model performed the best individually, even better than our deep learning models. We suspected that this was due to our not being able to optimize our neural networks to their most optimal set up because of limited time and the time-costliness to train and tune a deep learning model. Additionally, Random Forests inherently aggregate many models and data and improve accuracy with a feature known as bootstrap aggregation, essentially by combining the results of many randomly-generated, slightly differing decision trees. This aggregation takes advantage of there being only one dataset present and reduces risk of overfitting and variance by averaging.

Despite our efforts to create complicated features such as using language processing to detect common, uncommon, and misspelled words, the features that were most significant to detect phishing, according to the correlation model of our regression models and entropy gain of the Random Forest models, were seemingly the simplest ones. By far, the most important feature was URL length, followed by the number of numeric characters, and then create age. This may hint towards phishers better able to create believable URLs that surpass looking for faulty spellings or unusual appearances.

5. CONCLUSION

This summer hackathon project stemmed from a summer internship where I had to be proactive to find opportunities to code. Discovering this hackathon and getting to work on this project was therefore enriching and allowed me to apply to a real-life situation various machine learning techniques I had learned about in school. With a small dataset, we were able to utilize an aggregate of a Random Forest model, a CNN, and an RNN to produce a model that was able to detect phishing with high accuracy. While this project did not create any new breakthrough in machine learning research, it allowed my intern team and me to get an introductory experience as to what tools and methods future endeavors with larger scale data might use. It also raised in me a personal interest in pursuing a deeper knowledge for deep learning.

6. FUTURE WORK

Future work could immediately enlarge the scale of data to consider many more URLs which are more varied and complex to see how the behavior of our model scale in accuracy and computational time. Additionally, further research could investigate more recent models, reproduce, and extend upon the rationale of other's research with model selection and hyper tuning. Furthermore, additional work could be allotted to investigate better areas of feature selection and extraction, which was a main component of the winning team in the hackathon.

Because our model aggregated two deep learning models, a lot of computation was done behind a black box, so no way currently for a human to explain with conventional logic and decision why the deep learning model made its choice. Eventually, one area I would be interested in would be creating a tool that could act as an interface to explain why a model flags a URL as phishing instead of the answer itself to bridge the gap between user and AI, which could extend beyond merely a phishing detection application.

7. ACKNOWLEDGMENTS

I would like to thank my intern team members: Ben Wurster, Jacob Lane, Rhea Goswami, and Esha Karlekar and the organizers of the hackathon: Ali Zaidi and Steffani Silva. Thank you for creating an opportunity for me to learn and enjoy working.

REFERENCES

Feng, T., & Yue, C. (2020). Visualizing and interpreting RNN models in URL-based phishing detection. *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*. https://doi.org/10.1145/3381991.3395602

Gupta, B. B., Yadav, K., Razzak, I., Psannis, K., Castiglione, A., & Chang, X. (2021). A novel approach for phishing urls detection using lexical based machine learning in a real-time environment. *Computer Communications*, *175*, 47–57. https://doi.org/10.1016/j.comcom.2021.04.023

Main, K. (2023). *Phishing statistics by state in 2024*. Forbes. https://www.forbes.com/advisor/business/phishing-statistics/