Leveraging Content, Context, and Social Attributes to Detect Malicious Short URLs in Online Social Network

Raj Kumar Nepali
Dakota State University

Follow this and additional works at: https://scholar.dsu.edu/theses

Recommended Citation
Nepali, Raj Kumar, "Leveraging Content, Context, and Social Attributes to Detect Malicious Short URLs in Online Social Network" (2016). Masters Theses & Doctoral Dissertations. 300.
https://scholar.dsu.edu/theses/300

This Dissertation is brought to you for free and open access by Beadle Scholar. It has been accepted for inclusion in Masters Theses & Doctoral Dissertations by an authorized administrator of Beadle Scholar. For more information, please contact repository@dsu.edu.
LEVERAGING CONTENT, CONTEXT, AND SOCIAL ATTRIBUTES TO DETECT MALICIOUS SHORT URLs IN ONLINE SOCIAL NETWORKS

A Dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Science

In

Information Systems

By:

Raj Kumar Nepali

Committee in charge:

Dr. Yong Wang, Chair
Dr. Jun Liu
Dr. Mark Hawkes
Dr. Surendra Sarnikar

2016
DISCUSSION APPROVAL FORM

This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Science in Information Systems degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department or university.

Student Name: Raj Kumar Nepali

Dissertation Title: Leveraging Content, Context, and Social Attributes to Detect Malicious Short URLs in Online Social Networks

Dissertation Chair: ___________________________ Date: 11-16-2016

Committee member: ___________________________ Date: 11-16-2016

Committee member: ___________________________ Date: 11-22-16

Committee member: ___________________________ Date: 11-22-16
DECLARATION

I declare hereby that this dissertation is a product of my own research and appropriate credit is given for the ideas and expressions of others. I declare that the dissertation contains as its main content work, which has not previously been submitted for a degree at any tertiary education institution.

…………………………………”

Raj Kumar Nepali
ACKNOWLEDGEMENTS

This dissertation is dedicated to all those people who have played important parts throughout this journey. I couldn’t have completed this without their precious support.

No one deserves more credit than my wife, Shristi, the love of my life. She encouraged me to pursue my dream and supported me tirelessly in every step throughout this endeavor. I am grateful to have her in my life.

To my parents, Jaya Bishnu and Nanda Nepali, who brought me into this world and taught me the meaning of life and the value of hard work. I am forever in debt to them. Their never ending support, countless sacrifices, and trust helped me become everything that I am today.

I am forever indebted to my advisor Dr. Yong Wang. His patience, support, motivation, and persistent encouragement are invaluable to the completion of this dissertation. I could not have imagined having a better advisor and mentor for my doctoral study.

I am thankful to the rest of my dissertation committee, Dr. Mark Hawkes, Dr. Jun Liu, and Dr. Surendra Sarnikar, for their encouragement and insightful comments, which pushed me to broaden my knowledge and widen my research.

I would also like to acknowledge DSU family, Madison Community Center, DSU Futbol, and my colleagues for their support and encouragements. Thank you all.
ABSTRACT

Online social networks (OSNs) are platforms to connect and communicate with friends, families, and like-minded people. Users post thoughts, comment to other’s posts, share photos and videos, and share information. The shared information often includes URLs (Uniform Resource Locators), which direct users to web content like news, articles, and advertisements.

URL sharing is very popular on online social networks. However, URL sharing is not always convenient because of overly long and complicated URL strings. Thus, short URLs have become very popular on OSNs because of their simplicity. However, many risks have been found and reported in association with sharing short URLs. Malicious users utilize short URLs heavily in their sinister campaigns such as phishing, malware, spams, and scams. It is highly desirable to design and develop an effective short URL classifier to mitigate these threats on online social networks.

In this dissertation, we develop a short URL classifier, CONSOL, using the features collected from online social networks. We achieve an accuracy of 94.5% in identifying malicious short URLs using Random Forest machine learning algorithm. Unlike most existing techniques which depend on third party resources to classify URLs, our classifier does not depend on any third party service providers during its operation and leverages features available on OSNs only. Our research identifies 16 features that are important for short URL classification. These 16 features are logically categorized into three categories, i.e., content features, context features, and social features. Further analysis reveals that social features contribute significantly towards classifying short URLs and context features are also good indicators of the malignity of short URLs.
Compared to social features and content features, context features are less important. However, context features complement the classifier to be more effective. The comparisons of the CONSOL with the existing solution and Google Safe Browsing show that the classifier is promising in the real world too. CONSOL is slightly better than the existing solution. However, unlike the existing solutions relying on third party information, CONSOL runs on its own. Our testing also indicates that CONSOL identifies malicious short URLs much faster than the Google Safe Browsing. The results are validated and supported by VirusTotal. Our case studies further demonstrate that other online social networks can also adopt CONSOL for short URL classification.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................................................................................ IV

ABSTRACT .......................................................................................................................... V

CHAPTER 1 INTRODUCTION .............................................................................................. 1

1.1 ONLINE SOCIAL NETWORKS ......................................................................................... 1
1.2 URLS AND SHORT URLS .............................................................................................. 2
1.3 SHORT URL SECURITY .................................................................................................. 2
1.4 LEVERAGING SOCIAL NETWORK FEATURES TO DETECT MALICIOUS URLS ....... 4
1.5 CONTRIBUTIONS ........................................................................................................... 5

CHAPTER 2 LITERATURE REVIEW .................................................................................... 7

2.1 PRIVACY ON OSNS ...................................................................................................... 7
2.2 URLS AND SHORT URLS .............................................................................................. 8
2.3 MALICIOUS URLS ....................................................................................................... 10
2.4 BLACKLIST .................................................................................................................. 11
2.5 MALICIOUS URL DETECTION ...................................................................................... 12
2.6 SUMMARY .................................................................................................................... 17

CHAPTER 3 RESEARCH METHODOLOGY ......................................................................... 19

3.1 DESIGN SCIENCE VS. BEHAVIORAL RESEARCH ...................................................... 19
3.2 DESIGN SCIENCE RESEARCH METHODOLOGY ..................................................... 20
3.3 MACHINE LEARNING .................................................................................................. 22
3.4 SUMMARY .................................................................................................................... 29

CHAPTER 4 CONSOL: AN ONLINE SOCIAL NETWORK BASED SHORT URL CLASSIFIER  30
APPENDICES ................................................................................................................. 79
APPENDIX I TWEET ...................................................................................................... 79
APPENDIX II SAMPLE DATASET .............................................................................. 83
REFERENCES ................................................................................................................. 84
List of Figures

Figure 2-1 Short URL Working Mechanism ................................................................. 10

Figure 3-1 Research Methodology (Adopted from Peffers et al., 2007) ...................... 21

Figure 4-1 Conceptual Model for CONSOL ................................................................ 32

Figure 4-2 Architectural Diagram .............................................................................. 33

Figure 4-3 Frequency Table for Bag-of-Words ............................................................ 41

Figure 4-4 RMSE Logistic Regression ........................................................................ 48

Figure 4-5 RMSE Naive Bayes .................................................................................. 48

Figure 4-6 RMSE Support Vector Machine ................................................................ 49

Figure 4-7 RMSE Random Forest ............................................................................. 49

Figure 4-8 Testing Results .......................................................................................... 51

Figure 4-9 Results of Wrapper Method Feature Selection ......................................... 52

Figure 5-1 Graphical Representation of Comparison against Wang et al. (WDB1) .... 57

Figure 5-2 Graphical Representation of Comparison against Wang et al. (RDB1) .... 57

Figure 5-3 Comparison against Safe Browsing ........................................................... 59
List of Tables

Table 2-1 Schemes as Described in RFC 1738 .......................................................... 9
Table 4-1 Features and Their Descriptions ...................................................................... 40
Table 4-2 Output of InfoGainAttributeEval Function of Weka ........................................ 43
Table 4-3 Evaluation Metrics .......................................................................................... 44
Table 4-4 Datasets and Description ................................................................................ 45
Table 4-5 Performance Metrics for Feature Selection ..................................................... 50
Table 4-6 Selected Features for Random Forest with Filter Method ............................... 51
Table 4-7 Wrapper Feature Selection Results .................................................................. 53
Table 5-1 Comparison against Wang et al. .................................................................... 56
Table 5-2 Accuracy of the Classifier with Different Feature Sets ..................................... 61
**List of Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DS</td>
<td>Dataset</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>OSN</td>
<td>Online Social Networks</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-Peer</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comments</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WOT</td>
<td>Web of Trust</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Chapter 1 Introduction

1.1 Online Social Networks

An online social network (OSN) is a digital platform where users can create their digital persona with real-life information such as full name, email address, home address, and phone numbers. This digital persona is used to connect to friends, families, colleagues, etc. People from different walk of life join OSNs and connect with friends and like-minded people forming groups. OSN such as Facebook have attracted millions of users worldwide. According to Facebook, there were 1.09 billion daily active users in 2016 (Facebook, 2016). There are hundreds of different types of OSNs today.

OSNs are places where people are willing to share information. The shared information is scattered in various data sources, such as user profiles and instant messages. The shared information is generally available to public and can be retrieved by users around the world. Because of the massive amount of information collected and stored on OSNs, it also raises many security and privacy issues (Nepali & Wang, 2013; Y. Wang & Nepali, 2015b).

Many threats and attacks have been reported on OSNs such as phishing, spam, scam, malware propagation, and social engineering. Phishing is the most popular attack among others. In a phishing attack, attackers often trick users to click on URLs or short URLs and redirect them to malicious websites to steal their sensitive information such as birthdate, social security number (SSN), and credit card number. Due to the massive amount of information available and large numbers of unsuspected and vulnerable users, OSNs are very attractive to attackers. Jagatic et al. (2007) found that users are more
susceptible to phishing attacks when social relationships are exploited. According to research by Kaspersky, 22 percent of phishing scams on the web targeted users on Facebook in 2014 (Stern, 2014).

1.2 URLs and Short URLs

URLs (Uniform Resource Locators) are pointers to the information on the Internet. URLs are used to share content on online social networks. However, URL sharing can be problematic because of its length when shared via posts, messages, and emails. Particularly, it becomes a huge issue on Twitter. Twitter is a very popular OSN and has millions of active users. It limits the number of characters (140 characters) for each tweet that users can post. This led to the popularity of URL shortening services. URL shortening services take a long URL from users and create a short URL, an alias, for the long URL. Short URLs can reduce long URLs significantly and make URL sharing easy. When a user clicks on a short URL, the user will be directed to the URL shortening service provider, which will then redirect the user to the original long URL or the “Landing page”. All this happens on the background without any user intervention. These services have gained popularity ever since they appeared in 2001. As of today, there are more than 500 URL shortening services on the Internet. Bitly is one of the most widely used URL shortening service providers.

1.3 Short URL Security

Short URLs are very handy and are easy to share. They save text space and are easy to manage. Because of these traits, the use of short URLs has skyrocketed in the past five years. Unfortunately, URL shortening services have been misused by attackers to spread malicious URLs (Chhabra, Aggarwal, Benevenuto, & Kumaraguru, 2011; Maggi
et al., 2013). Short URLs can hide original URL information and may not be related to them at all in naming. Because of this, users have no idea where the URL directs them to when they click on a short URL. Attackers leverage this knowledge to their advantage in nefarious ways, such as phishing scams, spamming, and malware campaigns. Social networks have been found especially vulnerable to these attacks (Castillo, Mendoza, & Poblete, 2011) due to large numbers of easy targets, social relationship exploitation, and high success rate. Jagatic et al. (2007) found that users are five times more susceptible to attacks when social contexts are used.

Despite the popularity of short URLs, URL shortening service providers fail to prevent malicious URLs from being converted. In an experiment by Maggi et al. in 2013, the authors attempted to convert malicious URLs collected from Wepawet, PhishTank, and Spamhaus against 6 URL shortening services. They found that most of the malicious URLs were accepted for conversion (Maggi et al., 2013). Online social networking sites also fail to prevent such malicious short URLs from being posted. There are a couple of reasons behind this. First, both services often use blacklists to prevent malicious URLs from being posted. However, blacklists are not always comprehensive and up-to-date. Second, because a short URL may look completely differently than its full URL, many current mechanisms cannot provide good results in classifying short URLs. For example, multiple shortening of a URL will create a completely new short URL, which does not exist in any blacklists. Heuristics-based malicious URL classifiers are based on URL specific features, e.g., length, number of dots in URL, etc. These features are not presented in the short URLs after conversion. This makes conventional filters useless.
1.4 Leveraging Social Network Features to Detect Malicious URLs

In this research, we investigate how to leverage social network features to classify malicious short URLs. Previous research exists and utilizes limited features from online social networks to detect malicious short URLs. However, most of the works rely heavily on the information derived from third party URL shortening service providers. It is not always possible to obtain data from third parties provided that there are hundreds of URL shortening services available to date and many of them are proprietary. In addition, the results from previous research on short URL classification are also not satisfactory enough.

Online social networks contain huge amount of information about users, short URLs, and its propagation. To the best of our knowledge, the information has never been fully studied and utilized for malicious short URL detection. We believe that malicious short URLs can be detected using the information from OSNs only because malicious short URL propagates in online social networks following social phenomenon among friends and followers. The benefits of the approach can be twofold. First, it does not rely on the information from third parties. Second, it leverages the vast amount of information available on OSNs.

The features we investigate on OSNs include content features, context features, and social features. These features are available on OSNs and are easy to collect. Our approach utilizes these features to detect malicious short URLs. Our assertion is that detecting malicious short URLs is practical using information from OSNs only since malicious short URLs propagate through OSNs leveraging social relationships.

Our research questions in this dissertation include:
a) How can we leverage the information available on OSNs to detect malicious short URLs?

b) What are the most important features for malicious short URL detection?

c) How can we develop an effective mechanism to detect malicious short URLs with high accuracy on OSNs?

d) Can we adopt the model and use the model for other online social networks with simple or no changes?

1.5 Contributions

To the best of our knowledge, few works have been conducted in the literature to study the security of short URLs on online social networks. The contributions of the research include, but are not limited to,

- We limited the scope of contemporary short URL classification study to online social networks.

- We design and develop a new artifact, CONSOL, for short URL classification. We demonstrate that the information from OSNs can sufficiently and effectively classify short URLs with high accuracy. Our evaluation and analysis show that the classifier achieves comparatively better performance than the existing solutions.

- We evaluate the impact of different categories of features in classifying short URLs and find that social features and content features contribute the most in classification of short URLs.
• The findings of this dissertation will greatly help short URL classification in online social networks because the features used in the classifier is freely available on any online social networks.

The rest of the dissertation is organized as follows: Chapter 2 presents a literature review of the existing research and techniques used for short URL classification. Chapter 3 introduces the research methodology adopted in the dissertation. Chapter 4 introduces the design and development of the short URL classifier, CONSOL. Chapter 5 further evaluates and analyzes the performance of the short URL classifier. Chapter 6 demonstrates that the classifier can also be adopted in other online social networks. Chapter 7 finally concludes the dissertation and summarizes our future work.
Chapter 2 Literature Review

Online social networks have been found especially vulnerable to privacy and security attacks due to large number of easy targets, social relationship exploitation, and high success rate (Castillo et al., 2011). This research focuses on detecting malicious short URLs on online social networks. This chapter briefly introduces security and privacy issues on OSNs, how URLs and short URLs are used for malicious purpose, a conventional way of using blacklists to prevent malicious URL propagation, followed by a review of existing approaches for malicious short URL detection.

2.1 Privacy on OSNs

Personal information collection and privacy breach are major concerns on OSNs. Privacy threats range all the way from online stalking, sensitive information leakage, identity theft (Bilge, Strufe, Balzarotti, Kirda, & Antipolis, 2009), to sexual offense and physical security. Privacy and security threats are spreading through OSNs (Bilge et al., 2009; Narayanan & Shmatikov, 2009; Wondracek, Holz, Kirda, & Kruegel, 2010). However, because of the nature of the threats, attitude towards privacy, and insufficient knowledge, many people do not report privacy breach. Hence, it is difficult to quantitatively state the impact of privacy breach.

Many studies have been conducted to address security and privacy issues on OSNs. A threat modeling framework is presented in (Y. Wang & Nepali, 2015b). It provides a comprehensive overview of security threats and countermeasures on OSNs. To measure an individual’s privacy exposure on OSNs, privacy index is proposed (Nepali & Wang, 2013). Privacy index is a numerical measurement of a user’s privacy exposure on OSNs (Nepali & Wang, 2013). It considers how much information is disclosed on OSNs,
the sensitivity of the disclosed information, and the visibility of that information. Using privacy index, privacy impact can also be assessed in case a data breach occurs (Y. Wang & Nepali, 2013, 2015a). In addition to the anonymization techniques used to protect user privacy such as K-anonymity (Sweeney, 2002) and L-diversity (Machanavajjhala, Gehrke, Kifler, & Venkitasubramaniam, 2006), privacy index can also be used to monitor privacy exposure on the Internet and alert users when their privacy exposure changes dramatically. Other techniques used to protect user privacy on OSNs include, but are not limited to, P2P architecture (Buchegger & Schi, 2009), and SocBridge (Nepali & Wang, 2014).

2.2 URLs and Short URLs

URL is an abbreviation for Uniform Resource Locator. Defined in RFC 1738 by Tim Berners-Lee (1994), it is a reference of a web resource located on a computer network. “URLs are used to ‘locate’ resources, by providing an abstract identification of the resource location”. RFC 1738 describes the syntax of the URLs. URLs are written as follows:

<scheme>[:<scheme-specific-part>]

Schemes can be one of the followings as shown in Table 2-1 according to RFC 1738.

Most of URLs are web addresses, and are represented with http and https schemes only, such as,

http://www.facebook.com/12345678/photos/98765432111.jpg

https://www.paypal.com/login.php
Table 2-1 Schemes as Described in RFC 1738

<table>
<thead>
<tr>
<th>SCHEME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>http</td>
<td>Hypertext transfer protocol</td>
</tr>
<tr>
<td>gopher</td>
<td>The Gopher protocol</td>
</tr>
<tr>
<td>mailto</td>
<td>Electronic mail address</td>
</tr>
<tr>
<td>news</td>
<td>USENET news</td>
</tr>
<tr>
<td>nntp</td>
<td>USENET news using NNTP access</td>
</tr>
<tr>
<td>telnet</td>
<td>Reference to interactive sessions</td>
</tr>
<tr>
<td>wais</td>
<td>Wide Area Information Severs</td>
</tr>
<tr>
<td>file</td>
<td>Host specific file names</td>
</tr>
<tr>
<td>prospero</td>
<td>Prospero directory service</td>
</tr>
</tbody>
</table>

For the context of this research, we are particularly interested in http and https schemes.

Short URLs appeared around early 2000s to make URL sharing simple and easy. Sharing a long URL is complicated and consumes significant text space. It becomes a huge issue when a limitation of maximum text space is applied on users. For example, Twitter limits the text messages to 140 characters. This led to the significant increase in the adoption of short URLs. The use of short URLs was slow in the beginning but became popular in mid-2000s. Today, there are more than 500 URL shortening service providers, public or proprietary, on the Internet.

A short URL is an alias of a URL that is shorter in length, as its name suggests, compared to the original URL. URL shortening services take a long URL from users and create a corresponding short URL. A short URL is created by appending hash of a URL submitted by the user to the URL shorting service provider’s domain name. For example, using Bitly, the long URL, www.dsu.edu/research/informationassurance/shorturls.html, is hashed to generate a unique string ‘ab7cde8’. The hashed string is then appended to the service provider’s domain name to generate a short URL, http://bit.ly/ab7de8. Service providers also provide options to generate user desired domain strings, known as branded
short name domains, to replace the providers’ domain names. Users are able to create a
custom unique string, known as custom alias, and use it to replace the hash value. For
example, using custom string xyz.ly/ as a branded short name domain, all the short URLs
the user creates are automatically generated in the form of http://xyz.ly/abcdef.

Research found that short URLs can reduce the length of the URLs up to 91%
with 54% overhead in latency (Antoniades et al., 2011). When users click on a short
URL, they are directed to the service provider. The service provider will perform a look
up against its database with the short URL and return the corresponding long URL to the
browser. The browser will then redirect the user to the original long URL “Landing
Page”. All of these processes happen automatically without user intervention as shown in
Figure 2-1:

![Figure 2-1 Short URL Working Mechanism](image)

**2.3 Malicious URLs**

A URL is an address of content, information, or a document on the World Wide
Web (Berners-Lee et al., 1994). The address stores the document or the information of
interest. While some web content is genuine, some content might be infected with
unwanted programs like viruses, spyware, malware, etc. Unfortunately, URLs can be
used to direct users to malicious content. A URL which points to an infected file location or a fake web service location with the intent of causing harm to the user’s computer system is a malicious URL. Some malicious URL mimic genuine web application and lure users to disclose personal sensitive information, which otherwise they would not disclose. This is known as phishing attacks (PhishTank, 2016c).

Malicious URLs studied in this research include, but are not limited to, the phishing URLs that are intended to steal personal login credentials, the URLs that can spread malware by exploiting browser related vulnerabilities, the spam URLs that lure users to undesired content, and the scam URLs that lure users to activities that they otherwise would not do.

2.4 Blacklist

Blacklist is used to register known entities that are malicious in nature. Blacklists are used for different purposes. Examples of blacklists include DNS blacklist, software blacklist, etc. URL blacklist contains URLs that are known to be malicious. These URLs can be part of a phishing campaign, a scam campaign, or a malware campaign. Many techniques are used to populate backlists. One of the most popular technique is based on user voting (PhishTank, 2016a). This is particularly famous for phishing URLs. Users can flag any suspected URLs. The experts in the community will then manually verify a suspected address and add the verified phishing address to a blacklist. Automated analyzing techniques can be used to identify the URLs, which contain malicious content too. These techniques include, but are not limited to, signature analysis for malicious payload, sandboxing (Malwareblacklist, 2016), and machine learning (Developers, 2015). Signature analysis for malicious URL detection is an active detection mechanism where
signatures for malicious purposes, e.g., malicious code and credential harvesting, are analyzed within the content of the URL. Sandboxing includes opening the suspected URL in a sandboxed environment and analysis for any registry changes, zero-day exploits, and credential harvesting. Machine learning uses different features (e.g., lexical, host based, and other) to identify if a URL is malicious or not.

In this research, we use two very popular blacklists from VirusTotal and PhishTank to label URLs. PhishTank is a very popular phishing blacklist service operated by OpenDNS (PhishTank, 2016b). It provides APIs for developers to perform lookups for malignity of a URL or a batch of URLs. The results can be obtained in a JSON or XML format as requested by the user. VirusTotal is another popular web-based blacklist. It generates a report for malicious links scanned by 56 website scanning services. VirusTotal also provides APIs for developers to integrate the labeling service into their systems. The report is returned in JSON or XML format per request (Virustotal, 2015). The results can then be utilized as desired by the applications.

### 2.5 Malicious URL Detection

Malicious URL detection in general can be categorized into two approaches, active detection mechanism and passive detection mechanism. Active detection mechanism classifies a URL based on the content of the URL. An automated process retrieves and analyzes the content of the URL. Different approaches can be utilized to analyze the content, for example, file analysis, sandboxing, etc. However, attackers can leverage evasion techniques like selective forwarding where scripts can manipulate the automated process to obtain genuine content, and redirect real users to malicious content or temporal behaviors (K., C., J., V., & D., 2011; Rajab et al., 2011). Passive detection
mechanism classifies URLs based on the features from different sources without actually visiting the URL. Literature suggests the use of host based and lexical features (Ma, Saul, Savage, & Voelker, 2009a; McGrath & Gupta, 2008) are very popular. Passive detection mechanism is suitable when visiting the actual URL possesses risks or obtaining the actual content might not be possible. In this research, we use passive detection mechanism.

Most of the malicious URL detection approaches are based on machine learning. The remaining chapter reviews the machine learning approaches for malware detections, particularly the approaches to classify malicious URLs in general and the approaches to classify short URLs. Note that most of the techniques for malicious URL detection do not apply to short URLs. The chapter also reviews other approaches for malicious URL detection.

2.5.1 Machine Learning Approaches

Machine learning is a technique where computers make intelligent decision based upon the data provided. It has been used in many fields of information security. Machine learning is believed as the most important technique in classifying URLs in both academia and industry.

2.5.1.1 Malicious URL Detection

Malicious URL detection typically leverages lexical features and host-based features. Lexical features are the textual characteristics of the URL, for example, presence of '.', '/', '?', '=', '-', etc. The attempt is to find the properties of malicious URLs which look differently from the genuine URLs. For example, a malicious URL, www.ebay.com.payment.net, is different from a typical URL, www.ebay.com. McGrath
and Gupta (McGrath & Gupta, 2008) and Kolari et al. (2006) suggested a set of lexical features to be studied while classifying URLs. McGrath and Gupta (2008) found that phishing URLs look differently than the brands they target. However, they contain 50-75% keywords that are similar. Phishing URLs tend to be longer in length. Phishing domains tend to be shorter in length, use fewer vowels, and have fewer unique characters. Kolari et al. (2006) studied presence of words, anchors, metadata, 4grams and used machine learning approaches to classify spam blogs. Blum et al. (2010) leveraged lexical features for online learning of phishing URL detection. In addition to page content, they used 8 different previous studied URL features and introduced 3 new features in classification. They also used social reputation features from social media. Le et al. (2011) used lexical features of URLs to classify phishing URLs. They showed that lexical features alone can effectively classify phishing URLs and achieved accuracy up to 97%.

Whittaker et al. (2010) developed a classifier to maintain Google’s phishing blacklist using lexical features along with the content features and the reputation-based features obtained from Google. The classifier achieved 90% accuracy and maintained false positive below 0.1%.

Host based features include, but are not limited to, WHOIS record, geographic information, domain name properties, and IP address properties. The idea behind the approach is that malicious sites are hosted with less reputable services (Ma et al., 2009a). Studies have found that many malicious website are hosted on compromised client machines and malicious URLs often include IP addresses, domain tasting (Kolari et al., 2006), etc. The classifier from Whittaker et al. (Whittaker et al., 2010) also utilized host based features in addition to the content features, the lexical features, and the reputation-
based features. Ma et al. (Ma et al., 2009a) developed a classifier to detect malicious URLs and achieved 95-99% accuracy. Their studies suggested the power of host-based features in malicious URL detection. Ma et al. (2009b) further extended their studies to perform a large scale online learning of suspicious URLs and used lexical and host-based features to classify malicious URLs. It achieved an accuracy of 99% on a balanced dataset. Similar works have been conducted in other areas such as phishing URL detection.

2.5.1.2 Malicious Short URL Detection

Malicious short URL detection in social networks has gained many attentions recently. Although studies suggested the use of short URLs by malicious users happened long time ago (McGrath & Gupta, 2008), short URL research did not attract many interests. The first large scale exploratory research on short URLs was done by Antonaides et al. (Antoniades et al., 2011). They found that short URLs are mostly popular on OSNs. The short URLs are used to point to news and informative content and have word of mouth propagation on OSNs.

First ever study on the security implications of short URLs was conducted by Maggi et al. (Maggi et al., 2013) over the course of two years. They found that short URL service providers fail to prevent malicious URLs from conversion. They also found that Bitly accepted a malicious URL for conversion even if it flagged the URL as malicious. All of the services under study check the URLs when submitted. However, attackers have full privileges of the website they control. They can inject the malicious content into their website after the security check and convert a valid short URL address to a malicious short URL. This is a real problem with URL shortening services. Maggi et al. (2013) also
discovered that 95% of the malicious short URLs had life span of 4 days. Almost every URL shortening service security mechanism can be bypassed. Wang et al. (2013) studied the misuse of short URLs. They leveraged characteristics of spam URLs and developed a mechanism to detect them. The authors used click-traffic features from Bitly to develop a classifier and achieved accuracy up to 90.81% using Random Forest algorithm. Data labeling was performed using known suspended accounts, Google Safe Browsing, McAfee Site Advisor, URIBL, SURBL, and Spamhaus. Lee and Kim (2013) developed a mechanism to detect suspicious URLs in real time from twitter stream. Attackers also implement techniques to evade malicious URL detection, e.g., redirecting automated tools used by researchers to benign pages. Lee and Kim (2013) studied the redirection cases and developed a mechanism to leverage attacker’s URL redirection chain and tweet context. They developed a classifier with nearly 92% accuracy. Their approach was experimentally proven to be more effective than the mechanism used by Twitter. Gupta, Aggarwal, & Kumaraguru (2014) conducted exploratory study on Bitly’s spam URL/account detection mechanism. They found that suspicious Bitly account could undertake prolonged malicious activities and remain undetected. They used short URL features and two domain specific features to classify Bitly short URLs and achieved an accuracy of 86%. Klien and Strohmaier (2012) discovered that 80% of the shortened URLs contained spam content and the problem persisted on the global scale. They suggested the need of the research for techniques to discover malicious short URLs. Our research on malicious short URL classification can be found in (Nepali, Wang, & Alshboul, 2015).
2.5.2 Other Approaches to Detect Malicious URLs

Several other techniques have been proposed and used in the literature to classify URLs. The most popular approaches used are voting and crowdsourcing. Voting is the process where users provide their opinions on the malignity of the URLs. Usually, the process starts by some users/volunteers posting a suspected URL for voting in the community. The experts in the community then vote on the malignity of the URL. If enough votes are cast on the malignity of the URL, it is considered malicious. The malicious URLs are registered in the blacklist and distributed for public use. One of the most popular service that follows this approach is PhishTank (2016b).

Crowdsourcing follows the similar approach of voting except millions of users provide the review rather than a small group of experts in the community (Wikipedia, 2015). The users for crowdsourcing can be normal users as well as experts. One of the most popular services is Web of Trust (WOT, 2016). Web of Trust is available as an add-on module on many browsers. Millions of users provide safety ratings on different websites based on their opinions of a web page. Users can check the ratings in the form of different signals, e.g., green for safe and red for malicious, for the trustworthiness of a website.

The problem with these techniques is that it takes long time for the malicious URLs to appear on a blacklist. It also suffers from human errors due to subjective judgment on a web page.

2.6 Summary

Malicious short URLs detection is challenging. Many approaches used for URL classification do not work well for short URLs. Current mechanisms for short URL
classification depend on third party information, which limits its availability. Our literature review indicates more research needs to be conducted on malicious short URL classification. An effective short URL classifier is highly desirable on online social networks.
Chapter 3 Research Methodology

There are two dominant research approaches in information systems, i.e., design science approach and behavioral approach (Hevner, March, Park, & Ram, 2004). Our research follows design science approach. This chapter compares these two approaches and also summarizes how we apply design science approach to our research.

3.1 Design Science vs. Behavioral Research

Information systems research deals with artificial phenomenon which can be created and studied (March & Smith, 1995). According to Simon (1986), natural science is about understanding of how and why things work while design science is about building an artifact that can be utilized to achieve goals. Design deals with the invention of a new artifact. Design science research differs from routine design because the knowledge required to develop the artifact does not exist (Vaishnavi & Kuechler, 2004). Design science research results in an artifact that solves problems in information systems. Such artifacts can be constructs, models, methods, and instantiations (March & Smith, 1995).

Behavioral research in information systems has its root on natural science and it tends to develop and justify theories. Theories explain the natural phenomenon of interactions between information systems and its environment. Theories help develop valuable artifacts that meet their purposes in the design process. Without complete understanding of its environment and the interactions of an artifact with its environment, an artifact may fail and produce undesired outcomes (March & Smith, 1995). Design science is technology-oriented and aims to develop things that serve human purposes. In the process, design science consumes the theories developed by natural science to build
an artifact that meets its goals. Artifacts developed are used in information systems to interact with environments, and hence are bound by natural theories. Design science and behavioral research have different focus. Both paradigms are required and they complement each another.

In this research, we follow design science principle. Our objective is to develop an innovative artifact to solve a critical issue in information systems.

3.2 Design Science Research Methodology

Design science research in information systems has its roots in engineering (Simon, 1986). The main goal of design science research is to create innovative ideas and practices which result in efficient and effective design, management, and use of information systems (Denning, 1997). The output of design science, artifacts, can be in the form of software, formal logic, mathematical function, or informal natural language descriptions (Hevner et al., 2004). The artifacts of design science can be divided into four types, i.e., constructs, models, methods, and instantiation (March & Smith, 1995). Construct is a natural language, which describes a concept and characterizes a phenomenon. Models describe tasks at a higher level. Methods describe set of activities, which should be performed to meet the goal. Instantiation produces products such as software and processes. Design science goes through a building and evaluating cycle (March & Smith, 1995). During the building phase, an artifact is built to perform certain objective. The main goal in this phase is to prove that such an artifact can be built. Evaluation deals with developing the criteria for performance assessment.
We follow Peffers et al. (2007)’s design science research approach for this study.

Figure 3-1 shows our approach for this research.

**Figure 3-1 Research Methodology (Adopted from Peffers et al., 2007)**

Peffers et al. suggested six stages in design science research, i.e., problem identification and motivation, definition of the objectives for the solution, design and development, demonstration, evaluation, and communication. We adopt Peffers et al.'s research methodology in the design of malicious short URL classifier.

I. *Problem Identification and Motivation*: During this stage, specific research problem is identified and the importance to solve the problem is justified. Requirements for such a system can also be identified whenever possible. Chapter 1 and Chapter 2 describe the problem and the motivation for this research.

II. *Definition for the Objective*: Quantitative or qualitative objective is derived from the problem definition in this stage. The objective of this
research is to develop an effective short URL classifier using the features from OSNs as shown in Chapter 1.

III. Design and Development: Artifact is designed and developed in the form of construct, model, method, or instantiation in this stage. The artifact should meet its functionality. The design and development of the malicious short URL classifier are presented in Chapter 4.

IV. Demonstration: In this stage, the ability of the artifact to solve intended research problem is demonstrated. This can be achieved in terms of experiment, simulation, case study, etc. Chapter 4 includes experiments demonstrating the ability of the short URL classifier.

V. Evaluation: The artifact’s performance is measured in terms of pre-established criteria and metrics. Chapter 5 and Chapter 6 perform quantitative analysis of the performance of the artifact and measure how well the objectives of the research are met.

VI. Communication: The outcomes of this research, its relevance, utility, rigor in the development of the artifact, and effectiveness are presented in major conferences like AMCIS and HICSS.

3.3 Machine Learning

Machine learning is the study of computer algorithms that improve automatically through experience. “A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, its performance at tasks in $T$, as measured by $P$, improves with experience $E$” (Mitchell, 1997).
Machine learning depends on the objective of the task and is broadly classified into three categories, i.e., supervised learning, unsupervised Learning, and reinforcement learning. Supervised machine learning requires labeled data and is concerned with identifying the relationship between the data and the label. Label refers to the possible outputs, e.g., malicious or benign, good or bad, etc. Unsupervised learning does not require labeled data and deals with the patterns inherent in the data to find the hidden relationships, e.g., social network analysis. Reinforcement learning operates in a dynamic environment and learns to perform certain tasks, e.g., automated driving, playing poker, playing chess, etc.

We studied machine learning algorithms available and heavily used in the community and identified the most popular algorithms in the similar research to be tested for our classifier. Datasets include labeled datasets with training and testing datasets. Labeled datasets contain fields such as short URLs, features, and tags that represent malignity of short URLs.

We selected Twitter as a case because Twitter is one of the most widely used OSNs. URLs are collected along with user information from tweets using Twitter’s streaming APIs (Twitter, 2015). For simplicity, we focus on Bitly short URLs since Bitly is the most widely used URL shortening service to date. Tweets with Bitly URLs are collected. Content, context, and social information are extracted from tweets. Tweet related information such as clicks, favorites, and retweets are also collected. The relevance of a tweet will be determined using tweet trends. Twitter provides top 10 popular trends at any given point in time. It is believed that attackers also leverage relevant context information to spread malicious URLs. As we collect tweets, global
tweet trends are also collected per minute for the classifier. To simplify the study, non-English tweets are removed.

Extracted short URLs are expanded to long URLs using Bitly APIs. The long URLs are then submitted to popular web URL scanning services such as VirusTotal and PhishTank to obtain the labeled dataset. VirusTotal generates a report for malicious URLs from 67 website scanning services. Some of the popular scanning services VirusTotal uses include AlienVault, Avira, BitDefender, ESET, Google Safe Browsing, Kaspersky, PhishTank, Sucuri and Sophos, etc. The report is retuned in JSON format (VirusTotal, 2015). The landing page of the deferred redirection URL will be determined and submitted to VirusTotal for analysis. PhishTank is another blacklist service operated by OpenDNS (PhishTank, 2016b). It provides developers and researchers open APIs to download blacklists and look up if a particular URL is a malicious URL. The results are returned in JSON or XML format. The results are analyzed to detect malicious nature of the URL. A labeled dataset is created from the results.

After collecting all the desired information, the features that are unique for malicious short URLs are investigated. We leverage the knowledge from the literature to derive significant signatures for malicious URL detection. Features are then evaluated to find out the most relevant feature set to be used in our classifier.

Using the labeled dataset and the feature set determined, a classifier is developed using different machine learning algorithms. The best performing algorithm is then identified and carried on for further evaluation. Note that our classifier leverages third party information from Bitly, VirusTotal, and PhishTank in the training phase. However,
the classifier operates independently to detect malicious short URLs on online social networks without relying on any information from third parties including blacklists.

3.3.1 Supervised Machine Learning

In this research, we use supervised machine learning. Supervised machine learning leverages a labeled dataset, which contains independent variables and corresponding dependent variable. Independent variables are the features used in the learning process and dependent variables are the different classes that the instances belong to. The objective is to infer a function that maps out the relationship between the independent and dependent variables and utilize the function to make similar predictions. In this research, the dataset contains the features extracted from the tweet data and the corresponding label from the blacklist lookup.

The features are further classified into three categories, i.e., content features, context features, and social features. Content features are derived from tweet text only, e.g., length of tweet, bag of words, and number of URLs. Bag of words are created from tweet contents including malicious URLs. This is based on frequency analysis of the words in tweets. Context features are derived from the contextual information in a tweet, e.g., relevance, and user mentions. Social features are derived from the social information in a tweet and the user profile, e.g., friends and followers.

After the features are extracted, different machine learning algorithms are used to develop a classifier. 10-fold cross validation is used to minimize the bias in training. We test different machine learning algorithms including Random Forest algorithm, Support Vector Machine (SVM), Naïve Bayes algorithm, and Logistic Regression algorithm (Waikato, 2015). The selection of these machine learning algorithms is based on the
literature and tools available. The results will be further analyzed and the best classification algorithm will be identified.

3.3.2 Machine Learning Algorithms

Algorithms are instructions that must be followed to reach a desired result. In machine learning, these algorithms define the steps to build an optimized model from the data to make predictions. Many algorithms exist and could be used depending upon the type of task as mentioned earlier. In this research, we select four heavily used machine learning algorithms in the URL classification community, namely, Random Forest, Support Vector Machine, Naïve Bayes, and Logistic Regression.

3.3.2.1 Random Forest

Developed by Leo Brieman (2001), Random Forest algorithm is one of the most famous algorithm for classification task. It contains an ensemble of classification trees where each tree depends on the values of a random vector sampled independently and has same distribution. Let ‘n’ be the number of training observations with ‘N’ number of features (dimensionality). To determine the decision node at a tree, we choose k << N as the number of variables to be selected. We select a bootstrap sample from n observations in the training set and use rest of the observations to estimate the error in the testing phase. ‘k’ is chosen randomly as a decision at a certain node and the best split is calculated based on k. Trees are never pruned in order to achieve low bias.

There are several advantages of random forest. First, it can handle large number of features. Second, during forest building phase, it generates unbiased estimation of generalization error. Third, it handles missing data well. However, Random Forest lacks reproducibility as the process is random.
3.3.2.2 Support Vector Machine

Support Vector Machine is a widely used state-of-the-art binary classifier especially when dimensionality or number of features is high. The objective is to derive an optimal separating hyperplane between two classes by maximizing the margin between the class’s closest points. Suppose we have a linear discriminating function and two linearly separable classes with target values +1 and -1.

The equation for the hyperplane is:

\[ w \cdot x_i + w_0 > 0 \text{ if } t_i = +1 \]  \hspace{1cm} (1a)

\[ w \cdot x_i + w_0 > 0 \text{ if } t_i = -1 \]  \hspace{1cm} (1b)

where \( x_i \) is any point in the data, \( t_i \) is the target value and \( w \) is the normal vector to the hyperplane. The distance of \( x_i \) to a hyperplane is \( |w \cdot x_i + w_0|/||w|| \), and the distance to the origin is \( |w_0|/||w|| \).

Although SVM is very popular, it suffers from several drawbacks. First, it requires huge number of computations to train the data. In other words, SVM is computationally intensive. Second, it is very sensitive to noisy data and hence it is prone to overfitting.

3.3.2.3 Naïve Bayes

Naïve Bayes classifier is built on the basis of Bayes Theorem (Bayes, 1763) in probability theory and statistics. It defines the probability of an event based on the related events. Suppose A and B are two events, and \( P(A) \) and \( P(B) \) are the corresponding probabilities of those events to occur. Then, according to Bayes Theorem, the probability of event A provided event B is true is:
\[ P(A|B) = \frac{P(A)P(B|A)}{P(B)} \] (2)

It is a simple yet powerful and very famous algorithm used in many machine learning applications. Naïve Bayes is a probabilistic classifier that works efficiently when the dimensionality of feature vector is high and individual features are distributed independently. Let \( P(x|y) \) denote the conditional probability of the feature vector ‘x’ with the label ‘y’. Then,

\[ P(x|y) = \prod_{j=1}^{N} P(x_j|y) \] (3)

From Bayes Theorem, assuming malicious and benign URL having equal probability, the posterior probability that feature vector \( x \) belongs to malicious URL is:

\[ P(y = 1|x) = \frac{P(x|y=1)}{P(x|y=1)+P(x|y=0)} \] (4)

3.3.2.4 Logistic Regression

Logistic regression is the most widely used algorithm in classification problem. It is a binary data prediction model. It is modeled based on the probabilities of possible outcomes in a single trial using logistic function or sigmoid or logit function represented as:

\[ \log \frac{p(x;\beta)}{1-p(x;\beta)} = \beta^\top x \] (5)

where \( x \) is a vector of \( p \) predictors and \( \beta \) is a \( p \times 1 \) vector of regression parameters.

Although it is similar to linear regression, Logistic Regression differs in assumption that conditional distribution is Bernoulli distribution and probabilities are restricted to 0 or 1. Logistic regression is represented as:
\[
\log \frac{p(x)}{1-p(x)} = \beta_0 + \sum \beta_i x_i \quad (6)
\]

3.4 Summary

We adopt and use Peffers et al.’s research methodology for this dissertation work. Our objective is to design an effective malicious short URL classifier using the features collected from online social networks. The method used for the classifier is machine learning. We will identify the best machine learning algorithm and the most effective feature set for malicious URL classification in the research.
Chapter 4 CONSOL: An Online Social Network Based Short URL Classifier

System development is a systematic process including defining problem, designing and developing solution, testing, and implementing the solution. Problem definition states the information system problem that the new system is attempting to solve. Design provides a blueprint that describes the architecture of the artifact to be developed. Development uses the design to bring the new system to life. Testing is a process of testing whether the functional and non-functional requirements derived from the problem definition are met. Finally, system development is also an iterative process. One may have to go back and address the problem in the system at any phase during the process.

Our main objective in this research is to develop a classifier that can effectively classify short URLs using the features collected from an online social network. In order to develop a system and meet its intended objective, its internal structure, working mechanisms, input, and output must be carefully designed. This chapter presents our design and development of the short URL classifier, CONSOL.

4.1 CONSOL Classifier

Our literature review indicates that the existing works do not provide a satisfactory solution. First, existing solutions rely on the information acquired from third parties. However, it is not always possible to obtain information from a third party service provider. There are hundreds of URL shortening service providers on the Internet. Some of them are free such as Bitly, TinyURL, Google, Owly, and Twitter. Some of
them are proprietary service providers such as New York Times, Washington Posts, Juniper, and Cisco. A few service providers provide APIs (Application Program Interface) for other parties. However, APIs from service providers is not always available. Many proprietary service providers do not provide any information to use their services. Second, the accuracy obtained from the existing solutions is not satisfactory. The best accuracy achieved in classification is 90.81% by Wang et al. using click traffic features (D. Wang et al., 2013). We believe a more effective classifier for short URLs can be developed using the information available on OSNs.

We argue that features from OSNs alone can be used to effectively classify short URLs. The information available on OSNs includes, but is not limited to, personal information, social information, membership information, user posts, etc. OSNs also contain information like group membership, social graphs, user connections, friends, followers, etc. All of the information has unique characteristics and should be studied thoroughly. We aim to leverage the information available on OSNs to develop a classifier for malicious short URLs. The success of this research will have a significant contribution in the information security community and particularly in the URL classification and social media security.

**4.1.1 Conceptual Model**

The proposed classifier model is shown in Figure 4-1. The model meets the goal of leveraging features from online social networks to develop a classifier for short URLs. The principle used in the development is as follows:

“Any social network users have social structure and generate content based upon certain context”.
Therefore, we propose to use the features from social networks to classify short URLs. The features are broadly classified into three main categories, i.e., content features, context features, and social features. The terms are self-explanatory but will be defined and explored later in the chapter.

![Conceptual Model for CONSOL](image)

**Figure 4-1 Conceptual Model for CONSOL**

### 4.1.2 Architecture Diagram

The architecture diagram of the CONSOL is shown in Figure 4-2. It includes 4 components, i.e., streaming data collector, data preparation, feature extractor, and machine learning. Machine learning consists of two sub-components, i.e., labeling and classifier.
Streaming Data Collector: Streaming Data Collector is responsible for collecting data from Twitter. Twitter’s streaming APIs are used to collect data. Twitter provides data in JSON format. The JSON data is parsed using a python script. Desired data will be collected and stored in a flat file. We are particularly interested in the tweets that include Bitly short URLs. Tweets that do not include Bitly short URLs will be skipped in the process.

Data Preparation: The data collected by the Streaming Data Collector contains huge amount of information. Only a handful of information is required to develop a classifier. The rest is considered as noise and irrelevant information. Irrelevant information such as profile picture, background color, contributors, translator, protected,
and truncated will be removed. Data Preparation also deals with the data with errors and missing values and prepares them for further processes.

**Feature Extractor:** Feature Extractor extracts relevant features from tweets and OSNs. Initial features are selected based on the literature review. We classify the features into three categories, i.e., content features, context features, and social features. Feature selection will help assess the contribution of each category for short URL classification.

**Machine Learning:** Machine Learning consists of two parts. First, a labeled dataset is created using blacklisting service. Second, a classifier is developed for malicious short URLs.

- **Labeling:** Labeled dataset is obtained by performing URL lookup against blacklists such as VirusTotal and PhishTank. The labels are represented in binary format, 1 for malicious and 0 for benign. We use two blacklist services, i.e., VirusTotal and PhishTank, to create the labeled dataset. After cleaning the data, full URLs of the corresponding short URLs are retrieved from Bitly. Each URL will be checked against the two online URL scanning services. A labeled dataset will be created from the results.

- **Classifier:** Artifact will take the labeled dataset and train itself using 10-fold cross validation using a machine learning algorithm to build a classifier. The classifier will then predict the malignity of the short URLs using a new set of testing data. Performance metrics will be computed for each machine learning algorithm. The algorithm, which performs the best, will be used for further analysis.
4.2 Data Collection

Data is collected from Twitter using Twitter’s Streaming APIs (Twitter, 2015). Streaming APIs provide access to Twitter’s global stream of tweets. There are several challenges in the data collection. First, the volume of data can grow significantly. Collecting one hour of tweets requires more than 10 GB of space. Due to the limitations of our storage space, the data has to be trimmed. Since we are only interested in the tweets including Bitly short URLs and the rate of the data fluctuates, the program needs to consume the data quickly without breaking the API connection to the Twitter. To address these issues, we decided to slightly process the data while collecting it. We remove the information that is irrelevant to our goal. The information removed from the tweets includes profile background color, profile picture, and background theme. We adjust the script such that minimal processing is conducted, not to exhaust storage, during data collection without breaking the connection. The data is collected in JSON format (“Introducing JSON,” 2015).

Tweets are very large in size and arrive at a high rate. We did not collect the tweets for 24 hours continuously during a day. Tweets were collected for a few hours in a different time interval during a day to enrich the diversity of the data. There are millions of tweets in a typical day. If we capture all tweets, it will take significantly long time just to label the data. In this study, we collected tweets for 3 hours at a different time every day over the course of a week in April 2015. We collected 1.75 million tweets during the data collection period. Our final dataset contains almost 40GB of data.
4.2.1 Tweet Data

The fields in a tweet contain the necessary information about the tweet, the user, and the contents it possesses. Without proper understanding of the data, it cannot be processed properly and the expected results cannot be achieved. The data returned from the Twitter Streaming APIs is in JSON format. The JSON file includes different attribute-value pairs or fields. A typical tweet contains the following fields:

*Tweet Information*: This field is at the beginning of the tweet. It contains tweet specific information such as creation time and date, tweet ID, tweet text, source of tweet, if a tweet is truncated, geographical information, coordinates, place, contributors, retweet count, and favorite count. However, the contents may vary depending upon the type of tweets (e.g., retweet, replies).

*User Information*: This field contains the information specific to the creator of a tweet. It contains user’s unique user ID, name, screen name, location details, user URL, description, protected, verified, friends count, followers count, account creation date, time zone, etc. If a tweet is retweeted, the information about the retweet is also contained in this entry.

*Entities Information*: This field contains other information about a tweet. It contains all the hashtags, URLs and their corresponding full URLs (Twitter automatically shortens URLs), user mentions (name, id, etc.), any symbols, etc. Note that if a tweet includes multiple URLs, each URL will have its separate entry in the tweet.
4.2.2 Data Labeling

Data in nature has certain characteristics to categorize them. For example, photos, videos, music, and books can be classified into different categories such as happy, action, romance, sad, and adventure. Data can be labeled using human intelligence.

Labeling data is the process of tagging the data with a meaningful ‘tag’ to represent which category it belongs to. In our case, we are interested in two categories, i.e., malicious and benign. Labeling data is very important process in supervised machine learning. After the labeled data is obtained, appropriate machine learning algorithms can be applied to derive mathematical model that represents the relationship between the data and its tag.

We leverage popular blacklist services to label our dataset. The short URLs are extracted from tweets and their corresponding long URLs are extracted using the URL shortening service. Bitly APIs and python script are used to obtain the corresponding full URLs. The full URLs are then submitted to the blacklist service to determine the malignity of the URLs. The results are returned in JSON format and parsed by a script to obtain the label in binary format, i.e., 1 representing malicious URL and 0 representing benign URL. The results are then used to label the data. The blacklist services used in our research are VirusTotal and PhishTank. We rely heavily on the VirusTotal because VirusTotal scans a URL using 56 different online services. If any of the 56 services labels the URL as malicious, the URL is considered malicious.

There are several challenges when performing lookups against the blacklist services. The API service from these blacklists usually comes with a low rate of lookups. We were able to request a higher rate for academic research and obtained a daily lookup
rate up to 60,000 URLs/day from VirusTotal. Since the lookup rate from VirusTotal was satisfactory for our purpose, we use the VirusTotal as the main blacklist service in this research. PhishTank was used to label small datasets during the earlier phase in this project. Note that PhishTank result is also obtained from the VirusTotal.

Among all the 1.75 million tweets, most of the entries are not found in the blacklist. Our goal is to create a labeled dataset. Thus, the short URLs whose status cannot be identified were skipped. Some of the URLs took long time for processing. For those URLs, we tuned our script not to retrieve the results. Many of the tweets contained foreign languages and caused processing errors. We decided to remove all those tweets for simplicity. Our final labeled dataset (DS1) contains 50,291 true positive entries and 410,362 true negative entries.

4.3 Social Network Features

As shown in Table 4-1, features can be broadly classified into three categories, i.e., content features, context features, and social features. All three categories of features can be found on social networks.

**Content Features:** User-generated content is one of the main characteristics of online social networks. Users of OSNs produce massive content on a daily basis. On average, 58 million tweets are generated on Twitter alone (Brain, 2015). User-generated content includes messages, photos, videos, URLs, etc. In this research, we target to use content features for URL classification. In particular, content features we study include length_of_tweet, number_hashtags, number_urls, and words from bag-of-words such as contains_string_check, contains_string_watch, contains_string_sex, contains_string_klik,
contains_string_click, contains_string_amazing, contains_string_interested, and contains_string_single. These features can be derived from the tweet contents.

**Context Features:** Context features include the environment or settings where an event occurs. Tweets carry information related to an environment. It usually contains information related to news, photo sharing, web content, emotions, etc. Context of a tweet can reveal lots of information about a particular tweet and its content. In this research, we leverage five context features, i.e., in_reply_to_status, in_reply_to_user, retweet_count, favorite_count, and relevance. The first four features are simply obtained and extracted from a tweet. Relevance is obtained by leveraging tweet trends. Twitter produces top tweet trends using an algorithm around a user’s location and around the globe. The idea is to determine the hottest topics matching a user’s interests. We define a tweet is relevant if the tweet contains any top 10 trends at the time of tweet collection. Our program collects a list of top trends on twitter USA every minute during data collection phase. A python script was developed for the purpose.

**Social Features:** OSN contains huge amount of social information. This information conveys a lot about the user and the user’s interests. Typical social information contains friends, followers, family, personal information, group memberships, number of posts/tweets, etc. Social features also contain information about how the user’s messages/posts/tweets are absorbed in social networks. In this research, we use the social features that are directly obtained from a tweet. In particular, we leverage six social features, i.e., friends_count, followers-count, favorites_count, user_following, statuses-count, and follow_req_sent.
Table 4-1 summarizes all the features, the category they belong to, and their descriptions.

### Table 4-1 Features and Their Descriptions

<table>
<thead>
<tr>
<th>Categories</th>
<th>SN</th>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>1</td>
<td>Length_of_Tweet</td>
<td>Length of characters in tweet</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Contains_str_check</td>
<td>If word “check” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Contains_str_watch</td>
<td>If word “watch” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Contains_str_sex</td>
<td>If word “sex” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Contains_str_klik</td>
<td>If word “klik” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Contains_str_click</td>
<td>If word “click” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Contains_str_amazing</td>
<td>If word “amazing” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Contains_strInterested</td>
<td>If word “interested” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Contains_str_single</td>
<td>If word “single” exists in tweet</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Number_hashtags</td>
<td>How many hashtags does the tweet have?</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Number_URLs</td>
<td>How many urls does the tweet have?</td>
</tr>
<tr>
<td>Context</td>
<td>12</td>
<td>In_reply_to_status</td>
<td>If tweet is in reply of other tweets</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>In_reply_to_user</td>
<td>If tweet is in reply of other user</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>Retweet_count</td>
<td>How many retweets user has made?</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>Favorite_count</td>
<td>How many users favorited the tweet?</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Relevance</td>
<td>Contains any relevant trends?</td>
</tr>
<tr>
<td>Social</td>
<td>17</td>
<td>Followers_count</td>
<td>How many followers user has?</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Friends_count</td>
<td>How many friends user has?</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>Favorites_count</td>
<td>How many tweets are favorited by user?</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>Statuses_count</td>
<td>How many tweets user has made?</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>User_following</td>
<td>How many other users, she is following?</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>Follow_req_sent</td>
<td>Is any follow request sent?</td>
</tr>
</tbody>
</table>

Bag-of-words technique has been heavily used in the field of data mining such as information retrieval and natural language processing (Salton & Michael, 1983). The idea is to represent the text as a bag full of words disregarding its grammar but maintaining its multiplicity. In this research, we use bag-of-words technique to derive popular words in
true positive set of malicious tweets. The result from bag-of-words is used as content features in our classifier.

We derive bag-of-words by using a simple technique. We leverage frequency of words as a factor for selection of words. The principle behind frequency of words is that the words with higher occurrence are the words of value. We retrieved the top 100 words based on the frequency of occurrence and performed manual inspection to remove stop words and non-relevant words. The labeled dataset (DS1) is parsed to collect the tweet contents from the true positive malicious entries. The tweet contents are then processed using the python script to count the frequency of words. The script generates a list of top 100 words, which were analyzed manually to select the top 8 words after removing the non-significant words such as articles, pronouns, URLs, hashtags, etc. Figure 4-3 shows the frequency of these words.

![Figure 4-3 Frequency Table for Bag-of-Words](image-url)
4.4 Feature Selection

Feature selection deals with the process of identifying the most important features and removing the irrelevant features. It seems that more features should result in higher performance of an algorithm. However, it is not the case. Having irrelevant features can make the model overfit. Overfitting is the problem in machine learning where the model performs very well during training and performs poorly when it is tested. To achieve the best performance of a machine learning algorithm, all the irrelevant features must be excluded.

Feature selection is often confused with feature extraction. While the latter deals with the creation of new features by utilizing the original features, the former deals with selecting the best subset of features that performs well. Feature selection algorithms can be generalized into two categories, i.e., wrapper methods and filter methods. Wrapper methods evaluate the features based on their worth with the learning algorithm. Filter methods evaluate features using heuristics on the characteristics of data (Kohavi & John, 1997).

Our preliminary analysis of the dataset found that four features in our dataset do not have any variance. In another words, the values of these features are all the same; in this case it is all zeros. Particularly, retweet_count, favorite_count, user_following, and follow_req_sent are the features with zero values. We decided to immediately remove these features from the feature set because these features are not relevant in predicting malignity of short URLs.
We use filter method in this research. InfoGainAttributeEval function from Weka is used for feature selection. InfoGainAttributeEval evaluates the information gain with respect to the target class.

\[ \text{InfoGain(Class, Attribute) = } H(\text{Class}) - H(\text{Class} \mid \text{Attribute}) \]

The InfoGainAttributeEval function returns the features and their ranks based on the information gain with respect to the target class. Table 4-2 shows the output of the function. The decimal number before the features represents the information gain of that particular feature with respect to the target class.

<table>
<thead>
<tr>
<th>Information Gain</th>
<th>Attribute Number</th>
<th>Attribute Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2682333</td>
<td>16</td>
<td>Statues_count</td>
</tr>
<tr>
<td>0.1631349</td>
<td>13</td>
<td>Followers_count</td>
</tr>
<tr>
<td>0.1427712</td>
<td>14</td>
<td>Friends_count</td>
</tr>
<tr>
<td>0.127612</td>
<td>1</td>
<td>Length_of_tweet</td>
</tr>
<tr>
<td>0.0601966</td>
<td>10</td>
<td>Relevance</td>
</tr>
<tr>
<td>0.0379317</td>
<td>12</td>
<td>In_reply_to_user</td>
</tr>
<tr>
<td>0.0364155</td>
<td>15</td>
<td>Favorites_count</td>
</tr>
<tr>
<td>0.0185196</td>
<td>2</td>
<td>Contains_str_check</td>
</tr>
<tr>
<td>0.0181032</td>
<td>17</td>
<td>Number_hashtags</td>
</tr>
<tr>
<td>0.0121861</td>
<td>6</td>
<td>Contains_str_Click</td>
</tr>
<tr>
<td>0.0068112</td>
<td>11</td>
<td>In_reply_to_status</td>
</tr>
<tr>
<td>0.0059014</td>
<td>3</td>
<td>Contains_str_Watch</td>
</tr>
<tr>
<td>0.0046924</td>
<td>5</td>
<td>Contains_str_Klik</td>
</tr>
<tr>
<td>0.0034791</td>
<td>4</td>
<td>Contains_str_Sex</td>
</tr>
<tr>
<td>0.0028035</td>
<td>8</td>
<td>Contains_str_Interested</td>
</tr>
<tr>
<td>0.0020062</td>
<td>18</td>
<td>Number_URLs</td>
</tr>
<tr>
<td>0.0000726</td>
<td>9</td>
<td>Contains_str_Single</td>
</tr>
<tr>
<td>0.0000124</td>
<td>7</td>
<td>Contains_str_Amazing</td>
</tr>
</tbody>
</table>

**Table 4-2 Output of InfoGainAttributeEval Function of Weka**

4.5 Evaluation Metrics

Evaluation metrics establish the baseline for evaluating the performance of the artifact. Table 4-3 Evaluation Metrics shows the evaluation metrics used in the research. This metrics is widely adopted in machine learning community.
Table 4-3 Evaluation Metrics

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Malicious</td>
<td>TP</td>
<td>Malicious</td>
<td>FN</td>
</tr>
<tr>
<td>Benign</td>
<td>FP</td>
<td>Benign</td>
<td>TN</td>
</tr>
</tbody>
</table>

True positive (TP): TP is the correct identification of truly malicious URL as malicious.

True Negative (TN): TN is the correct identification of benign URL as benign.

False Negative (FN): FN is the incorrect identification of malicious URL as benign.

False Positive (FP): FP is the incorrect identification of benign URL as malicious.

We will also look at precision, recall, F-measure, and the accuracy of the classifiers.

Precision (P): is the rate of correctly identified malicious URLs to all instances.

Recall (R): is equivalent to TP.

F-measure (FM): Harmonic mean between precision and recall.

Accuracy (A): measures the overall rate of correctly detected malicious and benign URLs to all instances.

\[
P = \frac{TP}{(TP+FP)} \quad \text{………………(1)}
\]

\[
R = \frac{TP}{(TP+FN)} \quad \text{………………(2)}
\]

\[
FM = 2 \cdot \frac{P \cdot R}{P + R} \quad \text{………………(3)}
\]

\[
A = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad \text{………………(4)}
\]
4.6 Dataset Description

We created different datasets for the development, analysis, and evaluation of our classifier. The purpose to have many different datasets is to make sure our results are consistent. The datasets vary in size for the purpose they serve. DS1, DS2, and GDB1 are three completely different datasets. They were collected at different time during the testing. RDB1 is a labeled dataset derived from DS1. WDB1 and RDB2 are two labeled datasets derived from DS2. Finally, GDB1 is completely new dataset for comparison against Safe Browsing. Table 4-4 shows the datasets we used in this research.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Dataset</th>
<th>Description</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. At the end of</td>
<td>a. DS1</td>
<td>Labeled dataset at the end of data collection</td>
<td>Total: 460,653 entries</td>
</tr>
<tr>
<td>collection</td>
<td></td>
<td></td>
<td>Malicious: 50,291</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Benign: 410,362</td>
</tr>
<tr>
<td>2. Training and</td>
<td>a. RDB1</td>
<td>Balanced dataset created by reducing the false entries in unbalanced</td>
<td>Total: 100,580 entries</td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td>dataset collected DS1</td>
<td>Malicious: 50,291</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Benign: 50,289</td>
</tr>
<tr>
<td>3. Evaluation</td>
<td>a. DS2</td>
<td>Labeled dataset at the end of collection. This dataset is used to derive</td>
<td>Total: 30,100 entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WDB1 and RDB2</td>
<td>Malicious: 5600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Benign: 24,500</td>
</tr>
<tr>
<td></td>
<td>a. WDB1</td>
<td>Dataset created following the approach used by Wang et Al. from DS2</td>
<td>Total: 10,881 entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Malicious: 5454</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Benign: 5427</td>
</tr>
<tr>
<td></td>
<td>b. RDB2</td>
<td>Evaluation dataset created to compare the classifier to Wang et. Al’s method</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>from DS2</td>
<td>Total: 10,908 entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Malicious: 5455</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Benign: 5453</td>
</tr>
<tr>
<td></td>
<td>c. GDB1</td>
<td>Unlabeled Dataset to compare the performance in real-world</td>
<td>Total: 9986 entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Unlabeled</td>
</tr>
</tbody>
</table>
The data collection, processing, training, and testing were performed on a MacBook Pro with 2.9 GHz core i7 processor and 16 GB of RAM. Program was written in python programming language using Eclipse IDE (Integrated Development Environment). Python is a general purpose interpreted programing language and provides huge community support. It is rich in libraries for machine learning and data mining projects. Some of the libraries used in our research include Scikit-learn, Pandas, and Graphlab.

4.7 Classifier Training and Preliminary Testing

In machine learning, training a classifier refers to providing a labeled dataset to a machine-learning algorithm, which will derive a mathematical function that explains the relationship between the data and the corresponding label. The mathematical function derivation is an iterative process in which the algorithm goes through multiple iterations to minimize the error. The error refers to the difference between the prediction and the actual value. The algorithm tunes the parameter for the features depending on the error value to minimize the error. The final result is a mathematical model that represents the relationships between the input features and the label. The mathematical model is then tested against a new testing dataset to measure the effectiveness. This new testing dataset should not be the one that is used in training phase. The effectiveness of the model is usually measured in terms of accuracy.

As mentioned earlier, we chose four machine learning algorithms, i.e., Random Forest, Support Vector Machine, Naïve Bayes, and Logistic Regression, for the short
URL classifier. The algorithms are selected from the literature review based on community tools and existing technique approaches.

The labeled dataset used to train and test the classifier in this research is RDB1. We use 10-fold cross validation technique to utilize our dataset fully. K-fold cross validation, also called rotation estimation, is the technique of randomly splitting dataset into k mutually exclusive subsets (folds) of approximately equal size. Then, the classifier is trained k times using (k-1) subsets and tested against 1 subset at each iteration. The accuracy is computed at each iteration for the testing dataset. Finally, overall accuracy is computed as a mean of accuracies at each iteration.

We take the results from feature selection with InfoGainAttributeEval from Weka. We perform several tests on the features to find the best feature set. We decide to remove that feature which has the lowest information gain, one at a time and run the selected algorithms with 10-fold cross validation. The procedure is repeated until all but one feature is removed from the model. The lowest ranking feature removed at an earlier phase is not added back to the feature set while moving forward in the test. Our objective is to minimize the error. We compute Root Mean Square Error (RMSE) at each run. The model with the low RMSE, the best accuracy, precision, recall, and F-measure and the corresponding feature set is recorded and selected for further evaluation. The results of the four algorithms are shown in the figures below.
Figure 4-4 RMSE Logistic Regression

Figure 4-5 RMSE Naive Bayes
Figure 4-6 RMSE Support Vector Machine

Figure 4-7 RMSE Random Forest
Figure 4-4 RMSE Logistic Regression to Figure 4-7 RMSE Random Forest show the changes of RMSE values in each iteration, removing one feature at a time. In Figure 4-4, using Logistic Regression, we achieve the lowest RMSE when removing only one feature, contains_str_amazing, from the feature set. Hence, it contains all the other features in the Information Gain that rank above contains_str_amazing. The same concept applies to all other machine learning algorithms and the results are shown in Table 4-5.

Table 4-5 Performance Metrics for Feature Selection

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>RMSE</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>0.479</td>
<td>61.052</td>
<td>0.639</td>
<td>0.611</td>
<td>0.590</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.449</td>
<td>67.652</td>
<td>0.709</td>
<td>0.677</td>
<td>0.664</td>
</tr>
<tr>
<td>SVM</td>
<td>0.493</td>
<td>75.718</td>
<td>0.813</td>
<td>0.757</td>
<td>0.746</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.207</td>
<td>94.502</td>
<td>0.945</td>
<td>0.945</td>
<td>0.945</td>
</tr>
</tbody>
</table>

Table 4-5 Performance Metrics shows the RMSE and performance metrics computed from our testing using the best feature set, i.e. with lowest RMSE, for each machine learning algorithm. As shown in the table, Random Forest algorithm outperforms all other algorithms significantly in every aspect: RMSE, precision, recall, f-measure, and accuracy. Random Forest has the lowest RMSE, highest accuracy, highest precision, highest recall, and highest F-measure. The closest rival to Random Forest is Support Vector Machine algorithm with only 75.718% accuracy, 0.813 precision, 0.757 recall, and 0.746 f-measure. The RMSE for Random Forest is 0.493, significantly higher. This is a significant difference in all aspects. Our testing also reveals the best feature set for the
Random Forest algorithm as shown in Table 4-6 Selected Features for Random Forest with Filter Method.

<table>
<thead>
<tr>
<th>S.N</th>
<th>FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Length of tweet</td>
</tr>
<tr>
<td>2</td>
<td>Contains_str Check</td>
</tr>
<tr>
<td>3</td>
<td>Contains_str Watch</td>
</tr>
<tr>
<td>4</td>
<td>Contains_str Sex</td>
</tr>
<tr>
<td>5</td>
<td>Contains_str Click</td>
</tr>
<tr>
<td>6</td>
<td>Contains_str klik</td>
</tr>
<tr>
<td>7</td>
<td>Contains_str interested</td>
</tr>
<tr>
<td>8</td>
<td>Relevance</td>
</tr>
<tr>
<td>9</td>
<td>In_reply_to_status</td>
</tr>
<tr>
<td>10</td>
<td>In_reply_to_user</td>
</tr>
<tr>
<td>11</td>
<td>Favorites_count</td>
</tr>
<tr>
<td>12</td>
<td>Friends_count</td>
</tr>
<tr>
<td>13</td>
<td>Followers_count</td>
</tr>
<tr>
<td>14</td>
<td>Statuses_count</td>
</tr>
<tr>
<td>15</td>
<td>Number_hashtags</td>
</tr>
<tr>
<td>16</td>
<td>Number_URLs</td>
</tr>
</tbody>
</table>

Note that the four algorithms evaluated achieve the best performance with different feature set. Figure 4-8 also indicates that Random Forest algorithm outperforms all other algorithms significantly for malicious short URL classification.
Wrapper method was not selected for feature selection because of its complexity and the limitation of computing resources available during the research. However, the problem with feature selection using filter method is that the model tends to bias. To validate the results, we also applied wrapper method on Random Forest algorithm to compare the performance from filter method. Our objective here is to show that we achieve accuracy significantly close to that of wrapper method despite not using the wrapper method for feature selection.

We run Weka with wrapper method on RDB1 dataset using 10-fold cross validation and best-first search approach. The results are shown in Figure 4-9 Results of Wrapper Method Feature Selection.
As shown in Figure 4-9, the wrapper method selected 13 features for the classifier. We tested the short URL classifier again using the Random Forest algorithm and the 13 features from the wrapper method. The results are shown in Table 4-7 Wrapper Feature Selection Result.

### Table 4-7 Wrapper Feature Selection Results

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>RMSE</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>0.207</td>
<td>94.54</td>
<td>0.946</td>
<td>0.945</td>
<td>0.945</td>
</tr>
</tbody>
</table>

Comparisons between Table 4-5 Performance Metrics and Table 4-7 Wrapper Feature Selection Result show that the filter method performs significantly close to the
wrapper method. Thus, our testing in Table 4-5 Performance Metrics using filter method is appropriate. The features selected from filter method and wrapper methods are very similar. Filter method selects three more features, i.e., click, klik, and interested, than wrapper method. A close comparison of the filter method and the wrapper method could be further investigated in the future.

4.8 Summary

We design and develop a classifier, CONSOL, for malicious short URLs. The classifier utilizes features such as content features, context features, and social features available on online social networks to identify malicious short URLs. The classifier relies on blacklist services for training purpose. However, the classifier does not depend on any third parties during its operation. We test four machine learning algorithms, i.e., Random Forest, Naïve Bayes, Support Vector Machine, and Logistic Regression, for the classifier. Our testing indicates that Random Forest algorithm outperforms all other algorithms with 94.5% accuracy with its closest rival at 75% accuracy. We further evaluate the best feature set for the Random Forest algorithm and identify 16 important features for short URL classification. Our testing also shows that the Random Forest algorithm performs equally well using filter method and wrapper method.
Chapter 5 Evaluation and Results

We present the design and development of the short URL classifier, CONSOL, in Chapter 4. Our testing shows that the Random Forest algorithm performs the best among all the machine learning algorithms tested. However, there are more questions to be answered to assess the performance of the classifier, e.g., how well CONSOL performs compared to other existing approaches, how the accuracy changes with respect to features in different scenarios, and what the misclassified malicious short URLs have in common. This chapter further evaluates the effectiveness of the CONSOL compared with the existing solutions and Google Safe Browsing.

5.1 Comparison against Existing Work

To further evaluate the performance of CONSOL, we looked at how CONSOL performs when compared with other approach in the literature. As discussed in the literature review, there has been little research conducted on short URL classification. Among all the related work, the research conducted by Wang et al. (2013) is particularly interesting because their research leverages click-traffic features obtained from the URL shortening service providers to classify short URLs. In our research, we target to show that short URLs can be classified using the features obtained from the OSNs and it can still achieve high accuracy. Thus, we chose Wang et al.’s work to be compared with the CONSOL.

Wang et al. (2013) studied the misuse of short URLs and the characteristics of the spam and benign short URLs using a case study. The findings are then used to detect spam URLs on Twitter. Specifically, Wang et al. used click-traffic features for spam URL detection. They collected tweets including Bitly URLs and retrieved the click traffic
features using Bitly APIs. They utilized 9 click traffic features to develop the classifier and achieved an accuracy of 90.81% using Random Tree algorithm. Their dataset contains 641,423 short URLs (18,496 spam, 622,927 genuine).

We decide to leverage the same raw data to minimize the bias in order to make the results comparable. We leverage WDB1 and RDB2 datasets, which contain Bitly short URLs prepared from the same raw dataset DS2. WDB1 dataset contains the features used in Wang et al.’s study. RDB2 dataset contains the features for the CONSOL. WDB1 has 10,881 total entries with 5,427 malicious and 5,454 benign entries. RDB2 has 10,908 total entries with 5,453 malicious and 5,455 benign entries. Note that there is slight difference in the number of entries. This is due to the error in data processing in the program. We skipped few URLs due to long processing time when labeling data. We then run 10-fold cross validation using the four different algorithms on both datasets. The results are shown in Table 5-1.

Table 5-1 Comparison against Wang et al.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>WDB1</th>
<th>RDB2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pr</td>
<td>Re</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.926</td>
<td>0.924</td>
</tr>
<tr>
<td>SVM</td>
<td>0.915</td>
<td>0.908</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.705</td>
<td>0.704</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.612</td>
<td>0.590</td>
</tr>
</tbody>
</table>
The result in Table 5-1 show that CONSOL performs slightly better than the approach used in Wang et al.’s work. Random Forest algorithm outperforms all other algorithms in the study. In Wang et al.’s approach, Support Vector Machine performs very close to Random Forest algorithm as shown in Figure 5-1 Graphical Representation of Comparison against Wang et al. (WDB1)
Comparison against Wang et al. (WDB1) while others fall far behind. In CONSOL approach, the results are fairly consistent. The results demonstrate that short URLs can be effectively classified using the features from online social networks and still achieve high accuracy.

Wang et al. achieved 90.81% accuracy using their dataset in their original work. Using WDB1, we are able to achieve 92.25% accuracy using Random Forest algorithm. This increase in accuracy can be explained by the objective and the use of the dataset. The study by Wang et al. focused on spam short URL detection while our objective is to classify short URLs in general as benign or malicious which includes spam URLs as well. Wang et al. used 5 different public blacklists to label the data, i.e., Google Safe Browsing, McAfee SiteAdvisor, URIBL, SURBL, and Spamhaus. Our implementation uses the blacklist from VirusTotal which scans the URLs from 56 different online URL scanning services. Finally, their dataset was heavily unbalanced and the authors did not provide any information on balancing the dataset. We believe the discrepancy is due to the different blacklists used, unbalanced dataset, and the objective of the research.

5.2 Comparison against Google Safe Browsing

To evaluate the effectiveness of the CONSOL in real world scenario, we also compare the results of our classifier with Google’s Safe Browsing blacklist. The objective of this comparison is not to compete CONSOL against Safe Browsing since Safe Browsing is usually updated after rigorous process based on machine learning as well as manual analysis. Safe Browsing is a blacklist maintained and updated by Google. It can be used by other applications to look up URLs for malignity. The blacklist contains
phishing URLs, malware URLs, and unwanted software pages. The whole URL labeling process is complex and out of the scope of this dissertation.

We compare the performance of CONSOL against Google Safe Browsing blacklist. We collect the tweets including ‘unlabeled’ Bitly short URLs, extract the features, and then classify short URLs using CONSOL. The results from our classifier are then compared against Google Safe Browsing blacklist.

We convert the short URLs obtained from the tweets to their corresponding full URLs. We then identify the URL malignity using Google Safe Browsing service. Since Google Safe Browsing APIs limit the rate of lookup to 10,000 URLs per day, we collect 10,000 tweets including short URLs for the evaluation purpose (GDB1). Literature suggests that it takes up to 23 days for the malicious URLs to appear on blacklists (Sheng et al., 2009). We perform the lookup against the Safe Browsing service continuously every day for up to 23 days. We also perform lookup on 30 days, 45 days, and 60 days. The results (Figure 5-3 Comparison against Safe Browsing) are then used to compare the effectiveness of our classifier.
CONSOL is able to label 503 entries from GDB1 as malicious short URLs on Day 0. There was no malicious short URL detected by Google Safe Browsing. On 4\textsuperscript{th} and 5\textsuperscript{th} days, Google Safe Browsing detected one new URL as malicious that was not labeled by CONSOL. However, on the 5\textsuperscript{th} day, Google Safe Browsing labeled the earlier detected malicious short URL as benign. We believe it was a false positive from Safe Browsing. We were interested to see how Safe Browsing performs in comparison to the CONSOL. Thus, we ran the test until 23\textsuperscript{rd} day. However, Safe Browsing caught none of the URLs during the 23 days. This led us to seek other ways to verify whether the URLs caught by CONSOL were indeed malicious.

Several techniques exist and can be used to check for the malignity of a URL, e.g., using sandboxing or manual analysis, etc. In this research, we decide to leverage...
VirusTotal blacklist in order to confirm the malignity of the URLs flagged by the CONSOL.

On 30\textsuperscript{th}, 35\textsuperscript{th}, and 40\textsuperscript{th} day, Safe Browsing flagged one URL as malicious, which was not detected by the CONSOL. We performed lookup against VirusTotal from 35\textsuperscript{th} day. On the 35th day, VirusTotal caught 45 URLs as malicious, which were also identified by the CONSOL. On the 45\textsuperscript{th} day, Safe Browsing again removed the URL which was labeled as malicious earlier. VirusTotal identified the same 45 URLs as malicious on the 45th day. Safe Browsing did not catch any other URLs till 60\textsuperscript{th} day of our experiment. VirusTotal caught 46 malicious URLs on the 60\textsuperscript{th} day.

The results show that indeed CONSOL performs well in the real world. We were not able confirm how many of the short URLs caught by CONSOL are malicious due to the time constraint. However, our comparisons show that the results from the CONSOL are trustworthy and promising.

### 5.3 Analysis of Content, Context, and Social Features

We are also interested to assess the contribution of each category of features to the overall performance of CONSOL. We analyze the performance of each category using RDB1 dataset. We split the RDB1 dataset into training and testing sets with 70-30% split respectively. Our training set contains 70,406 entries and our test set contains 30,174 entries. We ran the Random Forest algorithm on the training set to train the classifier and test the classifier using the testing dataset. We achieved an overall accuracy of 93.24% on the testing dataset with all features.

<p>| Table 5-2 Accuracy of the Classifier with Different Feature Sets |</p>
<table>
<thead>
<tr>
<th>Feature Sets</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>All features</td>
<td>93.24%</td>
</tr>
<tr>
<td>Social features only</td>
<td>91.47%</td>
</tr>
<tr>
<td>Content features only</td>
<td>70.58%</td>
</tr>
<tr>
<td>Context features only</td>
<td>40.92%</td>
</tr>
</tbody>
</table>

As shown in Table 5-2, the social features alone can accurately classify 91.47% of the total malicious short URLs. Content features can classify 70.58% of the total malicious short URLs while context features can classify only 40.92% of the total malicious short URLs. From Table 5-2, it is clear that social features are the most powerful features with higher predictive power. After social features, content features perform well but are significantly poor than social features. The performance of the context features is very poor in identifying malicious short URLs. Hence, social features and content features are more effective than context features when used to identify malicious short URLs.

To measure the percentage of short URLs correctly predicted by each category of features, we added an ID field in the dataset using Weka, to identify each unique short URLs. We tested the classifier with combined features and each feature set separately and exported the results to a CSV file. We then parsed the CSV file to identify the unique short URLs classified by each category. The process was repeated for all three categories and the combined feature set. The results are shown in Figure 5-4. Note that the testing here is slightly different than the testing in Chapter 4. Previous tests in Chapter 4 used 10-fold cross validation. The dataset here is split into training and testing sets with 70-30% split. This is because we want to identify the short URLs that are caught by each category of features and to achieve that we need to make the dataset stable. The result is shown in Figure 5-4.
From the analysis in Figure 5-4, we found that only 27.44% of the short URLs are caught by all three categories of feature sets. Figure 5-4 also proves that social features are indeed the most powerful features and contributes extra 16.50% unique short URLs. Content features contribute 1.87% unique short URLs that are not caught by any other categories while context features only contribute 0.7% unique short URLs. However, there are 0.52% of the short URLs that are not caught by either of the individual feature sets but are caught when combining all features together. This shows the accumulative power of the features.

5.4 Theoretical Analysis

Empirical evidence on the predictive capability of different sets of features suggests that social features have significant power to classify short URLs. Results from Table 5-2 shows that social features alone can predict malignity of short URLs with up to 91.47% accuracy. This can be supported by the fact that malicious users have more
control to change the content and the context of the tweets. However, their social structure is more rigid and less flexible to change. This is also supported by the existing literature. Yang et al. found malicious users’ social network forms a small-world network and can be detected using social relationships and semantic coordination (Yang, Harkreader, Zhang, Shin, & Gu, 2012). Stringhini et al. found a pattern in spammer’s profiles and leveraged those patterns to classify profiles in social network (2010).

Content features are able to predict with 70.58% accuracy as shown in Table 5-2. This result is less than expected. This can be due to the fact that the content features do not have much difference between malicious and benign posts. This can be justified by the fact that malicious users have more control over how they present their tweets and thus short URLs to evade detection. For example, a malicious user can post a malicious short URL with different tweet length, keywords, hashtags, along with genuine URLs.

Context features have poor predictive capability with accuracy of only 40.92% (Table 5-2) although some features are very prevalent with malicious postings, e.g., relevance. Predictive capacity of context features is low. It might be due to the few context features we have in the classifier.

5.5 Analysis of Missing Short URLs

CONSOL achieves an accuracy of 94.5% for short URL classification. We are interested in analyzing the remaining 5.5% of the URLs that were missed in our classifier. For this purpose, we write a python script to collect all the features for those missed short URLs in RDB1 dataset. This sample contains false positive and false negatives prediction from our classifier on the RDB1 dataset. We then perform analysis on each category of features individually to find out the cause why those short URLs are
missed. Based on the statistical characteristics (mean, standard deviation) of all the features, we did not find any unusual characteristics on these entries. In the case of false negative entries, the only justification is that these tweets are posted from compromised accounts or social network users post tweets including short URLs unaware that they are malicious.

5.6 Summary

To further evaluate the performance of the CONSOL, we compared the CONSOL with the existing work by Wang et al. As shown in our testing, our classifier performs slightly better than Wang et al.’s work. Note that Wang et al.’s work uses click traffic features obtained from Bitly while our classifier achieves slightly better performance and does not depend on any third parties during its operation. We also compared the performance of CONSOL against Google Safe Browsing. We found that CONSOL performs better compared to Safe Browsing in true positive detection and speed of detection. We also performed theoretical analysis of why CONSOL works better. The use of social features is observed to be very powerful in classifying short URLs. For those malicious short URLs missed in the classifier, we found that there are no statistical anomalies in the missing short URLs.
Chapter 6 CONSOL to Other Online Social Networks

Our evaluation and analysis in 55 show that the short URL classifier, CONSOL, is practical and effective compared to the existing solutions. We design, develop, and evaluate the classifier using the data collected from Twitter. There are hundreds of OSNs all over the world. While business models for these services may vary, the general principles in which they operate are the same. All the online social networks possess information that can be further classified to content features, context features, and social features. Thus, it is very likely that these OSNs can also use CONSOL for short URL classification. This chapter demonstrates how CONSOL can be applied to other social networks. We select two OSNs for the case studies, i.e., Facebook and LinkedIn. These two OSN services have different business models. However, both services are widely used.

6.1 Case I: Application to Facebook

Facebook is the most popular OSN to date. It is a general purpose OSN. It is designed for audience from every walk of life and contains materials about everything. According to Facebook, there are 1.13 billion daily active users as of June 30, 2016 and 1.03 billion of them are mobile users (Facebook, 2016). Facebook users come from different fields, professions, and countries and have different interests. Almost 84.5% of the total users on Facebook are outside US and Canada (Facebook, 2016). The numbers speak itself for the popularity of Facebook. Facebook is estimated worth $245 billion according to CNN (Monica, 2015).

Users can create profiles with their personal information and post photos, posts, emotions, locations, videos, etc. on Facebook. Other users can follow, add as friend, and
communicate with them. People can also create a group, which can be joined by others with similar interests or create a closed group which can be joined only upon request and acceptance. Short URLs are also very popular on Facebook. Many people share web content with their friends, or as group posts. These short URLs could be malicious and can spread malwares to users’ computers.

CONSOL utilizes social network content, context, and social features to classify short URLs. Our objective here is to demonstrate that Facebook contains the features that can be used by CONSOL for short URL classification.

### 6.1.1 Social Features

CONSOL uses 4 social features for short URL classification, i.e., Friends_count, Followers_count, Favorites_count, and Statuses_count. Facebook includes similar social features which can be leveraged by CONSOL. Friends_count is directly available on Facebook. Friends_count keeps track of how many friends a user has. It is listed under “All Friends” in user’s profile. Followers feature is also available on Facebook whereby followers can view the content posted by the user as public. Although content posted by the user can be viewed without even following the user a user needs to search for the user to see the posts. These posts are often buried under the posts by friends. Followers_count is listed under the “Followers” tab under “friends” page. Favorites_count in Facebook follows the approach of likes. It is related to the post and can be directly observed on the posts. However, to obtain how much likes a user has made, one will have to search through the user’s activity log. This requires extra processing but it can be achieved. Statuses_count is available on Facebook too. Facebook provides the past 8 years Statuses_count information.
6.1.2 Content Features

Users generate content including photos, videos, status, etc. on Facebook. We are interested in the content including short URLs in this research. Posts that contain short URLs usually include keywords describing the content of the URLs. From these keywords, we can extract content features. CONSOL particularly uses nine content features, i.e., length_of_tweet, bag-of-words (check, watch, sex, klik, click, and interested), number of hashtags, and number of URLs.

Length_of_tweet can be calculated using the length of the post on Facebook. Bag-of-words can be looked upon the posts for keywords that fall in our bag-of-words. Hashtags are also used on Facebook. Hashtags represent the topics that are in trending or important. Hashtags can be found in posts and the number of hashtags can be counted. Number_of_urls can be obtained from a post as well.

6.1.3 Context Features

Context is the background information that a post is based on. Context could be happiness, sadness, excitement, announcements, advertisements, news, etc. CONSOL uses 3 context features, i.e., in_reply_to_status, in_reply_to_user, and relevance.

Unlike Twitter, replies are directly commented under the same post. In_reply_to_status can be retrieved using the post ID number. In_reply_to_user can be derived from the post if the post contains tags. Posts are often tagged to a particular user. It can be some funny messages, announcements, or hatred messages sometimes. Relevance can be obtained from the trending field provided on Facebook. Trends can be looked based on top trends on Facebook such as politics, science & technology, sports, and entertainment.
6.1.4 Facebook and CONSOL

As discussed in Table 4-6, CONSOL relies on 16 social network features to detect malicious short URLs. From our discussion, we are able to map all 16 social network features to corresponding information on Facebook. We are confident that CONSOL can be effectively applied to Facebook.

6.2 Case II: Application to LinkedIn

LinkedIn is one of the most popular professional OSNs started in 2003. People from different professions use LinkedIn to form professional networks, connect with their peers, follow organizations of their interests, form/join group, and post messages and comments. LinkedIn is also a platform where potential employers look for candidates for their job openings. Users can search jobs and apply for jobs using LinkedIn service. According to LinkedIn, there are more than 400 million members from more than 200 countries (Linkedin, 2015). It generated revenue worth $862 million in 2015.

Similar to Facebook, people create their profiles on LinkedIn with their personal information including education, work experience, skills, publications, certifications, etc. Users can write posts, like, and comments on posts, etc. The content posted on LinkedIn often includes long and short URLs. CONSOL can also be applied on LinkedIn to detect malicious short URLs.

6.2.1 Social Features

CONSOL uses four social features for short URL classification, i.e., Favorites_count, Friends_count, Followers_count, and Statuses_count. Favorites_count can be achieved via searching through user activities on LinkedIn. However, the duration of recent activities that can be viewed is changed constantly. The default is only 14 days.
Friends_count is equivalent to connections on LinkedIn. Connection forms on LinkedIn once a user accepts a request from another user. Followers_count can be retrieved directly from LinkedIn. There are many ways to get this number. One way to find the number of followers is to go to the Followers tab under activities. Statuses_count can also be retrieved similarly. However, it also has the limitation of 14 days.

6.2.2 Content Features

CONSOL relies on nine content features for operation, i.e., length_of_tweet, bag-of-words (check, watch, sex, klik, click, and interested), number of hashtags, number of URLs. Length_of_tweet can be achieved directly by counting the words in a post. The bag-of-words can be created based on the content of the posts. Hashtags are not popular on LinkedIn. However, it is still applicable since it depends upon users generating posts. Number_of_URLs can be directly retrieved from a post. However, the number of URLs found is one in most cases.

6.2.3 Context Features

CONSOL relies upon three context features including in_reply_to_user, in_reply_to_status, and relevance. In_reply_to_user can be retrieved similarly as Facebook. It usually is under comments and contains tags. In_reply_to_status is the same as comments on Facebook. Relevance is a complicated attribute to acquire on LinkedIn since all the posts are “relevant”. Thus, we suggest using ‘1’ for relevance for all the posts on LinkedIn.
6.2.4 LinkedIn and CONSOL

From discussion above, we are able to map all 16 social network features to LinkedIn with some minor challenges. We are confident that CONSOL can be applied to LinkedIn for malicious short URL detection with a little bit more extra efforts.

6.3 Summary

CONSOL utilizes features available on online social networks to classify malicious short URLs. Online social networks may operate in different business models. However, they all follow the same principles. The information on OSNs can be divided into content features, context features, and social features. These features can be used by CONSOL for malicious URL detection. In our two case studies, we show that we are able to map the information on Facebook and LinkedIn to the features used in the CONSOL. Thus, we are confident that CONSOL can be applied to other online social networks as well.
Chapter 7 Conclusion and Future Work

Online social network is a place where people are willing to share information. People share personal information, photos, news, announcements, etc. The shared information often includes URLs. Due to the complicity of long URL, short URLs become popular. Short URLs are alias of long URLs and ease the process of URL sharing. However, there are also inherent risks using short URLs. Phishing, malware, spams, and scams are common threats on OSNs. Short URLs have been used heavily to carry on these attacks. Existing malicious URL filters work well with long URLs. However, they do not work well with short URLs. OSNs fail to stop malicious short URLs from being propagated via their services. This led to the topic of the dissertation.

In this dissertation, we design and develop a short URL classifier, CONSOL, using the features available on OSNs. CONSOL utilizes content features, context features, and social features to identify malicious short URLs. It uses backlist services such as VirusTotal and PhishTank to label the data and train the classifier. However, the classifier will operate on its own using the features from online social networks without using any third party services once trained. Our testing shows that the classifier achieves high accuracy in identifying malicious short URLs. The comparisons of the CONSOL with the existing solution and Google Safe Browsing also show that the classifier is promising in the real world. Our case studies further indicate that other online social networks can also adopt the CONSOL with very minimal to no difficulty. The research in the dissertation follows the design science principles. Data collection is facilitated with python scripts using Twitter streaming APIs and Bitly APIs. We also leverage blacklist services from VirusTotal and PhishTank.
7.1 Research Questions

This dissertation answers the following research questions.

1. **How can we leverage the information available on OSNs to detect malicious short URLs?**

   The motivation of this study is to design and develop a classifier for short URLs using the information available solely from online social networks. Many existing solutions leverage information from third party service providers. However, it is not always possible to retrieve the desired information from a third service provider given that there are hundreds of them on the Internet. On the other hand, there is huge amount of information available on OSNs. The information is free and it is always accessible. In this research, we divide the information from OSNs into three logical categories, i.e., content features, context features, and social features. Using these features, we are able to develop a classifier, CONSOL, for short URL classification. Our testing and comparisons show that CONSOL achieves high accuracy in identifying malicious short URLs. This shows our assertion “that detecting malicious short URLs is practical using information from OSNs only since malicious short URLs propagate through OSNs leveraging social relationships” is correct.

2. **What are the most important features for malicious short URL detection?**

   Our classifier utilizes 16 features as shown in Table 4-6 Selected Features for Random Forest with Filter Method for malicious URL identification. These 16 features are further divided into three logical categories, i.e., content features, context features, and social features. To find out which category is the most important for short URL
classification, we performed experimental analysis on the three categories. We found that social features are the most significant feature set among all the three categories. Social features alone can predict short URLs with up to 91.47% accuracy (Table 5-2). Content features are also important for identifying malicious short URLs. Content features alone can predict short URLs with up to 70.58% accuracy. Among all the three categories, context features are the weakest. However, each category of features is unique in nature and contributes towards the overall performance of the classifier.

3. **How can we develop an effective mechanism to detect malicious short URLs with high accuracy on OSNs?**

In this research, our objective is to develop an effective classifier that can be used to detect malicious short URLs on OSNs. Thus, accuracy is important for the design of the classifier. As we demonstrated in Chapter 4, we are able to develop short URL classifier using features from online social networks only. The classifier achieves 94.5% accuracy using the Random Forest algorithm. The comparison of the classifier with the existing solution and the Google Safe Browsing all indicates the effectiveness of the classifier.

4. **Can the classifier be easily adapted/used in other social networks?**

Our goal is to build a short URL classifier that works for online social networks in general. In this research, we demonstrate the effectiveness of the classifier using the data collected from Twitter. There are many OSNs on the Internet. They may operate on different business models. However, the information on OSNs can all be classified into three categories such as content features, context features, and social features. Thus,
adopting the CONSOL in other OSNs is practicable. This has been demonstrated in our two case studies using Facebook and LinkedIn in Chapter 6.

7.2 Implications

This study, to the best of our knowledge, is the first work to demonstrate that the social network features alone can be used to classify short URLs. The implications of this research are divided into twofold, i.e., theoretical implications and practical implications.

7.2.1 Theoretical Implications

This research implements the well-recognized principles and practices, and applies them to malicious short URL detection context. The research elucidates the importance of information available on OSNs in the security and particularly in the detection of malicious short URLs. From the theoretical perspective, the research limits the scope of contemporary URL classification study to online social networks. Using experimental evidence, the research demonstrates that information from OSNs alone can sufficiently and effectively classify short URLs, adding to the knowledge base.

The research also studies the impact of different categories of features in classifying short URLs and shows that social features are the biggest contributor to the classification of short URLs. Furthermore, the study also provides a reference, which may assist future researchers to study security issues in online social networks.

7.2.2 Practical Implications

Without an artifact solving a real world Information Systems problem, a design science research is nothing. With better understanding of the malicious short URL domain, organizations can deploy effective security mechanism in their information
systems. The findings of this research help security practitioners to effectively deploy security mechanisms to prevent malicious short URLs from being propagated on OSNs. The key practical implication of the research is listed as below.

We developed CONSOL; an OSN based short URL classifier, which can effectively detect malicious short URLs on OSNs. We achieved an accuracy of up to 94.5% and a precision of 0.945.

7.3 Strengths and Limitations

This research is the first in-depth study of short URL classification using information available on OSNs. The study shows that short URLs could be effectively classified using the information from online social networks. The classifier does not depend on any third parties during operations.

In general, the study identifies the key feature set used in the classifier from the existing literature and our research findings. The study provides insight into the importance of different feature sets and their predictive capacity to classify short URLs. Finally, the study develops a short URL classifier, CONSOL, which achieves 94.5% accuracy and outperforms any existing solutions in the literature. We are confident and proved analytically that the CONSOL can also be adopted by other OSNs for short URL classification.

One of the limitations in this study is that CONSOL uses third party information for training. After the training, CONSOL does not depend on any information from third parties other than the OSN.
7.4 Future Research

A single study cannot extract all possible features from OSNs. This is especially true when the objective is to develop a short URL classifier that can work in many platforms with slight or no modification. Specifically, the features differ in different online social networks. Hence, a large-scale exploratory research to study all directly observed and derived feature sets can be very helpful in the short URL classification.

Bagging of different algorithms boosts the performance of the classifier and has been studied well in the machine learning community. Bagging provides a mechanism to leverage the performance of different machine learning algorithms and get the best of each and stack them into a single classifier. How to implement bagging method to improve the performance of the classifier will also be in our future research.

7.5 Summary

Securing social networking services is an important task as well as a big challenge. Phishing, malware, spams, and scams have plagued the OSNs. This dissertation addresses the short URL classification issue using the information available on online social networks. We identify the features which can be leveraged in such a classifier and show that in fact we are able effectively classify short URLs using the features obtained from OSNs. We also evaluate the predictive power of different feature sets and find that social features are the biggest contributors to the prediction of malicious short URLs.

In conclusion, this study points out a new direction for short URL classification by using the information obtained from OSNs. It helps future researchers to focus on social networking services rather than third party services. Based on the findings of this
dissertation, we believe that there are more prospective features on OSNs, which could be leveraged by the classifier to improve its accuracy. We hope the findings of this research will benefit the information security community and OSN security specifically.
Appendices

Appendix I Tweet

Tweeter provides streaming APIs to access Tweet data. The individual message/tweet streamed by the APIs are JSON encoded. A sample of the tweet is enclosed below. It includes fields such as user, tweet, URLs, retweet, and user mentions. Details of the tweet structure can be found at https://dev.twitter.com/overview/api/tweets.

```json
{
  "created_at": "Thu Feb 12 17:26:27 +0000 2015",
  "id": 565924909025881234,
  "id_str": "565924909025882112",
  "text": "RT @ElNacionalWeb: Urgente: Reportan 3 estudiantes heridos en San",
  "source": "u003c/a href="http://www.twitter.com/u003eTwitter for Windows Phone\u003c/a/u003e",
  "truncated": false,
  "in_reply_to_status_id": null,
  "user": {
    "id": 123456789, ## Edited to preserve user privacy
    "id_str": "123456789",
    "name": "Leo u0950",
    "screen_name": "ABC_XYZ", ## Edited to preserve user privacy
    "location": "",
    "url": null,
    "description": "Hispano-Europeo/FuturoOdontologo/Basquetbolista/Caraquista y dedicado a mis estudios y familia",
    "protected": false,
    "verified": false,
    "followers_count": 194,
    "friends_count": 188,
    "listed_count": 0,
    "favourites_count": 497,
    "statuses_count": 5153,
    "created_at": "Fri Jan 14 15:28:32 +0000 2011",
    "utc_offset": -36000,
    "time_zone": "Hawaii",
    "retweeted_status": {
      "created_at": "Thu Feb 12 17:07:11 +0000 2015",
      "id": 565920062734106624,
      "id_str": "565920062734106624",
      "text": "Urgente: Reportan 3 estudiantes heridos en San Crist\u00edbal este #12F http:\/\slash\slash t.co/PDYIYVPhQO (V\u00eda: @elproprioweb)",
      "source": "u003c/a href="http://www.hootsuite.com" rel="nofollow\u003eHootsuite\u003c/a/u003e",
      "truncated": false,
      "in_reply_to_status_id": null,
      "user":
    }
  }
}

Crist\u00f3bal este #12F http:\/\slash\slash t.co/PDYIYVPhQO (V\u00eda: @elproprioweb),
"source": "u003c/a href="http://www.hootsuite.com" rel="nofollow\u003eHootsuite\u003c/a/u003e",
"truncated": false,
"in_reply_to_status_id": null,
"user":
```
user privacy

noticias más recientes de Venezuela y el mundo, Deportes,

2008",

"geo":null,
"coordinates":null,
"place":null,
"contributors":null,
"retweet_count":238,
"favorite_count":10,
"entities":
{
"hashtags":[
{
"text":"12F",
"indices":[62,66]}
],
"trends":[]
,"urls":[]
},
"url":"http:\/t.co\/PDYIYVPhQO",
"expanded_url":"http:\/\/Bitly.com\!/1CZF3KP","display_url":"Bitly.com\!/1CZF3KP",
"indices":[67,89]
],
"user_mentions":[]
,"screen_name":"TYR", ## Edited to preserve user privacy
"name":"TYR", ## Edited to preserve user privacy
"id":456789012, ## Edited to preserve user privacy
"id_str":"456789012", ## Edited to preserve user privacy
"indices":[96,108]}},
"symbols":[]

"favorited":false,
"retweeted":false,
"possibly_sensitive":false,
"filter_level":"low"

"retweet_count":0,
"favorite_count":0,
"entities":
{
"hashtags":[]
{
"text":"12F",
"indices":[81,85]},

"trends":[]

"urls":{

"url":"http:\/\/t.co\/PDYIYVPhQO",
"expanded_url":"http:\/\/Bitly.com\/1CZF3KP",
"display_url":"Bitly.com\/1CZF3KP",
"indices":[86,108]},

"user_mentions":{

{
"screen_name":"MNO", ## Edited to preserve user privacy
"name":"MNO", ## Edited to preserve user privacy
"id":15000000, ## Edited to preserve user privacy
"id_str":"15000000", ## Edited to preserve user privacy
"indices":[3,17]}
},

{
"screen_name":"PQR", ## Edited to preserve user privacy
"name":"PQR", ## Edited to preserve user privacy
"id":000000000, ## Edited to preserve user privacy
"id_str":"000000000", ## Edited to preserve user privacy
"indices":[]
}
"indices": [115, 127],
"symbols": []
},
"favorited": false,
"retweeted": false,
"possibly_sensitive": false,
"filter_level": "low",
"lang": "es",
"timestamp_ms": "1423761987345"
Appendix II Sample Dataset

In this research, we create different datasets as shown in Table 4-4 Datasets and Description for our testing and evaluation purpose. A sample of the dataset is enclosed below. The dataset includes 16 fields as shown in Table 4-6 Selected Features for Random Forest with Filter Method. These 16 fields fall into three categories, i.e., context features, context features, and social features. These 16 fields are derived from raw tweet data collected from Twitter using its streaming APIs. The last column in the table indicates the malignity of the tweet, 1 for malicious and 0 for benign.

<table>
<thead>
<tr>
<th>length_of_tweet</th>
<th>contains_str</th>
<th>contains_str</th>
<th>contains_str</th>
<th>contains_str</th>
<th>contains_str</th>
<th>contains_str</th>
<th>contains_str</th>
<th>relevance</th>
<th>in_reply_to</th>
<th>in_reply_to</th>
<th>followers</th>
<th>friends</th>
<th>favorit</th>
<th>contains_str</th>
<th>contains_str</th>
<th>contains_str</th>
<th>number</th>
<th>number</th>
<th>url</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>104</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>128</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>133</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>138</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>140</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>139</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>119</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>108</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>117</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>113</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>134</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>89</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>133</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>134</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>140</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>140</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>88</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>108</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>117</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>113</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>134</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>89</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
References


Computational approaches to Analysing Weblogs. Stanford, California.


https://www.phishtank.com/faq.php


