Using Deep Learning on URL Strings to Detect Rogue Websites

While the Internet is now used to provide a range of useful services, it is also increasingly being used maliciously. As it is difficult to keep track of everything on huge, complicated systems manually, a range of automation strategies are employed. Security applications of deep learning have attracted attention in recent years. If deep learning can be used to assist humans in fields where experience and knowledge are crucial, this should enable a greater number of people to engage in higher-level tasks and, as a result, make it possible to provide safe services. In this issue, we present our attempt to use deep learning to prevent cyberattacks.

2.1 Advent of the Web and the Battle against Malicious Sites

Some 30 years have passed since Tim Berners-Lee, then a fellow at CERN (the European Organization for Nuclear Research), released CERN httpd, the first ever World Wide Web (WWW) server software. It constituted a means of using hypertext on the Internet, and combined with HTTP (Hypertext Transfer Protocol) and URLs (Uniform Resource Locators), it provided the technology to connect the world's information resources in a blink of an eye. It is no coincidence that the 1980s and 90s were also a time that saw the explosive spread of TCP/IP-equipped BSD UNIX, particularly among educational institutions, laying the groundwork for connecting the world's computers to one another. The release of Mosaic, a GUI-based Web browser, by the United States' National Center for Supercomputing Applications (NCSA) also made it possible for non-computer-experts to easily access the world's information. Web technologies continue to evolve even now, with new services popping up all over the place daily.

Something common to all technologies is that those technologies with the potential to make the world a better place can also make it worse. As all sorts of services become available online, so too emerge attempts to deceive and defraud via the Web. A common example involves setting up a fake version of a well-known service or banking website and sending out fake emails or other communications to steer users into the site, which is then used to steal their personal information, passwords, and the like. Fraud and deception existed before the advent of the Web, of course, but just as with email spam, digitization has lowered the cost and made it possible to target a larger number of people. The information age has brought benefits for both legitimate society and its underbelly alike.

Protecting users from accessing malicious sites has been a key issue for network services operators in recent years. On their end, ISPs commonly provide services that block rogue sites for their users. If you are responsible for your organization's information systems, perhaps your activities involve having some sort of security software installed for your organization's users. The technology commonly used to defend against rogue sites at present basically uses blacklists. But, as you can probably imagine, it is not really feasible to cover the entirety of the vast space that is the Web using blacklists alone. Researchers have been looking for more efficient ways of recognizing malicious sites. One attempt, for example, has involved using the domain names of known rogue sites to mechanically derive similar strings, thereby producing a large number of new potential malicious domain names from a short blacklist[*1]. Another has involved going beyond the idea of mere lists to look at features such as when the domain was registered and its Google search ranking, with domain names that have only recently been registered, have a low search ranking, and so on being treated as less trustworthy[*2]. Other techniques that have been proposed include actually retrieving and analyzing the content of web pages via a transparent proxy to detect whether a site is malicious or not[*3]. And as deep learning has advanced in recent years, it has also increasingly been employed in security applications.

*1 P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive blacklisting to detect phishing attacks," in 2010 Proceedings IEEE INFOCOM, ser. INFOCOM 2010, pp.1–5.

*2 S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in Proceedings of the 2007 ACM Workshop on Recurring Malcode, ser. WORM '07. New York, NY, USA: ACM, November 2007, pp.1–8.

*3 Y. Zhang, J. I. Hong, and L. F. Cranor, "CANTINA: A content-based approach to detecting phishing web sites," in Proceedings of the 16th international conference on World Wide Web, ser. WWW '07. ACM, May 2007, pp.639–648.

2.2 Meaning Hidden in URL Strings

The battle against rogue sites is a never-ending one. As soon as a technique for defending against such sites is devised, a mechanism for avoiding it appears. Even so, it is still important to consider new defense techniques if we are to make the Internet safer.

The proxy approach of actually retrieving page content to determine whether a site is rogue has the advantage in terms of detection rates. The act of actually accessing a site, however, can be dangerous in some cases. Given processing load, privacy, and other issues, methods that do not involve retrieving any actual content have also been proposed. The simplest of these is to look solely at the URL itself. The issue here is whether the strings that make up the URL contain any information that can indicate whether a site is rogue or not.

No one has a precise answer to this question. But a look at past research shows that some people have thought there may be meaning to be found. One well-known idea, for example, is to look at whether the domain name is a pronounceable string. Domain names often have something to do with actual goods or services, so URL strings are often based on natural language words and names and thus inevitably turn out to be strings that humans can pronounce. Some malware uses mechanically generated domain names (from a domain generation algorithm, or DGA), and in many cases these names consist of strings that cannot be pronounced. The idea is that if a distinction can be made here, it may be possible to distinguish between ordinary and suspicious Internet access.

Another idea is that an unusually large number of subdomains (host names with lots of dots in them) and an unusually deep path (URLs with lots of slashes in them) often indicate malicious intent. Under this approach, it is common practice to consider various criteria based on empirical rules, combining those criteria to determine whether something is malicious or not.

And at the forefront of techniques in this area, researchers are looking into the use of deep learning to assess URLs.

2.3 Resurgence of Deep Learning

Deep learning began to rise in the popular mindset around five or six years ago. Neural networks themselves, which are used in deep learning, have actually been around for long enough to be labeled as classical. However, the approach of deep learning, which uses multilayer neural networks, was long thought of as being hamstrung by practical impediments given the amount of calculation involved and the technical difficulties in getting models to learn properly. Flash forward to the 2010s, though, and the development of techniques that produced remarkable results in the area of image recognition flung the field into the spotlight. Opinion varies, but the most common account seems to be that the deep learning-based image recognition system[*4] demonstrated by Alex Krizhevsky and colleagues at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 marked the start of the modern wave of deep learning that has propagated through to today. The model achieved a sharp reduction of 10 percentage points in the error rate from the previous mark of around 25%, demonstrating that deep learning can be applied to real-world scenarios. Use of deep learning subsequently became widespread, mainly in image and voice recognition, and it has also been applied to natural language translation, document classification, and even the strategy game Go.

In networking as well, researchers continue to put forward deep learning-based techniques, mainly in the area of security. In this article, we describe our proposal[*5] for detecting malicious sites based on URL strings, but we note that this is naturally not the first proposed approach of its type in the

*4 A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, 2012.

*5 K. Shima, D. Miyamoto, H. Abe, T. Ishihara, K. Okada, and Y. Sekiya, "Classification of URL bitstreams using Bag of Bytes," in Proceedings of First International Workshop on Network Intelligence (NI 2018), 2018.

world and that we expect many researchers and engineers to put forward even better strategies going forward. Creating something that will work indefinitely is not easy in network environments, particularly in distributed autonomous environments such as the Internet. Systems and data change with the times, and it is impossible to know all of the information therein because we can only ever see a portion of the world at once, and the information we can see becomes outdated almost as quickly as you can blink.

Deep learning is not an all-powerful approach. We do not yet know whether it will produce an intelligence that exceeds human capabilities, an idea that is often bandied about, but we do know what it is currently capable of.

Deep learning is a subset of machine learning in which a set of operations are performed on a given input vector to produce a separate output vector. It is used in classification problems and identification problems. An example of an image recognition application is a system that accepts an image of a cat (converted into a vector representation) and gives either a 0 or a 1 as output to indicate whether or not the image is a cat. Deep learning requires a large amount of data to determine what set of operations to perform. In the cat example, this would be a large quantity of cat images as well as images of objects other than cats. This is called the training data. When the data are labeled so that the answers are known, this is called supervised learning, and when this is not the case, it is called unsupervised learning (semi-supervised techniques that fall between these two also exist). Deep learning has produced great results with these sorts of classification problems.

2.4 Vectorizin URLs

Now let's move on to our URL classification problem. Our objective is to determine whether a given URL points to an ordinary, unproblematic site or to a rogue site. To use deep learning methods, we first need to convert the URLs into a vector representation that a deep learning model can take as input.

Before the rise of deep learning, the task of defining these vectors (feature engineering) was crucial to machine learning. This is because how you define what information is necessary and sufficient for differentiating your data ahead of time greatly influences performance. As mentioned previously, in URL classification, a variety of factors have been studied and validated as features that can be used to distinguish URLs, including whether the strings are pronounceable, the number of dots and slashes, the ratios of alphabet, symbol, and number characters, the position of characters, the frequency of n-gram strings, and so on. Haphazardly increasing the number of features can affect how long it takes to crunch the numbers, so with conventional machine learning methods of the past, experts with a deep knowledge of the dataset in question had to carefully select features likely to be useful through painstaking analysis of existing data.

It is said that deep learning, in contrast, can discover features for itself when trained using a large amount of data. In reality, it is not that simple, as careful preprocessing of the data often influences the final results, but it is also true that quantity of data can mitigate the effort needed for feature engineering to an extent.

To classify URLs in our system, we will not use existing features. Instead, we define a simple transformation to convert URL strings into fixed-length vectors. Past wisdom certainly can be used to classify URLs, but it is not necessarily possible to define features that will always be useful when working with other datasets in the future. There is also the tentative prospect of perhaps being able to use the same strategy on other datasets if we discover that it is possible to distinguish URLs using simple preprocessing and a large quantity of training data.

The vectorization procedure we adopted is as follows.

1. Split the URLs into individual characters
2. Convert the characters to hexadecimal ASCII codes
3. Extract byte values beginning at the start of the host part and the path part separately, shifting 4 bits at a time
4. Count how many times each value (from 0x00 to 0xFF) appears in the host part and path part, respectively, to form 256-dimensional vectors
5. Combine the 256-dimensional vectors created from the host part and the path part to form a 512-dimensional vector
6. Normalize the vector

Figure 1 shows steps 1 through 3. In step 4, we then count the values. The counts (in parentheses) for the sequence shown in Figure 1 are as follows.

0x16 (1), 0x2E (3), 0x42 (1), 0x61 (1), 0x64 (1), 0x69 (2), 0x6A (2),0x70 (1), 0x72 (1), 0x77 (5), 0x96 (2), 0xA2 (1), 0xA7 (1), 0xE6 (3)

Denoting the host vector as ***V*** and the *i*th element as v_i (where *i* is the extracted value), $v_{0x16} = 1$, $v_{0x2E} = 3$, $v_{0x42} = 1$, ... , $v_{0xE6} = 3$. Positions corresponding to unextracted values will contain a 0. Any original URL of any length can be converted into a 512-dimensional vector in this manner. But the longer the URL, the larger the size of the vector, so we normalize the vectors in step 6. We define the resulting 512-dimensional vector to be the "URL feature vector".

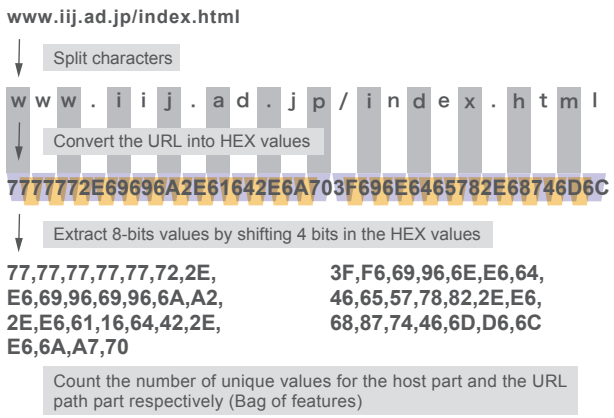

Figure 1: URL Vectorization

2.5 Designing the Neural Network

We are now ready to convert the URLs to fixed-length vectors. Next we have to decide how to train on that URL feature vector. In our attempt here, we used a simple 3-layer neural network. Although this is a rather shallow network for deep learning, it should be sufficient to determine whether this sort of method is effective or not.

Figure 2 shows the topology of the neural network we used. Some readers may have seen this topology somewhere before. This three-layer, fully connected topology appears as a sample in the Chainer open source deep learning library (https://chainer.org/) developed by Preferred Networks, and is used to recognize handwritten numerals from the MNIST dataset (http://yann.lecun .com/exdb/mnist/). Our work is based on this, with the following two changes.

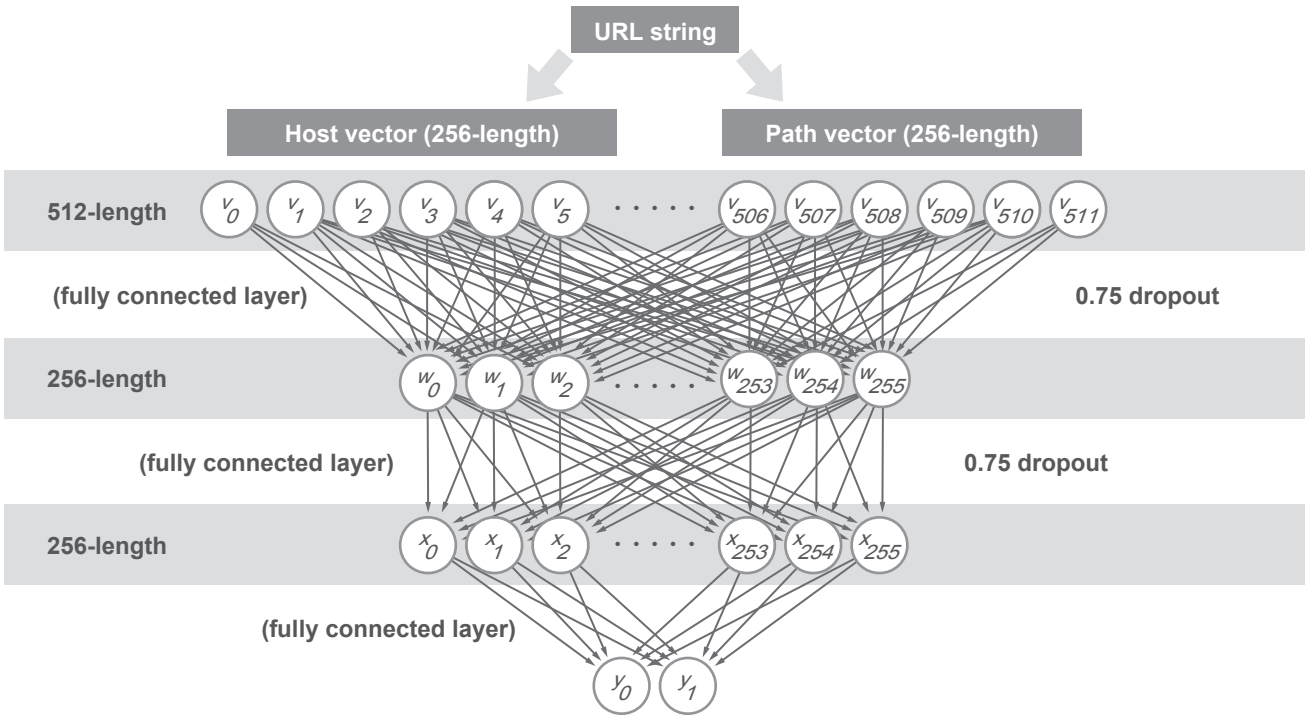

Figure 2: Neural Network Topology

1. Number of input/output dimensions: With the MNIST sample, input images are 28x28 pixels, yielding 784 input dimensions. And the output is 10-dimensional because the output values are the digits 0 through 9. Our input is the 512-dimensional URL feature vector, and our output is 2-dimensional, being the value 0 or 1, indicating whether a site is rogue or not.
2. Dropout rate: The MNIST sample does not use dropout, a method for preventing overfitting, but we observed serious overfitting with our data and thus set a fairly high dropout rate.

We used Chainer to verify our proposed approach. The neural network model we built in Chainer is shown in Table 1.

2.6 Data Source Selectionk

With our data structures and neural network model in place, we can now use actual data to verify our approach. Two major issues present themselves in dynamic environments like the Internet.

1. Data accuracy: With datasets like MNIST, the data are already fully validated and properly labeled (in the case of MNIST, this means the handwritten digits and the values they represent). Accurately labelling data observed/collected via the Internet, meanwhile, can be problematic. If the model is trained on incorrect information, it will naturally end up predicting incorrect answers.
2. Completeness: It is not possible to show whether the data used to train the model is a true representation of the general picture. If the model is trained on biased data, it will be unable to cope with different patterns when they appear. In the case of handwritten digit recognition, the problem space is somewhat limited since it only involves the digits 0 through 9, whereas URL strings on the Internet represent a virtually unlimited space, so the scope of applicability will naturally differ.

Under the provision that such problems exist, it is important to prepare data that is as accurate and complete as possible. The data we use comprises active phishing sites listed on PhishTank (https://www.phishtank.com/). PhishTank by no means provides an exhaustive list of all the rogue sites out there, so although completeness is not guaranteed, the data offer a degree of credibility since the process of determining whether a site is a phish or not involves human verification via a voting system. Compiling data on ordinary (non-rogue) sites is more difficult. For verification, we take sites appearing in a particular research institution's access logs and exclude those listed on PhishTank, defining such sites to be non-rogue sites. To enhance completeness, however, we would need to perform additional tests based on different types of access logs.

```
from chainer import Chain
import chainer.functions as F
import chainer.links as L
class Model(Chain):
    def __init__(self):
        super(Model, self).__init__()
        with self.init_scope():
            self.l1 = L.Linear(None, 256)
            self.l2 = L.Linear(None, 256)
            self.l3 = L.Linear(None, 2)
    def __call__(self, x):
        h1 = F.dropout(F.relu(self.l1(x)),
                       ratio=0.75)
        h2 = F.dropout(F.relu(self.l2(h1)),
                       ratio=0.75)
        y = self.l3(h2)
        return y
```

Table 1: Neural Network Model Built in Chainer

2.7 Applicability of Deep Learning

We randomly extract around 26,000 URLs each from the two types of data sources prepared per the previous section. Of this, 80% is used for training. When we used the remaining 20% to assess accuracy, we found that we were able to correctly distinguish between rogue and non-rogue URLs for 94% of the data. Opinion may vary on whether this is a good result or not. Some classification methods put forward in the past have achieved better outcomes than this figure. In some cases, those methods also used information other than just the strings (e.g., Whois information, Google search ranking), so simple comparisons with our approach are not possible. Another attempt employed deep learning on URL strings alone[*6] in a manner similar to our approach, yielding better classification accuracy than we achieved. Yet when we independently implemented the neural network proposed in that study and tested it on our dataset, we were unable to replicate the high accuracy reported in that paper. The takeaway here is that even with the same neural network model, accuracy can vary significantly depending on the dataset used for training.

Since it is impossible to obtain all the world's data, trained models will inevitably carry some bias. The term big data seems to have fallen by the wayside a bit lately, but we think it is likely that organizations with large stores of wide-ranging data will continue to occupy an advantageous position in the deep-learning world as well; indeed, that advantage may even widen.

2.8 Conclusion

We have described our attempt to apply deep learning to the task of identifying rogue URLs. Despite only using a simple neural network to test our approach, we were able to classify URLs with 94% accuracy. This exercise also reaffirmed the difficulties in collecting data and the advantages that having data imparts.

We can expect deep learning to increasingly be applied to network data ahead. With computing power increasing, it has become relatively easy to use deep learning. We hope to incorporate new technologies into our approach as we work toward making the Internet even safer going forward.

Acknowledgments
This research was supported by JST, CREST, JPMJCR 1783.

Keiichi Shima
Deputy Director, Research Laboratory, IIJ Innovation Institute

*6 J. Saxe and K. Berlin, "eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys," CoRR, vol. abs/1702.08568, February 2017.