Varun Kumar Reddy Modugu

plag

Sleep Disorder Paper

Software Defined Networking

SRM Institute of Science & Technology

Document Details

Submission ID

trn:oid:::1:3294372128

Submission Date

Jul 12, 2025, 7:12 PM GMT+5:30

Download Date

Jul 12, 2025, 7:14 PM GMT+5:30

File Name

Report_1.docx

File Size

17.2 KB

6 Pages

873 Words

5,166 Characters

5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

5	Not Cited or Quoted	5%	Matches with neither in-text citation nor quotation marks
0	Missing Quotations	0%	Matches that are still very similar to source material
0	Missing Citation	0%	Matches that have quotation marks, but no in-text citation
0	Cited and Quoted	0%	Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
2%	Publications
0%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **5** Not Cited or Quoted 5%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4%	Internet sources
2%	Publications
0%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	Khumukcham Robindro Singh, Nazrul Hoque, Arnab Kumar Maji, Sabyasachi Mon...	1%
2	Internet	link.springer.com	1%
3	Internet	speakerdeck.com	<1%
4	Internet	www.coursehero.com	<1%
5	Internet	yvm2020.authorea.com	<1%

Malicious URL Classification Using Machine Learning

Submitted by
Bandi Dilli Babu Reddy [RA2211003011441],
Vaibhav Prakash [RA2211003011442],
J. Sai Satvik []

Dept. Of Computing Technologies,
SRMIST.

ABSTRACT

With the rampant increase in phishing, malware, and malicious domains, identification of safe URLs from malicious ones has become critical to cybersecurity. This project proposes a machine learning system for classification of URLs as safe or not_safe according to the text features of URLs. The dataset employed consists of more than 650,000 URLs classified into types such as benign, malware, phishing, and defacement. To simplify and achieve binary classification, all non-benign types are categorized as not_safe.

Applying feature engineering and natural language processing methods, we obtain essential features from every URL, including length, HTTPS presence, IP address format, and dot frequency. A Term Frequency–Inverse Document Frequency (TF-IDF) vectorizer transforms URL strings to numeric vectors. We utilize the Multinomial Naive Bayes algorithm, which has proven strong in text classification, and obtain high accuracy in separating malicious from benign URLs.

The constructed model is incorporated into a friendly-to-use Streamlit web application where users can enter a URL and get real-time classification results. The project is used as a proof of concept for automated filtering of URLs and can be further applied to real-time web filtering and email security products.

Keywords: Malicious URL Detection, Naive Bayes, TF-IDF, Streamlit, Phishing Detection, Text Classification.

INTRODUCTION

The advent of the internet has also opened up new channels for cyber threats. Of the most prevalent and hazardous types is the malicious use of URLs in spreading phishing attempts, malware installation, or in tricking users into surrendering sensitive information. The classic method employed to fight these threats was by blacklisting and manually selected URL filters, but these are not timely nor scalable anymore.

Recent developments in machine learning have made it possible to create systems capable of automatically categorizing URLs according to statistical patterns, lexical attributes, and contextual cues. This project aims to construct such a system employing supervised machine learning to classify safe and unsafe URLs with minimal delay and high precision.

The project employs a labeled dataset that comprises malicious and benign URLs. Following data preprocessing, lexical feature extraction, and vectorization of text, a machine learning model is trained to classify the category of an input URL. The end-to-end system is implemented using Streamlit, which provides an easy-to-use interface for URL safety checks.

This solution is particularly pertinent in real-world applications including browser security extensions, email filtering systems, and enterprise firewall protections.

METHODOLOGY

1. Data Collection and Cleaning

The dataset used is the malicious_phish.csv, which is available on the internet. It contains labeled URL samples categorized as benign, malware, phishing, or defacement. The data was cleaned by removing duplicates and null values.

2. Label Transformation

For simplicity, the original multi-class labels were changed to binary classes:

- "benign" → safe
- All others (malware, phishing, defacement) → not_safe

3. Feature Engineering

- Several features were created from each URL:
- URL Length
- Presence of HTTPS
- Dot count in the URL
- Whether the domain is an IP address (using regex)

4. Text Vectorization

URLs were tokenized and vectorized using TF-IDF (Term Frequency-Inverse Document Frequency) with a max_features limit of 5000. This process converts text-based URLs into a format suitable for machine learning algorithms.

5. Model Training

The dataset was split 80:20 into training and testing sets. The chosen model was Multinomial Naive Bayes because of its efficiency and strength in handling high-dimensional text data.

6. Evaluation Metrics

The model was evaluated using:

- Accuracy Score
- Confusion Matrix
- Precision, Recall, and F1-Score

7. Streamlit Deployment

A web application was created with Streamlit. Users can input any URL, and the app predicts whether it is safe or not_safe based on the trained model.

OPTIMIZATION

5 Various experiments were run to compare various classifiers such as Random Forest, K-Nearest Neighbors, and XGBoost. Naive Bayes gave the optimal balance of speed and accuracy for this specific text dataset. The max_features and the n-gram range of TF-IDF parameters were tweaked to minimize noise and enhance generalization.

MultinomialNB was effectively used, so there was little hyperparameter tuning. Subsequent releases can include GridSearchCV for performance optimization.

RESULTS & DISCUSSION

The last Naive Bayes model obtained more than 95% accuracy on the test set. The confusion matrix revealed hardly any false positives and false negatives, reflecting solid performance for practical use cases.

The Streamlit web application was experimentally tested with a range of phishing and innocent URLs. It delivered real-time predictions at zero latency. This reflects its suitability for use with browser extensions, anti-phishing software, or enterprise security panels.

Although the model at hand considers only lexical features and not host-based or content-based features, it is strikingly effective in classifying by URL string alone.

CONCLUSION

This project showcases a light yet precise method of classifying malicious URLs using machine learning and NLP methods. Using lexical features and Naive Bayes classification, it is highly accurate without the need for costly feature extraction.

Future extensions can involve incorporating extra features such as WHOIS information, domain age, and webpage content analysis. Furthermore, implementing the model in live environments such as email filters or anti-virus software would go a long way in augmenting cyber protection measures.

GitHub Link:
Screen Recording Demo: