International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Spam Filtering for Social Media Context

Saloni Desai[1], Kathan Desai[2], Dilip Dalgade[3]

[1]*B.E. Computer Science, Rajiv Gandhi Institute of Technology, Mumbai University, Maharashtra, India*
[2]*B.E. Computer Science, Rajiv Gandhi Institute of Technology, Mumbai University, Maharashtra, India*
[3]*Assistant Professor, Dept. of Computer Science Engineering, Rajiv Gandhi Institute of Technology, Mumbai University, Maharashtra, India.*

Abstract - *Online social networks have become a popular way for people around the world for them to interact with friends and family members, reading news, discuss events and what not. Users spend a lot of time on these platforms (Facebook, Twitter) storing and sharing their personal information. Due to the increasing use of these networks, there are millions of active users who are then being used to spread malicious content for exploitation. Short URLs have gained immense popularity in the Online Social Networks domain. Malicious URLs is a common and serious threat to cyber security and lots of such URLs are spread via the social networks. In this paper, we have built a supervised machine learning classification model which will help to detect the malicious URLs present in the social networks. We have developed a dataset which provides us with labelled URLs and then with the help of TFID Vectorization and classic machine learning models we distinguish between spam and benign URLs.*

Key Words: **Online Social Networks, malicious URLs, spam, random forest, spam detection, xgboost.**

1. INTRODUCTION

Spam is a real threat to usefulness of the web. Spammers mask their content as useful or relevant and thereby deliver it to the user. The legitimate users consume this spam as authentic content that is relevant to their information needs. In this age of increasing technological usage, there is an increasing burden on social network security administrators to protect the security network management database and system, along with the users from the malicious content that is being spread over the social networks. The most prominent example of the injection of malicious URLs into the OSNs is the Koobface worm [1], Koobface initially spread by using an infected machine to send messages to Facebook friends of infected user, which included a third-party website that infected the machine of the user visiting it by installing malicious software. The worm was effectively executed on a number of OSNs due to the highly interconnected nature of the users. In the recent times, all the social networks have the client server architecture. The OSN service provider acts as the controlling entity. All the content in the system is stored and managed by it. OSN uses online spam filtering which is installed at the OSN service provider. When it gets installed, it inspects the message before transferring it to the recipients and makes the decision whether or not the message should be deleted. If the message is safe it is forwarded to the intended recipients otherwise it is deleted.

In the past, many detection systems have been employed; one of which is the blacklist technique. The problem with this technique is that it cannot identify URLs that have been purposely not included in the list. The hackers take advantage of this time gap between for spreading the malicious URLs and the time required to update the list with the new entries. Our project is focused on overcoming these limitations of the technique. In our model we collect data from different sources available on the Internet by checking its authenticity. These URLs that we are able to collect are differentiated according to their use in order to spread malware over the Web. In order to train the models, we now form a dataset labelling it as either “good” or “bad”.

Our URL detection technique consists of 5 major phases: Data collection- as the name suggests fetches the data in order to form a dataset; furthermore, the dataset undergoes measures in order to avoid undersampling or oversampling; the classifiers are selected; user interface is designed in order to interact with the users.

The rest of the paper is organized as follows. Section 2, reviews the related work. In Section 3, we describe the proposed system that we have designed. In Section 4 and 5, we discuss about the implementation algorithm and results that we have managed to achieve. Section V contains the conclusion and future scope of the work.

2. RELATED WORKS

A lot of research has been done in order to detect a better mechanism. These URLs may be part of scams, phishing, malware campaigns, etc. Machine learning is a well-known technique in order to classify these URLs. This section provides an overview of studies that show different approaches in order to detect malicious URLs.

Some preceding works are based on URL detection schemes which recommended a system which detect malicious websites by verifying lexical features and host-based features. This application is precisely applicable for online algorithms as the size of the training data is bigger than can be effectively processed in batch and because the distribution of features. Prior works relied on batch learning algorithms. But online techniques are far better for two reasons: (1) Online techniques can process huge numbers of examples far more efficiently than batch techniques. (2) Changes in malicious URLs and their features over time can simply be adapted. A study found in 2011 found that HTML

aspects, JavaScript aspects and URL aspects can be used for efficient detection of malicious websites. [2,3]

Another paper [4] focused on detecting fraudsters methods to get users to click on links that are designed to be familiar to the websites they trust or use. They used only URL lexical features. The features were presented as a bag of words after splitting URLs into three text strings: protocol, domain and path. The advantages of this detection are light weight data acquisition and the speed of implementation.

Another paper suggested a detection system based on message features such as interaction history between users, average number of posts containing URLs, average post rate and unique URL number. In OSNs, multiple users are connecting and interacting via the message posting and viewing interface. The system analyses every message and calculates the feature values before rendering the message to the intended recipients and makes immediate determination on whether or not the message under investigation are dropped. [5]

To determine whether a web page was likely to perform a malicious activity, [3] used static analysis of scripts embedded within a web page to classify pages as malicious or benign. [6] developed more interactive dynamic behaviour-based tool to identify malicious web pages, arguing that code-based analysis is more manually intensive, and instead monitored the run time interactions between a web page and the client machine for several minutes. If there were persistent state changes on the client machine, such as new processes created, registry files modified or executable files created, this behaviour was used to signal a malicious exploit. The study actually identified a zero-day exploit of an unpatched java vulnerability that was operating behind 25 malicious URLs, making strong case for dynamic behaviour over static code analysis.

Although the recent surveys have compared several different algorithms along with the different mechanisms, it has failed to elaborate on the important stages in the process while building the model needed for prediction.

3. PROPOSED SYSTEM

The proposed system begins with the data collection stage which is required in order to train the classifiers. Machine learning approaches, use a set of training data and based on the statistical properties, learn a prediction function to classify a URL as malicious or benign. This gives them the ability to generalize to new URLs unlike blacklisting techniques. The primary requirement for training a machine learning model is the presence of training data. In the context of malicious URL detection, this would correspond to set a large number of URLs. The system allows the user to identify between spam or benign URLs on the basis of the predictions made by the models.

Fig -1: Proposed System

3.1 Data collection and labelling

The process begins with forming a dataset in order to train the models with. The dataset that we were able to form consists of nearly 10000 URLs which are classified as either good or bad. The malicious URLs were obtained from a lot of different reliable sources like Phishtank, www.malwaredomainlist.com, etc. In case of building the dataset for clean URLs, we took the help of Github. Thus, by combining data from both the resources, we formed our final dataset. In order to build a model, there is a necessity for clean and well labelled dataset. Therefore, we processed the data set and eliminated any redundant or missing values. The URLs in the dataset are labelled as either 'good' or 'bad'. In order to perform operations on them, we need to convert them into numeric values and therefore bad is labelled as '1' and good is labelled as '0'.

3.2 TFID Vectorization

In information retrieval or text mining, the term frequency-inverse document frequency is a well-known method to evaluate how important a word is to the document. It is a very interesting way of converting the textual representation of information into a Vector Space Model (VSM). The first step in this procedure is to create a dictionary of all the words present in the text. In order to do that you can select the terms and convert it to a dimension in vector space. In addition to that we also need to ignore the stop words i.e the words which are quite frequent in all the documents like "the", "is", "at"," on", etc. After the word removal, we calculate the term frequency which is the measure of how many times the term is present in the vocabulary. The tf-idf differs from tf as it takes under consideration not only the term but also the term within the document collection. What tf-idf then does is to solve the problem by scaling down the frequent terms and scaling up the rare terms.

3.3 Model Selection

There is a rich family of machine learning algorithms in literature which can be used for detection of malicious URLs. There are plenty of classification algorithms which can be used for this purpose. However, there are certain properties of the URLs which make the modelling process a tad bit

trickier and thus machine learning approaches try to analyse the information of a URL and its corresponding websites or webpages. In order to train the model with the designated data set we need to split the data into training and testing data set which can be in the ratio 80:20. Furthermore in order to get accurate results we need to have a dataset which has near equal values of both the classes. Thus, we should make sure that the condition of overfitting as well as underfitting is avoided at any cost.

Decision Tree: It is one of most popular methods for inductive inference and has a major advantage of its highly interpretable decision tree classification models which can also been converted into a rule set for human readability. In the tree building phase, it recursively partitions a dataset using depth-first greedy approach or breadth first approach until all the data items belong to the same class label. In the tree pruning phase, it works for improving the classification accuracy by minimizing overfitting problem. A huge advantage of this algorithm is that it makes open all the likely options and follows each option to its end in one view, giving room for straightforward evaluation among the different nodes of the tree.

Random Forest: Random Forest is a very flexible and easy to use machine learning classifier that consists of a collection of tree structured classifiers. It randomly selects the features to construct a collection of decision trees. It is one of the most widely used algorithms, because of its simplicity and it can be used for both classification and regression tasks. [7].

XG Boost: XGBoost has become a widely used and popular machine learning algorithm. It is an implementation of gradient boosted decision trees. More particularly, it is an ensemble method which sequentially adds predictors and corrects the previous models. However, instead of assigning weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss. [8].

Logistic Regression: Logistic regression is an extension of simple regression method. In logistic regression, the output of linear regression is passed through the activation function. Softmax function can be used as an activation function, it is a very popular function to calculate the probabilities of the events.

When these models are trained using the Jupyter Notebook, they are saved on the disk using the Pickle model available in Python. Python pickle module is used for serializing and de-serializing a Python object structure. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script. Pickle module comes handy to save complicated data. Also, the pickled file generated is not easily readable and thus provide some security.

4. ALGORITHM AND CALCULATIONS

Initialize the prediction function as $w_1 = 0$;
for $t = 1, 2, \ldots, T$ do
  Receive instance: $x_t \in \mathbb{R}^d$;
  Predict $\hat{y}_t = f_t(x_t)(= \text{sign}(w_t^\top x_t)$ for binary classification);
  Receive correct label: $y_t \in \{-1, +1\}$;
  Suffer loss: $\ell_t(w_t)$, which depends on the difference between $w_t^\top x_t$ and y_t;
  Update the prediction function w_t to w_{t+1};
end for

Fig -2: Pseudo Code

In order to decide the accuracy of a model, we performed certain calculations which helped us derive the precision values of all the models.

Precision is the ratio of true level of positive or negative detection of the classifier to overall test samples.

Precision= TP / (TP + FP)

Recall is the ratio of correct true positive classifier decisions to the all true positive examples in the test set.

Recall= TP / (TP + FN)

F-measure (F1) represents the previous metrics precision and recall combined as follows.

F-m= 2 * (precision * recall) / (precision + recall)

Data Set		Classifier decision	
		SPAM	Not SPAM
True	Spam	True Positive	False Negative
	Not SPAM	False Positive	True Negative

Table -1: Confusion Matrix

5. RESULTS AND ANALYSIS

While designing the system, we developed an interface where the user will be able to generate results for a single URL using multiple models. Depending upon the accuracy of the models, the user can then use it for classification purposes. After having trained the models with 80% of the data, we can get the confusion matrix for all the models which help us determine the optimality of each classifier. These matrices are obtained with the help of the Scikit library. But in case of XGboost it is not possible and therefore we use the inbuilt functions that are available to us when we import the xgboost model to get the accuracy percent.

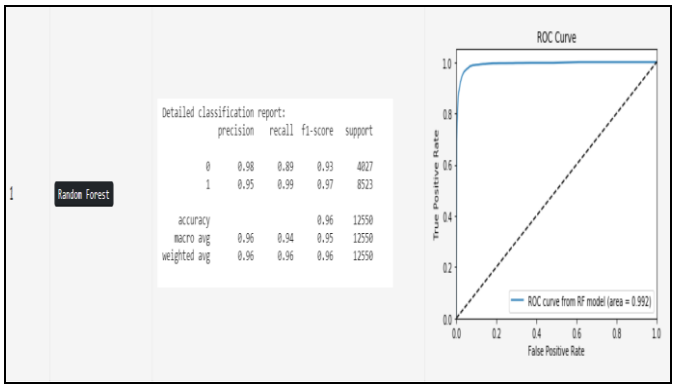

Fig -3: Random Forest Results

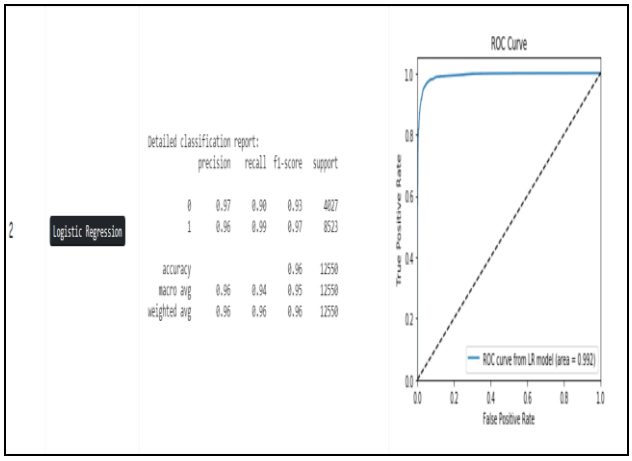

Fig -4: Logistic Regression Results

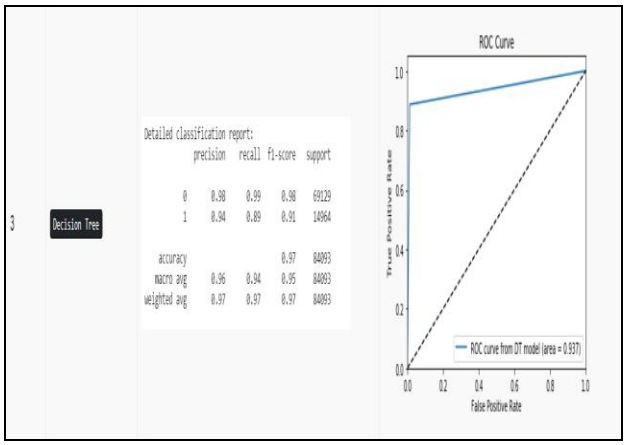

Fig -5: Decision Tree Results

Now since we cannot use these functions to calculate the optimality of XG Boost model, we use accuracy_score function and get the value to be around 85% which is the value that we get by taking all the possible criteria into consideration. In terms of rest of the models, we take a closer look at the precision, recall and f1-scores which then help us evaluate the accuracy of the model. After looking at all the calculated values, Decision tree stands out to be the most promising model with an accuracy of 96%.

These results indicate that when these models are trained and tested with a data of 10000, then we get the accuracy to be around 96%. Similarly, it can be observed that as and when we start increasing the values in the dataset the values fluctuate and the result starts becoming more and more accurate and precise with low false positives and false negatives. Also, in order to bring about consistency in the results, the samples undergo measures in order to prevent the condition of oversampling or undersampling. This also increases the rate of getting optimal results.

6. CONCLUSIONS

Malicious URL detection acts as a crucial milestone for many cyber security applications and the advent of machine learning algorithms combating these threats seems promising. The proposed system was developed in order to distinguish between spam and benign URLs with the help of machine learning. In particular, we ordered a systematic formulation of Malicious URL detection from a machine learning perspective, and then detailed the discussions of existing studies for malicious URL detection, particularly in the forms of developing new feature representations, and designing new learning algorithms for resolving the malicious URL detection tasks. In particular, despite the extensive studies and the tremendous progress achieved in the past few years, automated detection of malicious URLs using machine learning remains a very challenging open problem. Future directions include more effective feature extraction and representation learning (e.g., via deep learning approaches), more effective machine learning algorithms for training the predictive models particularly for dealing with concept drifts (e.g., more effective online learning) and other emerging challenges (e.g., domain adaption when applying a model to a new domain), and finally a smart design of closed-loop system of acquiring labelled data and user feedback (e.g., integrating an online active learning approach in a real system).

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to Dr. Sanjay U. Bokade, Principal and Dr. Satish Y. Ket, H.O.D of Department of Computer Engineering of Rajiv Gandhi Institute of Technology for giving us an opportunity to do our project work on- Spam filtering for Social media context. We sincerely thank our project guide Mr. Dilip Dalgade for his guidance and encouragement in carrying out this synopsis work. Finally, we would like to thank our colleagues and friends who helped us in completing the work successfully.

REFERENCES

[1] K. Thomas and D. Nicol, "The koobface botnet and the rise of social malware in malicious and unwanted software", 2010 5th International Conference, pages 63-70, October 2010.

[2] J. Ma, L. K. Saul, S. Savage, G. M. Voelkar, "Identifying suspicious URLs: An application of large-scale online

learning", Proc. 26th Intl. Conference Machine Learning (ICML), 2009.

[3] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages", Proc. 20th Int'l World Wide Web Conf. (WWW), 2011.

[4] A. Blum, B. Wardman, T.Solorio, G. Warner, "Lexical feature based phishing URL detection using online learning" , Proc. 3rd ACM Work. Artif. Intell.Secur. -AlSec '10, no. August 2016, p.54,2010.

[5] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards Online Spam-Filtering in Social Networks, Proc. 19th Network and Distributed System SecuritySymp. (NDSS), 2012.

[6] Y. min Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King, "Automated web patrol with strider honey monkeys: Finding web sites that exploit browser vulnerabilities", In NDSS, 2006.

[7] L. Breiman, "Random forests", Machine learning, vol. 45, no. 1, pp. 5-32, 2001.

[8] \eXtreme Gradient Boosting (XGBoost)," accessed on 20 April 2018. [On-line]. Available: https://www.kdnuggets.com/2017/ 10/xgboost-top-machine-learning-method-kaggle-explained.html