Information Security Project

Machine Learning Based URL Filtering

By Fahad Kamran

20i-0983

CS-B

Table of Contents

Introduction

The development of machine learning-based URL filtering systems has become crucial in ensuring internet security and protecting users from potentially harmful websites. This report presents the design, implementation, and functionality of a prototype URL filtering system. A comprehensive literature review is included to establish a solid theoretical foundation.

Literature Review

State-of-the-Art in Machine Learning-Based URL Filtering

The state-of-the-art in machine learning-based URL filtering involves leveraging large datasets to train accurate models capable of categorizing URLs into different classes such as Benign, Defacement, Malware, Spam, and Phishing. Recent research papers highlight the importance of robust models in handling a wide variety of URLs and staying resilient against emerging threats.

Challenges in URL filtering include dataset limitations, model interpretability, and adaptability to evolving online threats. Existing solutions often face difficulties in classifying common websites accurately, and there is a need for more diverse and extensive datasets to improve model generalization.

Prototype Design and Implementation

Dataset Selection

The prototype's development initiated with a thorough literature review, leading to the selection of the ISCX-URL-2016 dataset. However, challenges in compatibility with the front-end prompted a transition to neural network models. Subsequent improvements were made by incorporating additional datasets, including Kaggle's malicious URL dataset and the Mendeley Data Dataset of Malicious and Benign Webpages.

Neural Network Training

The neural network model was iteratively trained with fine-tuning to address issues with common website misclassifications. Larger datasets necessitated an enhancement of the model architecture with more layers, neurons, and dropout layers. Optimizations such as larger batch sizes and refined feature counts for TF-IDF vectorization were implemented to mitigate memory issues.

Flask Application Integration

To facilitate user interaction, a Flask application was developed to connect the trained neural network model to a front-end HTML page. Users can input a URL, and upon submission, the system classifies the website into one of the predefined categories, providing a quick and user-friendly interface.

Prototype Functionality

User Input and Output

The prototype allows users to input a URL through the front-end interface. Upon submitting the URL, the system processes it through the trained neural network model and displays the corresponding category on the front-end. The output informs users whether the website is classified as Benign, Defacement, Malware, Spam, or Phishing.

Real-Time Classification

The prototype ensures real-time classification, providing users with immediate feedback on the safety of the entered URL. The neural network's efficiency enables quick decision-making, enhancing the user experience and promoting seamless interaction.

Evaluation

The final model achieved a training accuracy of 97.45%, validation accuracy of 96.97%, and test accuracy of 96.94%. Despite slightly lower accuracy than initial models, the improved model demonstrated enhanced capabilities in classifying URLs outside the original dataset, showcasing its ability to handle real-world scenarios.

Conclusion and Future Directions

In conclusion, the prototype successfully integrates a machine learning-based URL filtering model with a user-friendly front-end interface, ensuring real-time classification and efficient decision-making. Future directions involve continuous model refinement, addressing challenges in common website misclassifications, and exploring additional datasets to enhance model generalization. Ongoing research will contribute to the evolution of URL filtering systems in response to emerging online threats.

Step-by-Step Guide of My Implementation

1. I first researched a few research papers regarding machine learning based URL filtering to find dataset for training. In result I got the ISCX-URL-2016 dataset
2. The ISCX-URL-2016 dataset classifies URLs into 5 categories: Benign, Defacement, Malware, Spam, Phishing.
3. I first trained the model in random forest but there were issues regarding its compatibility with front-end so instead I trained the model on a neural network several times with fine tuning.
4. It resulted in high train-validation-test accuracy but when testing with front-end, there were issues in the results. For example, I tested a very common website ‘google.com’ or ‘youtube.com’ which is supposed to be classified as benign, but the result was phishing. Eyeing my model, and its metric, I concluded the problem was with the dataset being too small.
5. In the next phase I gathered a couple more datasets which can be combined with my current ISCX-URL-2016 dataset (about 167,000 records). I found the Kaggle malicious URL dataset with about 651,000 records more and Mendeley Data Dataset of Malicious and Benign Webpages with about 1.2 million records.
6. With the dataset being larger, I increased my neural network to more layers and more neurons in every receding layer and additional dropout layers, larger batch size for quicker processing, a more refined feature count for tfidVectorizer to cater lack of memory issue.
7. Before training I tried combining data into a single file but with there being such large datasets, there was loss of data in combining, so I left the data files separate. The Mendeley Dataset contained additional information which wasn’t compatible with the initial 2 datasets, so I first modified it before starting my training.
8. After I had my dataset ready, I trained the model several times with a sufficient callback function to avoid unnecessary waste of time and/or overtraining of the model. Which Resulted in 97.45% Training Accuracy, 96.97% Validation Accuracy, 96.94% Test Accuracy. Although the accuracy is

slightly lower than my first couple of models having 98.9% accuracy, they lacked knowledge in classifying URLs outside the dataset properly. Whereas the final model is highly accurate in classifying URLs outside the dataset.

9. Finally, I implemented a Flask application to help connect the model to the front-end HTML page where the user can enter a URL and press Submit to find out if the website is safe to visit or not by displaying the URL category.

References

Brunswick, U. o. (2016). *ISCX-URL-2016 Dataset*. Retrieved from unb.ca: https://www.unb.ca/cic/datasets/url-2016.html

Ma, J., Saul, L., Savage, S., & Voelker, G. (n.d.). *URL Reputation Dataset*. Retrieved from archive.ics.uci.edu: https://archive.ics.uci.edu/dataset/187/url+reputation

Rathore, M. (2016, September 26). *Detecting-Malicious-URLs-Using-Lexical-Analysis*. Retrieved from semanticscholar.org: https://www.semanticscholar.org/paper/Detecting-Malicious-URLs-Using-Lexical-Analysis-Mamun-Rathore/01bb00b24fb2bcf1d11748d0c39ba60367b4c264

Saqib, M. N. (2022). *URL Filtering by Using Machine Learning.* Retrieved from paper.ijcsns.org: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://paper.ijcsns.org/07_book/202208/20220834.pdf

SIDDHARTHA, M. (2021). *Malicious URL Dataset*. Retrieved from Kaggle: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset/

Singh, A. (2020, May 2). *Dataset of Malicious and Benign Webpages*. Retrieved from Mendeley Data: https://data.mendeley.com/datasets/gdx3pkwp47/2

Singh, A. (2020, October). *Malicious and Benign Webpages Dataset Article*. Retrieved from Science Direct: https://www.sciencedirect.com/science/article/pii/S2352340920311987