Comparative Analysis of Malicious Detection of Short URLs from Tweets

[1]Tareek Pattewar, [2]Manish Wagh, [3]Umesh Bisnariya, [4]Lomesh Patil
[1]Assistant Professor, [2 3 4] Students
Department of Information Technology
R C Patel Institute of Technology, Shirpur Dist. Dhule-425405
Maharashtra, India

Abstract : ***Online Social Networks (OSNs) have become fundamental parts of our online lives, and their popularity is increasing at a surprising rate every day. Growing popularity of Twitter has attracted the attention of attackers who attempt to manipulate the features provided by Twitter to gain some advantage, such as driving Twitter users to other websites that they post as short URLs (Uniform Resource Locators) in their tweets. Even short URLs are also used to avoid sharing overly long URLs and save limited text space in tweets. Significant numbers of URLs shared in the OSNs are shortened URLs. Despite of its potential benefits from genuine usage, attackers use shortened URLs to hide the malicious URLs, which direct users to malicious pages. Although, OSN service providers and URL shortening services utilize certain detection mechanisms to prevent malicious URLs from being shortened, research has found that they fail to do so effectively. In this paper, we developed mechanism to develop a machine learning classifier which detect malicious short URLs. And comparative analysis of detection rate with various classification algorithm, and but also with online detection on Virus Total.***

Index Terms - Online Social Networks, twitter, short URLs, malicious URLs, machine Learning, classification

---

I. INTRODUCTION

In the age of web 2.0, information is contained in the form of webpages and shared via Online Social Networks (OSNs), emails, messages, and texts. OSN is a platform to connect and communicate with each other and share knowledge, information, news, announcements, etc. While doing so, people share URLs (Uniform Resource Locator) of the webpages that possess the information. However, URL sharing can be problematic because of its length; some URLs are overly long and complicated. One particular OSN that is very famous as information sharing social network is Twitter [1]. Twitter is a microblogging social network that allows users to post messages up to 140 characters known as tweets. It is difficult to share long URLs because of this restriction. This led to the popularity of URL shortening services on Twitter. URL shortening services take, usually a long URL from users, and create a short URL alias for that long URL. The short URL can then be shared among friends easily. When other users visit the short URL, they will be directed to the shortening services upon which users are redirected to the original long URL Landing page.

There are several techniques to implement a URL shortening. Keys can be generated in base 36, assuming 26 letters and 10 numbers. In this case, each character in the sequence will be 0, 1, 2, ..., 9, a, b, c, ..., y, z. Alternatively, if uppercase and lowercase letters are differentiated, then each character can represent a single digit within a number of base 62 (26 + 26 + 10). In order to form the key, a hash function can be made, or a random number generated so that key sequence is not predictable. Or users may propose their own keys. For example, http://example.com/product?ref=01652type=shirt can be shortened to http://tinyurl. com/exampleshirt. Not all protocols are capable of being shortened as of 2011, although protocols such as http, https, ftp, ftps, mailto, mms, rtmp, rtmpt, ed2k, pop, imap, nntp, news, ldap, gopher, dict and dns are being addressed by such services as URL Shortener. Typically, data: and javascript: URLs are not supported for security reasons. Some URL shortening services support the forwarding of mailto URLs, as an alternative to address munging, to avoid unwanted harvest by web crawlers or bots. This may sometimes be done using short, CAPTCHA-protected URLs, but this is not common.

Makers of URL shorteners usually register domain names with less popular or esoteric Top-level domains in order to achieve a short URL and a catchy name, often using domain hacks. This results in registration of different URL shorteners with a myriad of different countries, leaving no relation between the country where the domain has been registered and the URL shortener itself or the shortened links. Toplevel domains of countries such as Libya (.ly), Samoa (.ws), Mongolia (.mn), Malaysia (.my) and Liechtenstein (.li) have been used as well as many others. In some cases, the political or cultural aspects of the country in charge of the top-level domain may become an issue for users and owners,[4] but this is not usually the case. Tinyarro.ws, and qoiob.com use Unicode characters to achieve the shortest URLs possible, since more condensed URLs are possible with a given number of characters compared to those using a standard Latin alphabet.

II. RELATED WORK

URL shortening services take as input a long URL (e.g. http://blog.bitly.com/post/138381844/spamand-malware-protection) and generates a short URL (e.g. bit.ly/RSwVGo) in return. Short URL so generated redirects to the same long URL but looks random and unrelated to the actual link. Imposed character limit has lead to immense popularity of such services in social media landscape. Due to their ubiquitous usage, these services have been hit by adversaries to obfuscate and disseminate malicious content. This section presents the work done in understanding the usage pattern, behavior, and misuse of short URLs.

1. Malicious Long URL Characterization /Detection

A number of studies have been conducted to understand the propagation of spam on OSM, many of which revealed heavy usage of URLs to spread malicious content. Benevenuto et al. in their research identified distinctive features to detect spammers on Twitter [2]. Researchers also evaluated the effectiveness of popular blacklists in evading spam and observed it to be inefficient. On Twitter, checking blacklists becomes even slower because of the URL shortening services used to obfuscate long URLs. Using these services, a spammer can complicate the process of detection by using chains of multiple shortenings [4]. Ma et al. proposed an approach to detect malicious URLs based on their lexical and host based features and achieved an accuracy close to 95% [5].

Thomas et al. in 2011 developed a system called Monarchwhich classifies a URL submitted to any web service as malicious or benign in real time [1]. This system relies on the features of URL landing page (like page content, hosting infrastructure, etc.) and detected malicious links with an accuracy close to 91%. In year 2012, Aggarwal et al. also proposed a real time phishing detection system for Twitter, called PhishAri [3]. Authors coupled the Twitter and URL based features to classify phishing tweets and achieved an accuracy of 92.52%. In another real time suspicious URL detection technique on Twitter proposed by Lee et al., authors addressed the problem of conditional URL redirects [6]. A combined feature set of correlated URL redirects and tweet context information was used and authors attained an accuracy of 86.3%.

2. Short URL Analysis

With the introduction of short URLs in OSM, a comparative study is necessary to be able to understand the level of acceptance of short URLs over long URLs. Kandylas et al. performed a comparative study of long and short Bitly URLs propagation on Twitter and found that Bitly links received orders of magnitude more clicks than an equal random set of long URLs [5]. To further comprehend short URL distinctive characteristics, Antoniades et al. studied the lifetime of short URLs which revealed that the life span of 50% short URLs exceeds 100 days [3].

Other than this generic analysis, Neumann et al. looked at malicious short URLs in emails and highlighted their privacy and security implications [1]. Chhabra et al. also gave an overview of evolving phishing attacks through short URLs on Twitter and found that phishers use URL shorteners not only to gain space but hide their malicious links [13]. Their results show that most of the tweets containing phishing URLs comes from inorganic (automated) accounts. Later in year 2012, Klien et al. presented the global usage pattern of short URLs by setting up their own URL shortening service and found 80% short URL content to be spam related [2].

In year 2013, Maggi et al. performed a large scale study on 25 million short URLs belonging to 622 distinct URL shortening services [4]. Their results highlight that the countermeasures adopted by these services to detect spam are not very effective and can be easily by-passed. Experimental results from their data shows that Bitly allow users to shorten malicious links and does not include any initial level lightweight check to prevent it (though detects it after some time). Unlike their study which focused on multiple URL shorteners, our research is an in depth analysis of the effectiveness of a single URL shortener. Another scheme was proposed by Yoon et al. in year 2013 about using relative words of target URLs in short URLs [14]. This can give hints to user to guess the target URL, making it comparatively safe from phishing attacks.

3. Malicious Short URL Characterization / Detection

There is little research done in the area of malicious short URL characterization. One such work that presents short URL based features to detect malicious accounts is given by Wang et al. in year 2013 [7]. In their experiment, they investigated the creators of 600,000 short Bitly URLs and associated click traffic from different countries and referrers. Based on the analysis, they classify a link as spam / non-spam using only 3 click traffic based features with maximum accuracy of 90.81%, but ignored all short URLs with zero clicks. Our study on the other hand incorporates all URLs irrespective of their click state. In addition, their results reveal that legitimate Bitly users also generate spam and most clicks on short malicious URLs comes from popular websites. After reviewing the above literature, it is evident that a lot of work has been done in the identification and analysis of malicious URLs. Surprisingly, very little work has been done in analyzing only suspicious short URLs to expose the gaps in security mechanisms adopted by a specific URL shortener. Our work significantly differs from the prior studies, as we focus on understanding in depth, the loopholes in spam detection mechanisms of a URL shortening service. We also propose and evaluate a semi supervised classification framework for spam detection in URL shorteners.

III. IMPLEMENTATION

1. Architecture of The System

Following modules are consist in Architecture.

i. Streaming data collection.
ii. Data cleaning.
iii. Extracting short URLS.
iv. Extracting features for short URLS.
v. Labelling class name, (benign, malicious).
vi. Classification.
vii. Virus Total detection.
viii. Comparing both the results.

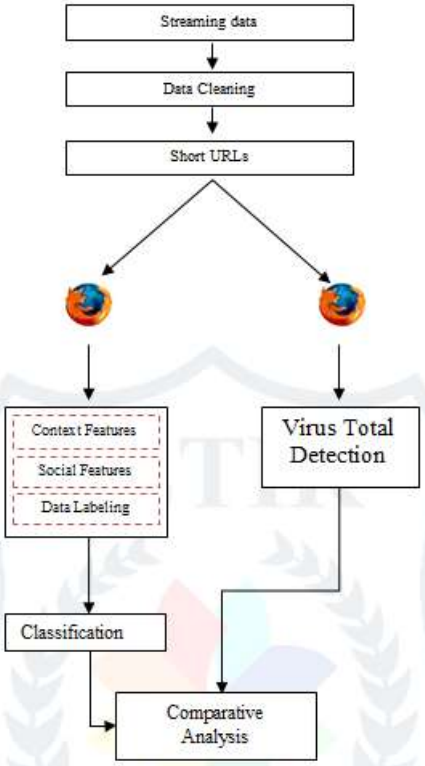

Figure 3.1: Architecture of the system

2. Streaming data collection.(Short URLs)

Streaming HTTP protocol to deliver data through an open, streaming API connection. Rather than delivering data in batches through repeated requests by your client app, as might be expected from a REST API, a single connection is opened between your app and the API, with new results being sent through that connection whenever new matches occur. This results in a low-latency delivery mechanism that can support very high throughput.

2.1 Configure your stream

If you are using a PowerTrack or Replay stream, you need to set up some rules to define what content should be delivered through your stream. You can find out more about the PowerTrack rules syntax by reading our documentation on Premium Operators. After you have configured your stream, you are ready to connect.

2.2 Connect to the API

To open the data stream to have Tweets delivered, you need to send a connection request to the API. In the streaming model, this connection opens up the pipeline for data to be delivered to you as it happens, and will exist for an indefinite period of time. See the documentation for the specific APIs for information on establishing the connection.

2.3 Consume the data as its delivered

Once the connection is established, the stream sends new Tweets through the open connection as they happen, and your client app should read the data off the line as it is received. Your client app will need to recognize and handle various types of messages, which are described in our documentation below.

2.4 When disconnected, reconnect to the API

nevitably, at some point the connection to the stream will close a disconnection. With realtime streams, it is important to understand that you may be missing data any time you are disconnected from the stream. Whenever a disconnection occurs, your client app must restart the process by establishing a new connection. Additionally, to ensure that you do not miss any data, you may need to utilize a Redundant Connection, Backfill, or a Replay stream to mitigate or recover data from disconnections from the stream.

3. Extraction of features and data labeling

The data collection is done 25 days continuesly. After which we collected 2280 short URLS, which consist of 27 character length. To extract the features of this 2280 short urls we used selenium webdriver of firefox webbrowser.

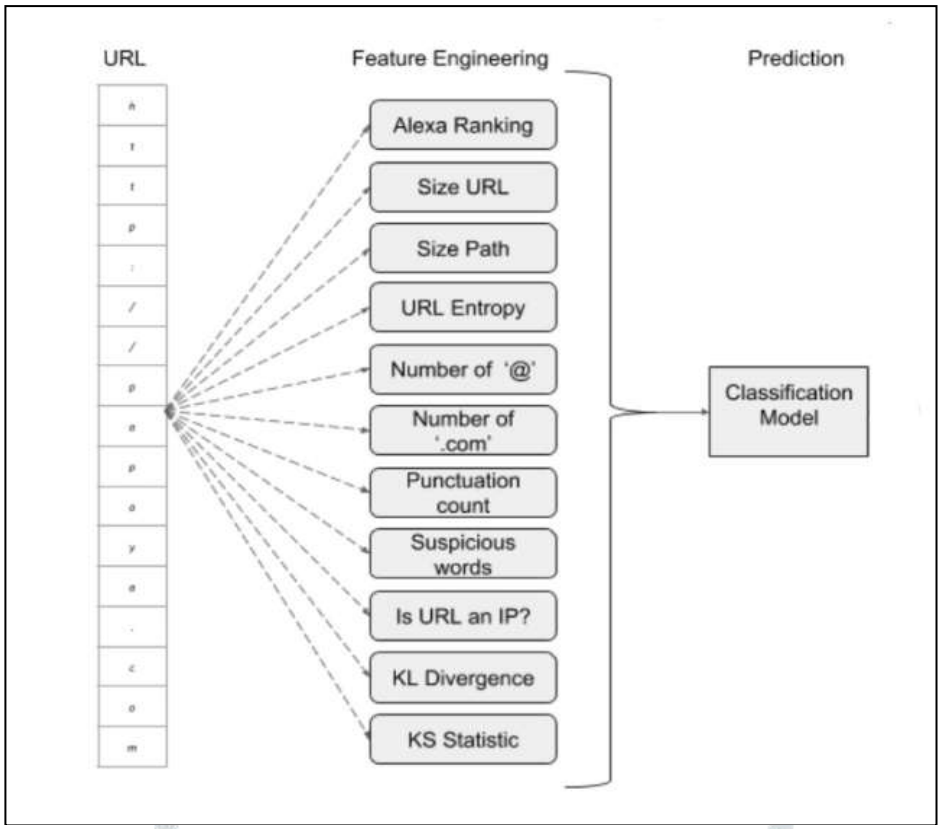

Figure 3.2: Lexical features of malicious URL [12]

Table 3.1: Common URL based features [13]

Common URL Based Features	
Feature	**Example Link**
Redirect Using //	http://imessageaudits.org/profile,mail.buse@example.com
Extremely Long URL	https://docs.google.corn.alpo.edunorms/c1/17,MsBmbra4tvu3VqcXM3hux.n.yuU0B.TK
@ Symbol in the URL	http://imessageaudits.org/profile,mail.buse@example.com
- Separator	http://irstax. wap-kacom/index.x.1
Sub/Multisub Domains	http://www.grandimperial.com.myN2/en/
Nonstandard Port	http://wwwbelcotech.com:320001maillwaithtml
IP Address in the URL	http://194.78.154.195/CFIDE/servicesAabanquepostale.html
HTTPS within URL	http://www.rorna.md/templates/systern/httpshmv2..u.corn.brlatendimentol

Table 3.2: Features collected from URLs

Collected Features			
dots	at	Fulllenght	query
equal	no of a tag	protocol	filename
question	no of img	protocollength	filenamelength
colon	no of link	authority	ref
underscore	no of meta	codepoint	dport
dash	no of script	host	hash code
ampersand	no of div	hostlength	classname
percentage	no of form	port	
dollar	Website	path	
Hash	int count	pathlength	

4. Selection of Features and building Dataset

We obtained features related to short URLs, which are redirected when opened into a browser. We find out 37 features for 2280 short URLs, when they redirected. The differentiation of URLs is done by Benign and Malicious labeling class. We have made balanced dataset from which 1240 URLs are Benign and 1240 URLs are Malicious.

5. Machine Learning Classification

Now we describe the mechanisms used in our classification of malicious short URLs. The experiments involved a 3 step process – i) creating a labeled dataset, ii) training the suitable machine learning classifier, and iii) testing an unlabeled dataset on the trained classifier. In order to assess the most appropriate and efficient mechanism to detect malicious short URLs, we inspected various machine learning classifiers which were best suited for our study. For this, we used the popular classification algorithms implemented in Weka software package [8]. Weka is an open source collection of machine learning classifiers for data mining tasks. Now, we give a brief description about these classifiers.

5.1 Naive Bayes

It is a simple probabilistic classifier based on the Bayes' theorem. It assumes all classification features to be independent of one another and works best when the dimensionality of inputs is very high. An advantage of using this model is that it does not have a large training data requirement for parameter estimation and classification. It uses variance of variables in each class and is also not sensitive to irrelevant features.

Bayesian formula can be written as [12]:

$$P(H|E) = \frac{[P(EjH) * P(H)]}{P(E)}$$

where,

- P(H) is the probability of hypothesis H being true. This is known as the prior probability.
- P(E) is the probability of the evidence(regardless of the hypothesis).
- P(E|H) is the probability of the evidence given that hypothesis is true.
- P(H|E) is the probability of the hypothesis given that the evidence is there. The basic idea of Bayess rule is that the outcome of a hypothesis or an event (H) can be predicted based on some evidences (E) that can be observed from the Bayessrule [13].

5.2 J48

It is a popular classification method that uses decision tree as a predictive model. It uses a rule based approach to observe data features and make inferences about item's target value. Decision Tree starts at the root and makes binary (yes / no) decisions at each level until it reaches the leaf node.

Algorithm [9] J48:
INPUT
D // Training data
OUTPUT
T // Decision tree
DTBUILD (*D)
T = Null;
T = Create root node and label with splitting attribute;
T = Add arc to root node for each split predicate and label;
For each arc do
D = Database created by applying splitting predicate to D;
If stopping point reached for this path, then
T= Create leaf node and label with appropriate class;
Else
T = DTBUILD (D);
T = Add T to arc;
While building tree J48 ignores the missing value. J48 allows classification via either decision tree or rules generated from them [10] [11].

5.3 AdaboostM1

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gdel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner. Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier.[20][21] When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the

tree growing algorithm such that later trees tend to focus on harder-to-classify examples. AdaBoost refers to a particular method of training a boosted classifier. A boost classifier is a classifier in the form.

$$F_T(x) = \sum_{T=1}^{T} ft(x)$$

IV. Experimental Results

1. Comparison of Classifiers

We have made balanced dataset from which 1240 URLs are Benign and 1240 are Malicious. according to their class name, we classify them. It was found that all three classifiers have better results on dataset. After calculating Accuracy and Error for all three classifiers, It was found that, Naive Bayes have highest Accuracy and lowest Error compare to J48 and AdaboostM1 with 83.37% and 16.62% respectively. The following table shows the calculation and comparison of classifiers.

Table 4.1: The Table Shows Accuracy and Error of Classifiers

No.	Classifier	Accuracy	Error
1	Naive Bayes	83.37%	16.62%
2	J48	83.15%	16.84%
3	AdaboostM1	83.11%	16.88%

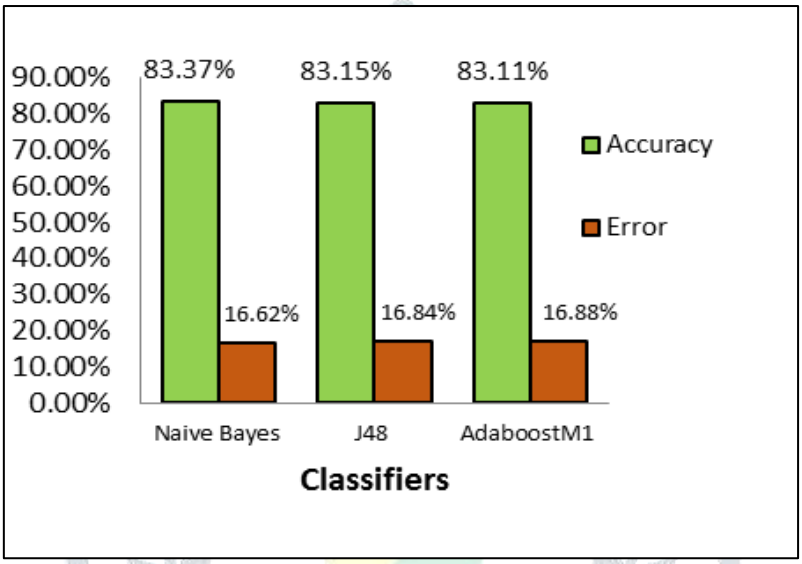

Figure 4.1: Comparison of Classifiers

2. Performance of Virus Total

VirusTotal not only tells you whether a given antivirus solution detected a submitted file as malicious, but also displays each engine's detection label (e.g., IWorm.Allaple.gen). The same is true for URL scanners, most of which will discriminate between malware sites, phishing sites, suspicious sites, etc. Some engines will provide additional information, stating explicitly whether a given URL belongs to a particular botnet, which brand is targeted by a given phishing site, and so on. We scanned more than 2200 of URLs and get 45.05% of accuracy only.

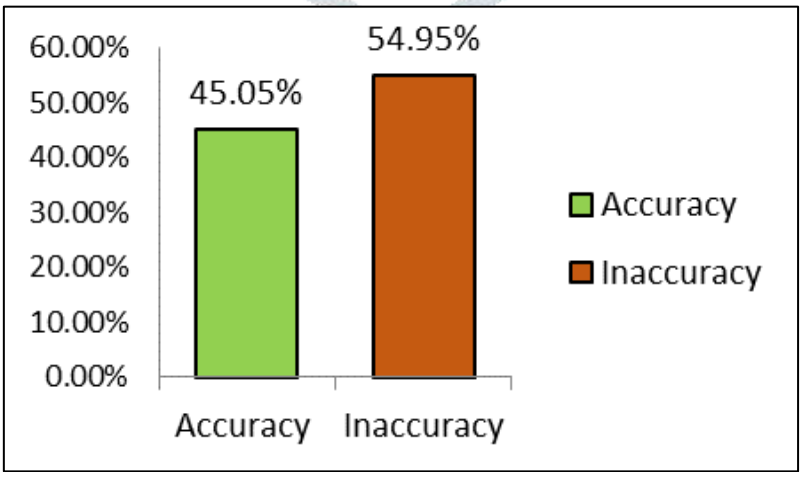

Figure 4.2: Online evaluated result of Virus Total

3. Comparison between Classifiers and Virus Total

We developed both techniques, Machine Learning Classification as well as Virus Total URL scanner, using Weka and Virus Total API to find Malicious Short URLs. In our analysis we found that, the Machine Learning Classification has best option to find malicious URLs. In our comparative analysis Naive Bayes classifier has best among all algorithms and Virus Total. Below table shows the comparative analysis of Classifiers and Virus Total.

Table 4.2: Comparison between Classifiers and Virus Total

	Classifiers			Anti Virus Engine
	Naïve Bayes	J48	AdavoostM1	Virus Total
Accuracy	83.37%	83.15%	83.11%	45.05%

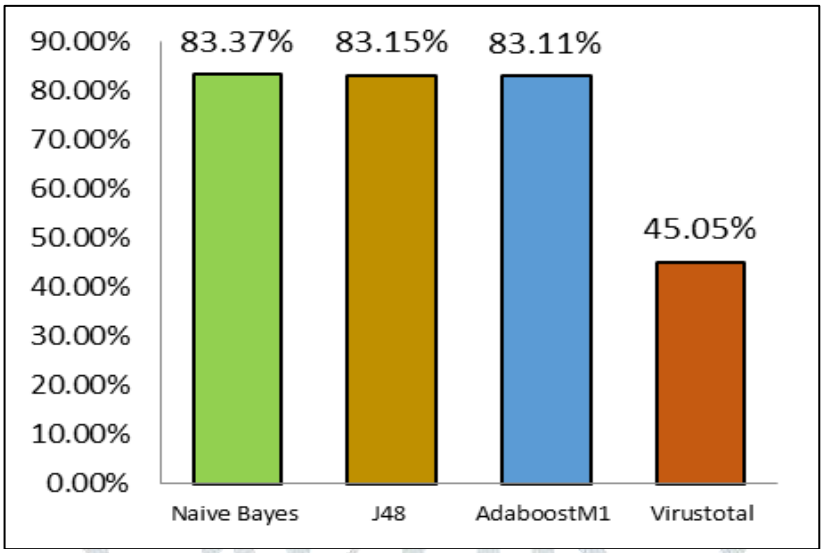

Figure 4.3: Comparative analysis of Classifiers and Virus Total

V. CONCLUSION

We have successfully extracted the features of short URLs from tweets. These features where very useful for classification using Naive Bayes, J48 and AdaboostM1. We observed the results of Naive Bayes classifier is best among J48, AdaboostM1 and compare to Virus Total. The work covered in this paper is up to the software handling and need to find malicious URLs manually. Many short URLs redirect the URLs to malicious content. To overcome this hazardous situation, we can develop a system which will connected to network, by which the incoming short URL will be automatically scanned.

REFERENCES

[1] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and Evaluation of a Real-Time URL Spam Filtering Service. Security and Privacy (SP) IEEE Symposium (2011), Pages 447 - 462.
[2] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting Spammers on Twitter. In Collaboration, Electronic messaging, AntiAbuse and Spam Conference (CEAS) (2010)
[3] A. Aggarwal, A. Rajadesingan, and P. Kumaraguru. PhishAri: Automatic Realtime Phishing Detection on Twitter. In Seventh IEEE APWG eCrime researchers summit (eCRS) (2012). Master's thesis, IIIT-Delhi (2012).
[4] C. Grier, K. Thomas, V. Paxson, and M. Zhang. @spam: The Underground on 140 Characters or Less. In proceedings of the 17th ACM conference on Computer and communications security (2010), Pages 27- 37.
[5] J. Ma, L.K. Saul, S. Savage, G.M. Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (2009), ACM, Pages 1245-1254.
[6] S. Lee and J. Kim. WARNINGBIRD: Detecting Suspicious URLs in Twitter Stream. NDSS 2012 (2012).
[7] D. Wang, S. B. Navathe, L. Liu, D. Irani, A. Tamersoy, and C. Pu. Click Traffic Analysis of Short URL Spam on Twitter. Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference (2013), Pages 250-259
[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. ACM SIGKDD Explorations Newsletter (2009), vol. 11, no. 1, Pages 1018.
[9] M. Kamber, L. Winstone, W. Gong, S. Cheng, and J.Han. Generalization and decision tree induction: Efficient classification in data mining. In Proc. 1997Int. Workshop Research Issues on Data Engineering (RIDE'97), pages 111-120, Birmingham, England, April 1997.
[10] http://www.jstor.org/discover/10.2307/40398417?uid=3738256uid=2134uid=36
[11] http://stackoverflow.com/questions/10317885/decision-tree-vs-naive-bayesclassifier
[12] Doyen Sahoo, Chenghao Liu, and Steven C.H. Hoi."Malicious URL Detection using Machine Learning: A Survey" pp.243 254.
[13] Gupta, N., Aggarwal, A., and Kumaraguru, P. 2014. bit.ly/malicious: Deep dive into short URL based ecrime detection, in eCrime, pp. 1424.