Feature Extraction And Detection of Malicious URLs Using Deep Learning Approach

Rajni Kushwaha

Features Extraction And Detection of Malicious URLs Using Deep Learning Approach

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Rajni Kushwaha

[Roll No: CS-1707]

under the guidance of

Dr. K.S Ray

Professor

Electronics and Communication Sciences Unit

Natural Computing Lab

Indian Statistical Institute
Kolkata-700108, India

July 2019

To my guide and my family

CERTIFICATE

This is to certify that the dissertation entitled **"Features Extraction And Detection of Malicious URLs Using Deep Learning Approach"** submitted by **Rajni Kushwaha** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by her under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Kumar Sankar Ray
Professor,
Electronics and Communication Sciences Unit Natural Computing Lab,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Prof.Kumar Sankar Ray*, Electronics and Communication Sciences Unit Natural Computing Lab, Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement.I would like to thanks to him, for giving me the opportunity to carry out this project. He has taught me how to do good research, and motivated me with great insights and innovative ideas.

I would also like to thank *Siladittya Manna*, for his invaluable suggestions and discussions., who is doing project in the area of deep learning under the supervision of Professor K. S. Ray.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my dissertation work.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Rajni Kushwaha
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

Phishing Attack is one of the cyber bullying activity over the internet. Most of the phishing websites try to look similar to legitimate websites, their web content and URL features memic the legitimate URL. Due to emerging new techniques, detecting and analyzing these malicious URL is very costly due to their complexities. Traditionally, black and white listing is used for detection, but these technique was not good for real time.To address this, recent years have witnessed several efforts to perform Malicious URL Detection using Machine Learning. The most popular and scalable approaches use lexical properties of the URL string by extracting Bag-of-words like features, followed by applying machine learning models such as SVMs, Randon Forest etc. Various machine learning and deep learning techniques are used to improve generalization of malicious URLs.These approaches suffer from several limitations: (i) Inability to effectively capture semantic meaning and sequential patterns in URL strings; (ii) Requiring substantial manual feature engineering; and (iii) Inability to handle unseen features and generalize to test data.

To address these Limitation, In this dissertation work, we are focused to built the real time and language independent phishing detection model by analyzing the anatomy of the URLs using deep learning techniques. To achieve this, we firstly try to find static and dynamic features manually using some previous work. After getting the featured valued data set, we tried to find the lexical features of Url using CNN which has both characters and words of the URL String to learn the URL embedding. After that we merge features which we manually selected and features learned from CNN and applied on Bi-LSTM Model to keeps the sequence information of URL. A hybrid model of CNN (convolution neural network model) and Bi-directional LSTM(Long Short Term Memory) are to achieve the goal. Our model analyze the URL without accessing the web content of websites. It eliminates the time latency.

Keywords: *Malicious URL, Feature Extraction,CNN, LSTM,TCN, Lexical, Mutliview Features Selection.*

Contents

Chapter 1

Introduction

1.1 Problem Statement and Research Motivation

In past few years a lots of research had been done to prevent malicious attack in order to prevent to internet crime. URL globally addressed the documents and other resources on the world wide web. Malicious URL is the serious threat to cyber security. Malicious web sites host's broadcast unsolicited content over the internet and unsuspecting users can visit such websites and become a victim of these types of attacks and had some information loss.

In Feb 2019, Google statistics shows 1,300,000 malicious URL block per day (See in fig). Existence of these phishing URL leads to great threats to the security of web application. Many researcher and practitioner have been working on designing an effective and real time system to detect malicious URLs. The most common method to detect is using black and white list method, but this method is efficient for real time, because it is almost impossible to maintain an exhaustive list of malicious URLs, since new URLs are generated everyday. In practice these technique is very fast due to simple query processing and easy to implement. Attackers tried to various approaches to evade the blacklist and modified the malicious URL which look similar to legitimate via obfuscation. This technique has several limitations, and bypassing them seems almost trivial, particularly because blacklists are ineffective to predict the fresh URLs. Therefore, an immediate issue is how to solve or design an automated tool to rapidly and precisely differentiate emerging malicious websites from URLs and other ordinary web pages.Identification of kinds of attacks is helpful as understanding of the nature of a prospective threat enables us to take a correct response as well as a relevant and efficient threat countermeasure.

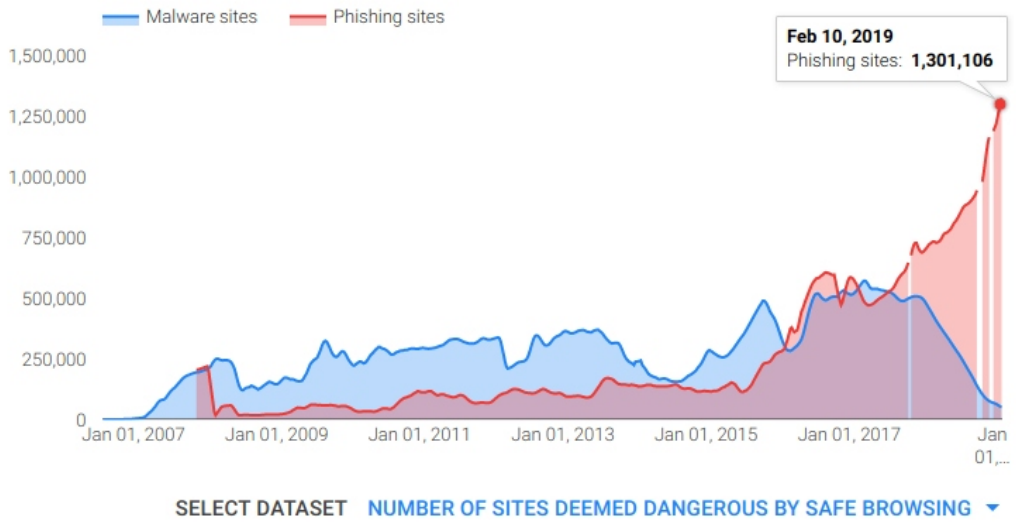


Fig 1.1

To address these problem, we design a system that take URL string as input and applies CNNs to both characters and words in the URL. For Character-level CNNs we first identify unique characters in the training corpus, and represent each character as a vector. Using this, the entire URL (a sequence of characters) is converted to a matrix representation, on which convolution can be applied. Character CNNs identify important information from certain groups of characters appearing together which could be indicative of maliciousness. For Word-level CNNs, we first identify unique words in the training corpus, de- limited by special characters. Using a word-embedding matrix, we obtain a matrix representation of the URL (which in this context, is a sequence of words). Following this convolution can be applied. Word-level CNNs identify useful patterns from certain groups of words appearing together. However, using word-embeddings faces some challenges: (i) it cannot obtain embeddings for new words at test time; and (ii) too many unique words (specifically in mali- cious URL detection) - resulting in memory constraints while learn- ing word embeddings. To alleviate these, we propose advanced word-embeddings where the word embedding is learned using the character-level information of each word. This also helps recognize subword level information. Both Character-level and Word-level CNNs are jointly optimized to learn the Bi-lsprtm model that preserve the sequence information of URL and classify using softmax function.

Our goal is to classify a given URL as malicious or not. We do this by formulating the problem as a binary classification task. Consider a set of 'T' URLs, $\{(u_1, y_1), ..., (u_T, y_T)\}$, where u_t for t = 1, . . . ,T represents a URL, and $y_t \in \{0, 1\}$ denotes the label of the URL, with $y_t = 1$ being a malicious URL, and $y_t =$

0 being a benign URL. The first step in the classification procedure is to obtain a feature representation $u_t x_t$ where $x_t \in R^n$ is the ndimensional feature vector representing URL u_t . The next step is to learn a prediction function $f : R^n R$ which is the score predicting the class assignment for a URL instance x. The prediction made by the function is denoted as $\hat{y_t} = sign(f(x_t))$. The aim is to learn a function that can minimize the total number of mistakes ($\sum_{t=1}^{T} I_{\hat{y_t} \neq y_t}$) in the entire dataset. This is often achieved by minimizing a loss function. Many types of loss functions can be used, which may also include a regularizer term. For Deep Learning, the function f is represented as a Deep Neural Network, such as a any standard network or hybrid model of networks.

1.2 Dissertation Contributions

Our first contribution is to design a Real time, Language independent phishing detection model. Since our model trained on real time data, and it understand the anatomy of malicious Url, So it classified the new unknown test url. Since we are using the URL of website, we does not use the web-content of the url, So it is language independent model. It also eliminates the time latency, because it does not go to analyze the web content of url so time latency is removed.

The second contribution is understanding the "natural" evolution of malicious websites over the time. Since our model trained on real time data of url, so our model understand the strategy of attackers used to modified the Url over time. Eg:- Initally phishing Urls is long in length to hide the suspicious information in url, but researchers built a model that classify long length url as phishing, then attacker become smart and tried to trick the user and used some famous domain name in url and make a length of average size, user generally ignore it and become a victim, but then research used host based feature and detect phishing url. Since our model learn from real time data, So they understand the strategy used by attackers in phishing URL over time.

Third contribution is our model learn the semantic and sequential patterns in which the characters and words appear in the URL. Most of the researcher focus to built a model that extract lexical features from URL, but forget to keep the sequence information of URL, Since URL followed some standard sequence.

Chapter 2

Previous Work

Phishing is an attempt to steal a user‘s personal information typically through a fraudulent email or website. We conducted a study on phishing sites, which are either fake sites that are designed to appear similar to legitimate sites or sites that simply have phishing related behaviors. Almost all phishing sites include the functionality in which users enter sensitive information.These sites can include links to connect to other phishing sites and malicious code that contaminates a user‘s computer.

2.1 Classification based approach

The most common method techniques of phishing detection is using black and white list and heuristic-based approaches. The blacklist-based approach maintains a database list of addresses (URLs) of sites that are classified as malicious, but this method is inefficient for real time, because it is almost impossible to maintain an exhaustive list of malicious URLs, since new URLs are generated everyday. In practice these technique is very fast due to simple query processing and easy to implement. Attackers tried to various approaches to evade the blacklist and modified the malicious URL which look similar to legitimate. This technique has several limitations, and by passing them seems almost trivial, particularly because blacklists are ineffective to predict the fresh URLs. The blacklist based approach has the advantages of easy implementation and a low false positive rate, however, it cannot detect phishing sites that are not listed in the database, including temporarily sites.

The heuristic-based approach analyzes phishing site features and generates a classifier using those features. When a user requests a web page, the classifier determines whether that page is a phishing site. This approach can detect new phishing sites

and temporary phishing sites because it extracts features from the requested web page. Nevertheless, it has the disadvantage of being difficult to implement, moreover generating a classifier is time intensive process.

The work by Garera et al[9] use logistic regression over 18 hand-selected features to classify phishing URLs. The features include the presence red flag key words in the URL, features based on Google's Page Rank and Google's Web page quality guidelines. They achieve a classification accuracy of 97.3 % over a set of 2,500 URLs. But the problem with this approach is that we manually select some features and weighted equally to all features. But we may missed some important features and cannot correlated the presence of features in specific sequence. Since Urls structure followed some sequence pattern, so we must concern about it.

McGrath and Gupta do not construct a classifier but nevertheless perform a comparative analysis of phishing and non-phishing URLs. For examples, they compare non-phishing URLs drawn from the DMOZ Open Directory Project to phishing URLs from PhishTank and a non-public source. The features they analyze include IP addresses, WHOIS thin records (containing date and registrar-provided information only), geographic information, and lexical features of the URL (length, character distribution, and presence of predefined brand names etc). We build on their work by incorporating similar sources and features into our approach.

CANTINA classifies phishing URLs by thresholding a weighted sum of 8 features (4 content related, 3 lexical, and 1 WHOIS related). Among the lexical features, it looks at dots in the URL, whether certain characters are present, and whether the URL contains an IP address. The WHOIS related feature like:- examines is the age of the domain, DNS record etc. We use similar features in our approach, but entirely different models of classification.

2.2 Machine Learning Based Approach

A study has proposed a method using machine learning to detect malicious URLs of all the popular attack types like spam, phishing, malware etc. and to identify the nature of attack a malicious URL attempts to launch. They have used features like lexical, link popularity, Webpage content, DNS, DNS fluxiness and network traffic. They have collected real life data from various sources like benign URLs from DMOZ Open Directory Project, Yahoo!s directory, Spam URLs from jwSpamSpy, Web spam dataset, Phishing URLs from PhishTank and Malware URLs from DNS-BH. They have used three machine learning algorithms like the Support Vector Machine (SVM) to detect malicious URLs, RAkEL and ML-kNN learning algorithms for multi-label

classification problem to identify attack type. They have evaluated their method on 40,000 benign URLs and 32,000 malicious URLs and achieved the accuracy of 98 % in detection of malicious URLs and 93identification of attack types.

An approach based on automated URL classification, using statistical methods to discover the lexical and host-based properties of malicious Web site URLs is proposed by [4]. They have extracted the Lexical features and Host-based features. The host-based features include IP address properties, WHOIS properties, domain name properties and geographic properties. They have used machine learning algorithms like Naive Bayes, Support Vector Machine (SVM) and Logistic Regression for evaluation. According to them, the resulting classifiers obtain 95.99% accuracy, detecting large numbers of malicious Web sites from URLs, with only modest false positives.

A machine learning based approach is proposed by [7] to detect phishing Web pages. They have used many novel content based features and applied cutting-edge machine learning techniques such as 6 batch learning algorithms, Random Forests, Support Vector Machines (SVM) with rbf linear kernels, Naive Bayes, C4.5, Logistic Regression (LR) and a set of 5 online learning algorithms: updatable version of Naive Bayes (NB-U), updatable version of LogitBoost (LB-U), Perceptron, Passive- Aggressive (PA) and Confidence-Weighted (CW) algorithms. They have used 179 Web page features such as lexical based features, keyword based features, search engine based feature and reputation based features to demonstrate their approach. To conduct all the experiments, they used WEKA and CW libraries. The experimental results show that their proposed approach can detect phishing Webpages with an accuracy of 96.9%, false positive rate of as low as 3.00% and false negative rate of 3.06%.

2.3 Deep Learning Based Approach

Deep Learning or Representation Learning has received increasing interest in recent years owing to their success in several applications. The core idea is to automatically learn the feature representation from raw or unstructured data, in an end-to-end manner without using any hand designed features. Following this principle, we aim to use Deep Learning for Malicious URL Detection, in order to directly learn representation of the raw URL string, without using any hand designed expert features.

Since we aim to train Deep Networks over lexical features, a closely related area is Deep Learning for Natural Language Pro- cessing (NLP). Deep learning methods have found success in many NLP tasks: text classification, machine translation, question answering, etc. Recurrent neural networks (e.g. LSTM) have been widely used

due to their ability in capturing sequential information. However, the problems of exploding and vanishing gradients is magnified for them, making them difficult to train. Recently, Convolutional Neural Networks have become excellent alternatives to LSTMs, in particular showing promising performance for text classification using Word-level CNNs and Character-level CNNs.

There have been very limited attempts at using Deep Learning for Malicious URL Detection. We recently noticed a work parallel to ours in paper(URLNet) that attempted to use Character-level CNNs for this task. However, they ignored several types of structural information that could be captured by words in the URLs. In contrast to their work, (i) we consider both word-level and character-level information; (ii) through extensive analysis we show the importance of word-level information in capturing longer temporal patterns; (iii) we develop character-level word embedding for effectively utilizing word-level information - in particular handling the presence of too many unique words, and obtaining embeddings for unseen words at test time (iv) After obtaining the features vector from CNN, we trained our model on B-LSTM model to keep the sequence informtion of URL . Our model captures the structural information available in the URL String through both character and word-level information.

Chapter 3

Data Collection

We now describe the methodology underlying our study, including data collection, data prepossessing, evaluation metrics and data analysis methods. For implementation, we are using two sources to collect data:

- For malicious URLs we are collecting data from (https://www.phishtank.com/)

- For benign URLs we are collecting data from (https://www.alexa.com/)

In order to facilitate cross-layer analysis and detection, we need an automated system to collect both the application-layer website contents and the corresponding network-layer traffic. The architecture of our automated data collection system is depicted in Figure 2.1. At a high level, the data collection system is centered on a crawler. The crawler takes a list of URLs as input, automatically fetches the website contents by launching HTTP requests and tracks the redirects that are identified from the website contents (elaborated below). The crawler also uses the URLs, including the input URL and the detected redirection URLs, to query the DNS, Whois, and Geographic services. This collects information about the registration dates of websites and the geographic locations of the URL owners/registrants.

The input URLs may consist of malicious and benign websites. A URL is malicious if the corresponding website content is malicious or any of its redirects leads to a URL that corresponds to malicious content; otherwise, it is benign. In this, the terms malicious URLs and malicious websites are used interchangeably.

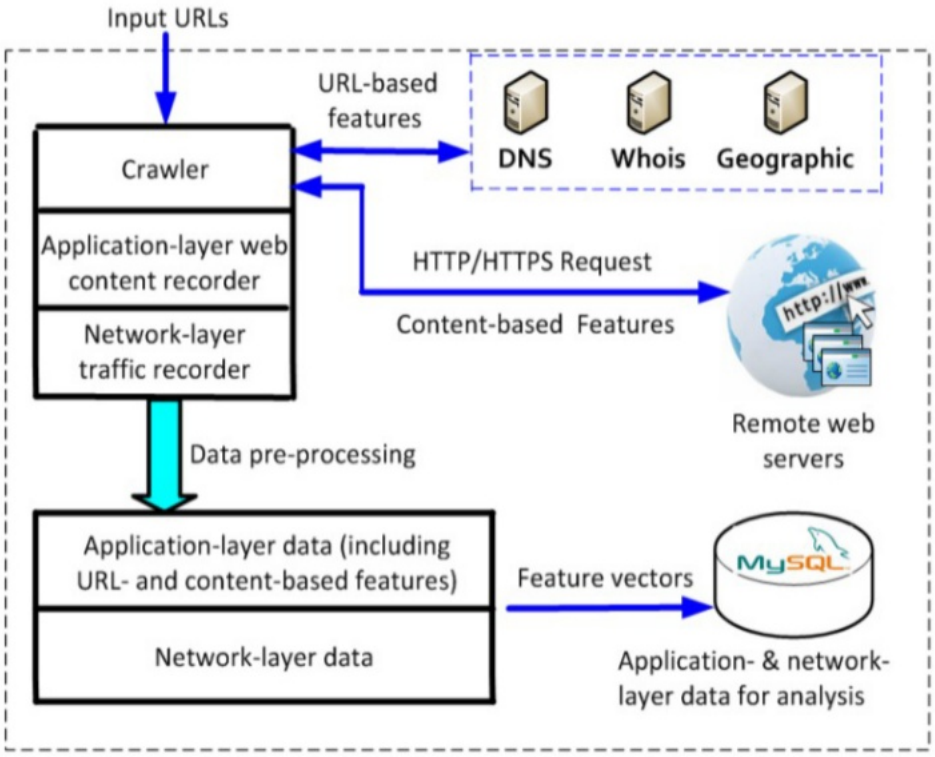


Figure 2.1: Data collection system architecture.

Chapter 4

Features Engineering

4.1 Feature based on the URL lexical information

URL dataset is not as simple as text data. So we need to do some feature engineering (use domain knowledge of the data) to extract some feature, we do it manually by using some novel approach of researcher as well as used deep learning architecture to learn features. Manually selected features are given below:

URL Length : Long URL to Hide the Suspicious Part. If the length of the URL is greater than or equal 54 characters then the URL classified as phishing.

Presence of @ Symbol in Url : Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol.

Presence of Redirection Symbol : The existence of "//" within the URL path means that the user will be redirected to another website. An example of such URL's is: "http://www.legitimate.com//http://www.phishing.com". We examine the location where the "//" appears. We find that if the URL starts with "HTTP", that means the "//" should appear in the sixth position. However, if the URL employs "HTTPS" then the "//" should appear in seventh position..

Prefix or Suffix Separated by (-) to the Domain:The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage.

For example http://www.Confirme-paypal.com/..

Sub-Domain and Multi Sub-Domains: The legitimate URL link has two dots in the URL since we can ignore typing "www.". If the number of dots is equal to three then the URL is classified as "Suspicious" since it has one sub-domain. However, if the dots are greater than three it is classified as "Phishy" since it will have multiple sub-domains.

Presence of IP Address: If an IP address is used as an alternative of the domain name in the URL, such as "http://125.98.3.123/fake.html", users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link "http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html".

Using URL Shortening Services "TinyURL: URL shortening is a method on the "World Wide Web" in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an "HTTP Redirect" on a domain name that is short, which links to the webpage that has a long URL. For example, the URL "http://portal.hud.ac.uk/" can be shortened to "bit.ly/19DXSk4".

Existence of protocol in domain part : The phishers may add the "HTTPS" token to the domain part of a URL in order to trick users. For example, http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/. domains.

Abnormal URL : This feature can be extracted from WHOIS database. For a legitimate website, identity is typically part of its URL.

Google Index This feature examines whether a website is in Google's index or not. When a site is indexed by Google, it is displayed on search results (Webmaster resources, 2017). Usually, phishing webpages are merely accessible for a short period and as a result, many phishing webpages may not be found on the Google index.

Website Traffic This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as "Phishing". Otherwise, it is classified as "Suspicious".

Rule: IF Website Rank <100,000 $->$ Legitimate Website

If Rank¿100,000 $->$ Suspicious Otherwise Phishing

Domain Registration Length: Based on the fact that a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance. In our dataset, we find that the longest fraudulent domains have been used for one year only.

Age Of Domain : This feature can be extracted from WHOIS database (Whois 2005). Most phishing websites live for a short period of time. By reviewing our dataset, we find that the minimum age of the legitimate domain is 6 months.

Abnormal URL : This feature can be extracted from WHOIS database. For a legitimate website, identity is typically part of its URL.

DNS Record: For phishing websites, either the claimed identity is not recognized by the WHOIS database (Whois 2005) or no records founded for the hostname (Pan and Ding 2006). If the DNS record is empty or not found then the website is classified as "Phishing", otherwise it is classified as "Legitimate".

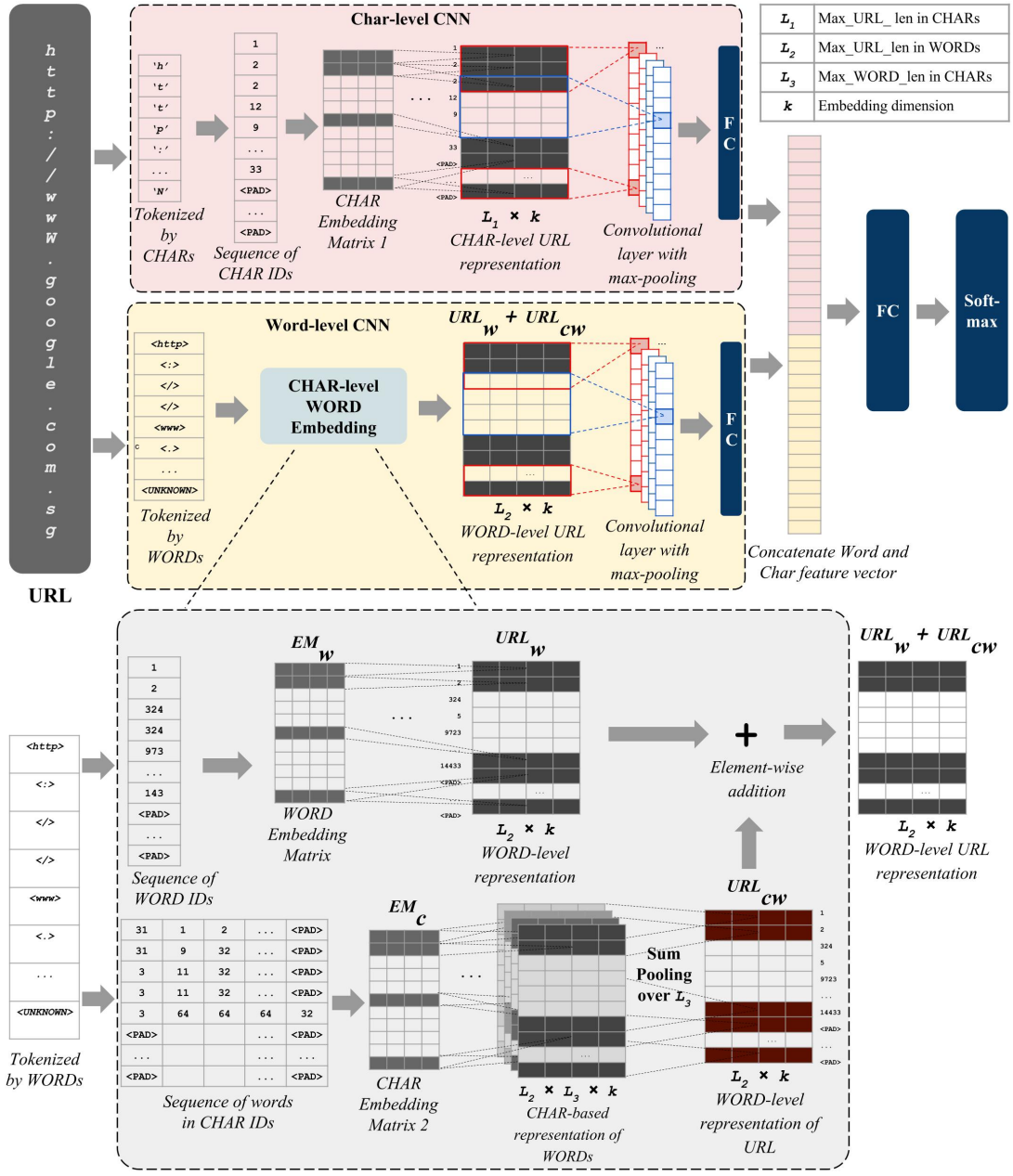
Statistical-Reports Based Feature: Several parties such as PhishTank (PhishTank Stats, 2010-2012), and StopBadware (StopBadware, 2010-2012) formulate numerous statistical reports on phishing websites at every given period of time some are monthly and others are quarterly.

4.2 Feature Extracted From CNN

CNNs could learn useful structural information in text from raw values of word or character embeddings. In Our model, CNNs are used to learn structural information about the URL. Specifically CNNs are applied at both the character-level and word-level.

Character CNNs identify important information from certain groups of characters appearing together which could be indicative of maliciousness. For Word-level CNNs, we first identify unique words in the training corpus, delimited by special characters. Using a word-embedding matrix, we obtain a matrix representation of the URL (which in this context, is a sequence of words). Following this convolution can be applied. Word-level CNNs identify useful patterns from certain groups of words appearing together. However, using word-embeddings faces some challenges: (i) it cannot obtain embeddings for new words at test time; and (ii) too many unique words (specifically

in malicious URL detection) - resulting in memory constraints while learning word embeddings. To alleviate these, we propose advanced word-embeddings where the word embedding is learned using the character-level information of each word. This also helps recognize subword level information. Both Character-level and Word-level CNNs are jointly optimized to learn the prediction model.



After getting features from Character-Level embedding, Word-Level embedding and some manually selected features. We Concatenate all together. And give input to the Bi-LSTM model to keep the sequence information of URL.

Chapter 5

Preliminaries

This section briefly describes some architecture of neural network which we used in our model. CNNs have achieved extraordinary success in Computer Vision tasks. Their designs allowed them to automatically learn the salient features in the images from raw pixel values. Eventually their principles were adopted for Natural Language Processing, where the CNNs could learn useful structural information in text from raw values of word or character embeddings. We use CNNs to learn structural information about the URL. Specifically CNNs are applied at both the character-level and word-level. Next, we used Bi-directional LSTMs to keeps the sequence information of feature vector, and used to classify the URL.

5.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimize through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply. The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data. Common machine learning bench marking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28 28. With this dataset a single neuron

in the first hidden layer will contain 784 weights (28281 where 1 bare in mind that MNIST is normalized to just black and white values), which is manageable for most forms of ANN. If you consider a more substantial coloured image input of 64 64, the number of weights on just a single neuron of the first layer increases substantially to 12, 288. Also take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to classify colour-normalized MNIST digits, then you will understand the drawbacks of using such models.

5.2 Long Short Term Memory(LSTM)

Long Short Term Memory networks are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

5.3 Bi-directional Long Short Term Memory

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. LSTM in its core, preserves information from inputs that has already passed through it using the hidden state.

Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as it is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem. Using bidirectional will run your inputs in two ways, one from past to future and one from future to past and what differs this approach from unidirectional is that in the LSTM that runs backwards you preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

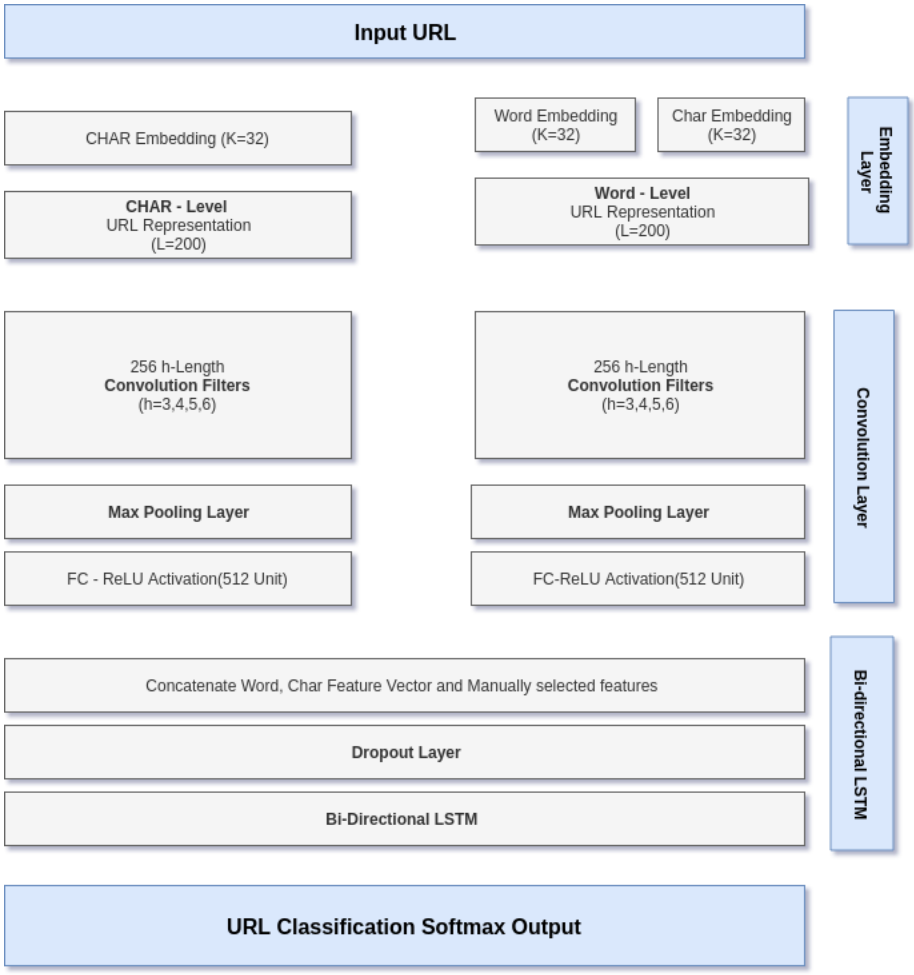
Chapter 6

Model Configuration

The overview of our model can be seen in Figure below. The raw URL input string is processed by 3 branches: a character-level branch, a word-level branch, and some manually seleted features. The character-level branch gives a character-level representation of the URL, while the word-level branch does the same with words. The word-level branch itself is split into word-embedding and character-level word embedding, which are combined to finally give the word-level URL representation, Convolutional operations are applied to both these branches, followed by a fully connected (FC) layer, which is regularized by dropout for both the branches. Then the outputs are concatenated with some features which contain host based information we manually selected. This is followed by Bi-LSTM finally leading to the output classifier. This model is then trained by an adam optimizer using Backpropagation.

A URL u is essentially a sequence of characters or words(delimited by special characters). We aim to obtain its matrix representation $u \rightarrow x \in \mathrm{R}^{L*k}$, such that the instance x comprises a set of contiguous components xi ,i = 1, . . . , L in a sequence, where the component can be a character or a word of the URL. Each such component is represented by an embedding such that $x_i \in R^k$, is a k-dimensional vector.

After finding the feature vector our aim is to find the sequence information of these feature vector.For this we use Bi-directional LSTM model, which takes feature vector x of size R^n as an input to our LSTM model and classified is based on soft max function. Entire model is trained using back propagation algorithm with binary cross entropy loss function.



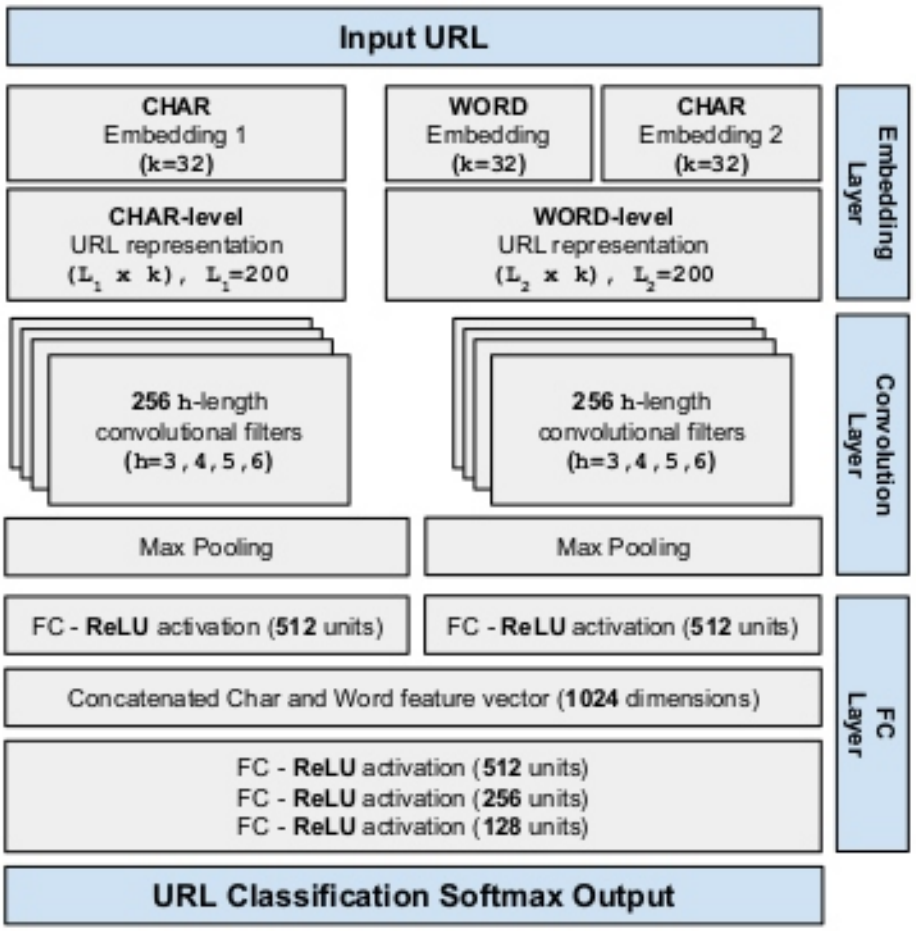
6.1 Combining CNN and LSTM

In this architecture we try to combining CNNs (convolutional neural nets) and LSTMs (long short term memory). After all they're optimized for completely different problem types.

- **CNNs** are good with hierarchical or spatial data and extracting unlabeled features. Those could be images or written characters. CNNs take fixed size inputs and generate fixed size outputs.
- **Bi-LSTMs** are good at temporal or otherwise sequential data. Could be letters or words in a body of text, stock market data, or speech recognition. RNNs can input and output arbitrary lengths of data. LSTMs are a variant of RNNs that allow for controlling how much of prior training data should be remembered, or more appropriately forgotten.

Our requirement are both, CNN model extract the unlabeled features from URL and Bi-LSTM model keeps the sequence information, since every url follwed some standard sequence. There are several emerging models proposed to combine these tools. In our case CNNs and Bi-LSTM have been married as separate layers with the output of the CNN being used as input to the Bi-LSTM.

This is the original URLNet architecture proposed by Hung Le which we used and make some modification on it. We replace the last Fully Connected convolution layer by Bi-directional LSTM model to maintain the sequence information of URL and classified it.



Chapter 7

Model Comparison

7.1 Accuracy and Loss Plot of various traditional Deep Learning Models

These are the hyper parameters used to train the model by CNN, LSTM and Bi-LSTM are:
Optimizer: Adam optimizer
Loss Function: Binary Cross Entropy
Learning Rate: 1e-4
epsilon: 1e-08
No. of training Urls: 5613
No. of validating Urls: 1871

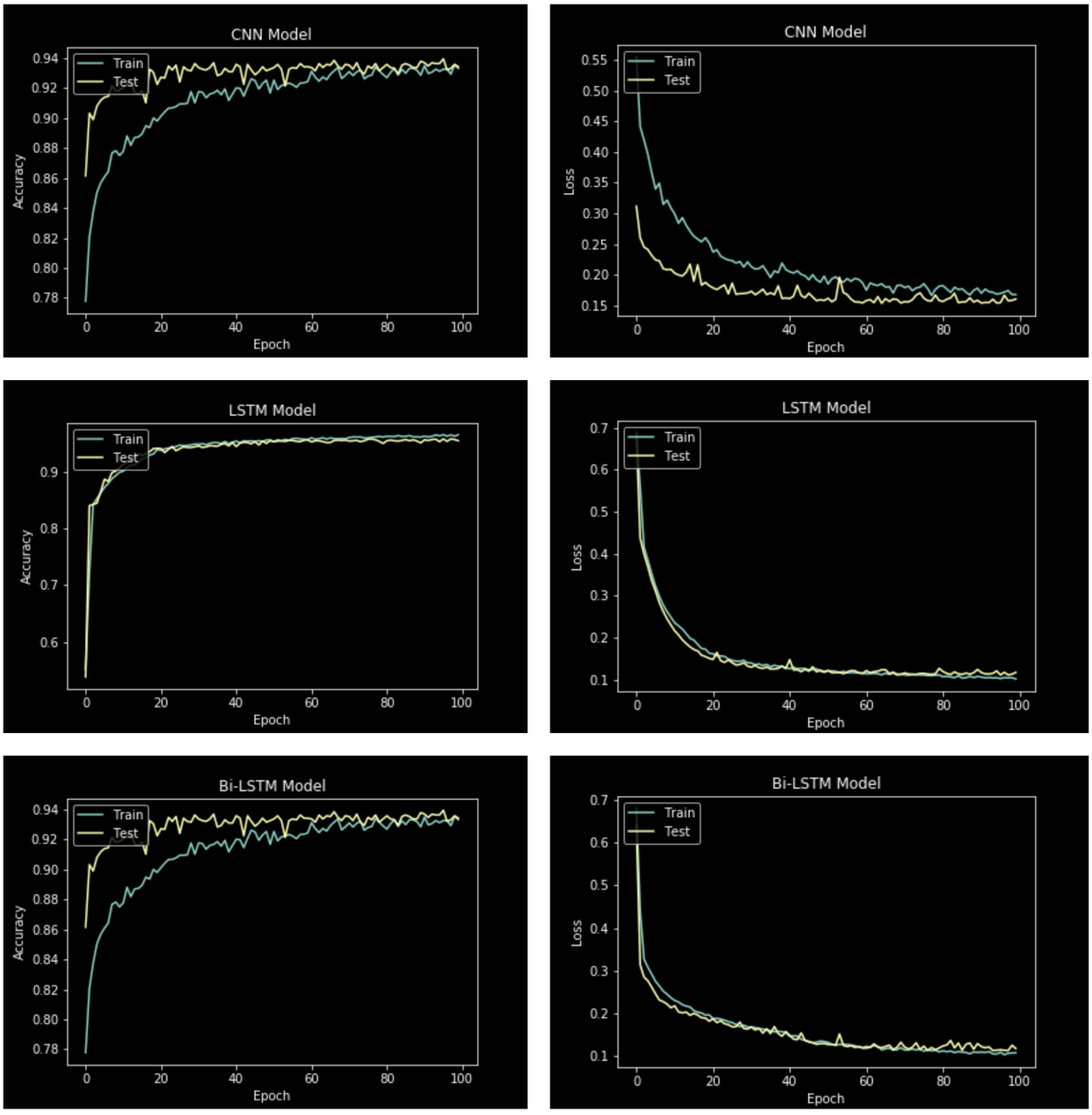
These are the hyper parameters used to train the emerging model of CNN & LSTM and CNN & Bi-LSTM are:

Optimizer: Adam optimizer
Loss Function: Binary Cross Entropy
Learning Rate: 0.001
epsilon: None
No. of training Urls: 5613

Model	Dataset	Accuracy
CNN	Alexa and Phishtank	0.934
LSTM	Alexa and Phishtank	0.945
Bi-LSTM	Alexa and Phishtank	0.9459

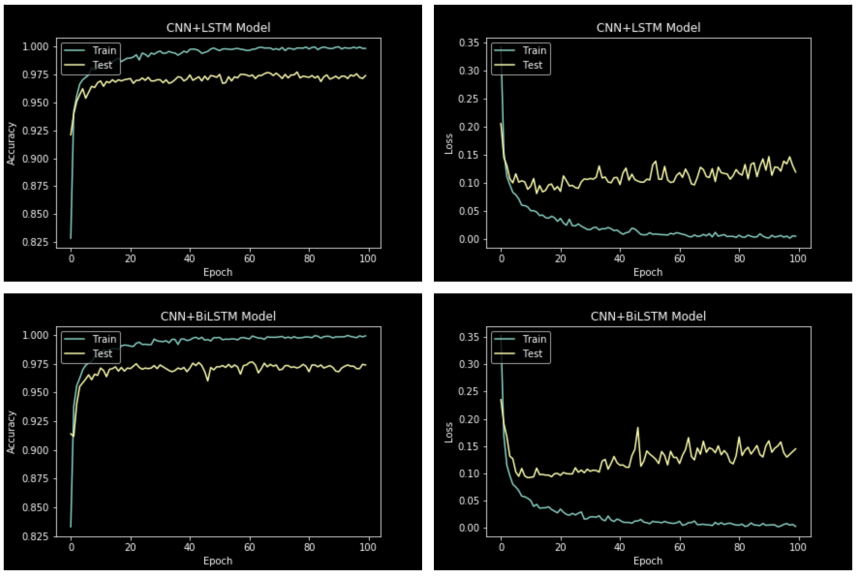
No. of validating Urls: 1871

Model	Dataset	Accuracy
CNN + LSTM	Alexa and Phishtank	0.9738
LSTM +Bi-LSTM	Alexa and Phishtank	0.975



Accuracy Vs Epochs

7.2 Accuracy and Loss Plot of various hybrid Deep Learning Models



Accuracy Vs Epochs

Chapter 8

Limitation of Our Model

Considering our approach, it is also not free from limitations. Following are some of the limitations of our multi-class URL classification and attack type identification system.

- Our methodology lacks analysis and detection of obfuscated JavaScripts in the Webpages, which is the major cause behind attacks like drive-by downloads, XSS, malware-delivery etc.
- There is need to investigate more discrimina- tive spam URL features to differentiate them efficiently from benign URLs.
- There is need to investigate more features of short URLs for the effective detection and at- tack type identification, because it is the most growing trend today on the micro-blogging sites like Twitter, Facebook etc.

Chapter 9

Conclusion and Future Work

In this paper we proposed URLNet, a CNN based deep neural network for Malicious URL Detection. Existing approaches mostly used Bag of Words like features, and this caused them to suffer from some critical limitations, including inability to detect sequential concepts in a URL string, requiring manual feature engineering, and inability to handle unseen features in test URLs. We proposed Character CNNs and Word CNNs for this task, and jointly optimized the network. Moreover, we proposed advanced word-embedding techniques which are particularly useful to deal with rare words, a problem usually observed in malicious URL Detection tasks (and not in traditional NLP tasks). This approach also allowed to learn embeddings from unseen words at test time, and exploit subword information.By comparing the three aspects (precision, recall, f-socre) of CNN, Bi-LSTM and CNN-BiLSTM, we can get a conclusion. Through the following three accuracy graph comparison, we can see that the use of CNN+BiLSTM model are slightly higher than the CNN and Bi-LSTM algorithm.

We plan to develop a framework using this approach and deploy it for a large scale real world test. We will also investigate the effectiveness of online algorithms as they have been found to outperform traditional batch algorithms in problem similar to ours. We believe that by looking into the contents of web pages, we can further improve false positives and negatives. We will also investigating this matter as well.

Various famous deep learning architecture proposed which are improved the accuracy upto great extend. We will tried to implement attention network which gives special attention on specific features of URLs. Suppose, in transaction websites we must have to pay special attention to the protocol of URL and domain name of URL instead of other features etc. Attention network improve result in most of the case. We are working it in our case but we did not reached upto the standard, but we are unable to find the reason, may be some fine tuning are required which we did not done. Complexity of attention network is high. Since Deep learning model learns weights and

create a model, everything inside the model acts like a black box. Its was proven that in deep learning as we increase the complexity of model explainabilty going to reduce.

References

[1] Bengin Data Collection from alexa. https://www.alexa.com/topsites. Accessed: 2019-06-30.

[2] Mozilla Phishing Protection using black-listing. https://support.mozilla.org/en-US/products/firefox/protect-your-privacy. Accessed: 2019-06-30.

[3] Phishing Data Collection from phishtank. https://www.phishtank.com/phish_search.php?verified=u&active=y. Accessed: 2019-06-30.

[4] Understanding CNN for text dataset. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/. Accessed: 2019-06-30.

[5] Understanding emerging technique. https://towardsdatascience.com/get-started-with-using-cnn-lstm-for-forecasting-6f0f4dde5826. Accessed: 2019-06-30.

[6] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*, pages 54–60. ACM, 2010.

[7] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

[8] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.

[9] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[10] Hung Le, Quang Pham, Doyen Sahoo, and Steven CH Hoi. Urlnet: Learning a url representation with deep learning for malicious url detection. *arXiv preprint arXiv:1802.03162*, 2018.

[11] Jin-Lee Lee, Dong-Hyun Kim, and Lee Chang-Hoon. Heuristic-based approach for phishing site detection using url features. In *Proc. of the Third Intl. Conf. on Advances in Computing, Electronics and Electrical Technology-CEET*, pages 131–135, 2015.

[12] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM, 2009.

[13] Joshua Saxe and Konstantin Berlin. expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. *arXiv preprint arXiv:1702.08568*, 2017.

[14] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[15] Toshiki Shibahara, Kohei Yamanishi, Yuta Takata, Daiki Chiba, Mitsuaki Akiyama, Takeshi Yagi, Yuichi Ohsita, and Masayuki Murata. Malicious url sequence detection using event de-noising convolutional neural network. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2017.

[16] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. Relation classification via convolutional deep neural network. 2014.

[17] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[18] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM, 2007.

[19] C Zhou, C Sun, Z Liu, and F Lau. A c-lstm neural network for text classification. arxiv 2015. *arXiv preprint arXiv:1511.08630*.