What's in a URL: Fast Feature Extraction and Malicious URL Detection

Rakesh Verma
Department of Computer Science
University of Houston
Houston, Texas
rmverma@cs.uh.edu

Avisha Das
Department of Computer Science
University of Houston
Houston, Texas
adas5@uh.edu

ABSTRACT

Phishing is an online social engineering attack with the goal of digital identity theft carried out by pretending to be a legitimate entity. The attacker sends an attack vector commonly in the form of an email, chat session, blog post etc., which contains a link (URL) to a malicious website hosted to elicit private information from the victims.

We focus on building a system for URL analysis and classification to primarily detect phishing attacks. URL analysis is attractive to maintain distance between the attacker and the victim, rather than visiting the website and getting features from it. It is also faster than Internet search, retrieving content from the destination web site and network-level features used in previous research.

We investigate several facets of URL analysis, e.g., performance analysis on both balanced and unbalanced datasets in a static as well as live experimental setup and online versus batch learning.

Keywords

Online Learners, Character N-grams, Historical experiment, Batch classifiers

1. INTRODUCTION

In phishing, attackers lure an unsuspecting victim via electronic communication and compromise their personal information for malicious use.According to the RSA Online Fraud Report[1] for 2016, "phishing attacks have cost global organizations $4.6 billion in losses in 2015." The total loss incurred increased by 19% from the second half of 2011 to the first half of 2012 according to RSA reports for Online Fraud. Phishers use a number of methods to 'phish' their victims; but one generic feature that most of these attack methodologies share is: the use of a 'poisoned' link (URL) pointing to the fraudulent website shared with the victims through emails, chat sessions, blog posts, etc. Through such carriers, attackers tempt the individuals to click the URLs, who unknowingly end up divulging their private information to the attackers who use the knowledge for malicious use.

Since phishing websites are up for only a few hours to a few days, a fast and robust classifier is indeed indispensable. For efficiency, URL analysis is particularly attractive as opposed to website analysis (which can be defeated by copying the entire target web site) or host/DNS based features. Since phishing sites are short-lived, some features such as popularity, may not be available for them. URL analysis has the additional benefit of maintaining a distance between the victim and the malicious website. A critical component for detecting phishing attacks is a strong and robust URL classification system. Hence we focus on URL analysis in this paper. Our contributions are as follows:

- We extract a richer class of features, viz., *Character N-grams* (Section 2) from a range of URL datasets, which subsume features consisting of punctuation marks, special symbols or single-letter distributions for the classification and analysis of URLs.
- We demonstrate the robustness of our classifiers on a couple of large real-world URL datasets (Section 2.3) that have been used for previous related research.
- We also analyze the performance of our classifiers when trained on balanced as well as unbalanced URL datasets for the live URL setup.
- We present fast, robust and accurate classifiers using a range of both online and batch learners (Section 3.1). We also compare the classifier performance with on-the-fly and retraining methods for model building.
- We include our evaluation on recent URLs from publicly available resources in a live URL extraction setup (Section 3.2).
- We show that phishing URL analysis is significantly different from spam website detection through URL and domain analysis (Section 3.3).
- We present an analysis of the usefulness of Character N-grams and a security analysis of our work in Sections 4 and 5 respectively.

Previous researchers [14, 27] have considered URL classification based on link analysis only by proposing systems which use search engine based features as well as web site

[1] https://www.rsa.com/en-us/perspectives/industry/online-fraud

IWSPA '17, March 24-24 2017, Scottsdale, AZ, USA.

ACM ISBN 978-1-4503-4909-3/17/03..

DOI: http://dx.doi.org/10.1145/3041008.3041016

popularity and blacklists as major attributes for URL detection. However, with search engines rate-limiting queries and charging for them, methods based on Web search are becoming infeasible. There have been some phishing URL detection attempts in the past using special symbols [20], domain information [27, 29], or single-character distributions [26], but few have investigated character N-grams as features for phishing URL detection. Moreover, mostly batch machine learning algorithms have been used for phishing URL classification and analysis.

2. EXPERIMENTAL SETUP

2.1 Feature extraction

Character N-grams: The hypothesis is that obfuscation techniques used by phishers to fool people skew the character distributions in the phishing URLs. This is the main motivation behind our set of features: *Character N-grams* extracted from URLs are overlapping sequences of N consecutive characters extracted from the URLs where the value of N ranges from 1 to 10. For example, the first three bigrams from the URL 'example.com', would be: ex, xa and am. This is much richer than the bag-of-words approach used by researchers in [11] as it captures the punctuations, misspellings, etc. in the URLs. Here N represents the length of the character substring. As N becomes large, not only words but also smaller substrings with no obvious meaning might be captured. For our feature sets, we extracted all substrings of length ≤ N. For example, when $N = 3$ we extract from 'example.com' the following set of features: e, x, a, m, ex, xa, am, exa, xam, etc.

More verification of this underlying hypothesis is done in Section 3. Additionally, our feature extractor is exceptionally fast, processing unigrams from a dataset of around 300,000 URLs takes around 111 seconds to generate 94 features, while bigrams take around 373 seconds with a larger number of features, around 6100.

2.2 Learning algorithms

We use Online Learning algorithms as the basis of our experimental classification and compare their performance with traditional batch learners (J48; PART and RandomForest). Online learners [23] learn a classification model on each URL instance at a time. For each learning round, the learner first receives a new incoming URL instance for detection; it then applies the classification model to predict if it is malicious or not. At the end of each classification round, if the true class label of the instance is available, the learner will make use of the labeled instance to update the classification model whenever the classification is incorrect.

We use the online learning algorithms: Perceptron (P) and Averaged Perceptron (AP) [8]; Passive-Aggressive (PA), Passive-Aggressive I (PA-I) and Passive-Aggressive II (PA-II) Algorithms [6]; Confidence Weighted (CW) Algorithm [9] and Adaptive Regularization of Weights (AROW) algorithm [7]. They have been used in two experimental setups: Retraining and On-the-fly training to evaluate the classifier performance.

2.3 Datasets

Here we give a brief overview of the sources and class distributions of the datasets used in our work. We use two major datasets, which are publicly available, and we demonstrate the results of our classifiers on a live URL dataset. We also study our classifier performance on a dataset consisting of Spam URLs and domains clubbed with legitimate URLs extracted from [28]. These datasets are described below.

Phishing URL datasets: In phishing URL detection experiments, we used static datasets as well as live URL feeds.

(a) Static URL datasets: In Section 3.1 we used the following 'static' datasets, **DMOZ** [26] and
APWG100 (this is our largest individual dataset). They are described in Table 1. However, these are older datasets with the URLs extracted directly from publicly available open projects like DMOZ (for legitimate) and PhishTank and Anti-Phishing Working Group (for phishing).

(b) Live URL datasets: To demonstrate the live nature of online learners to update on URLs fed to it in a sequential manner, we collect URLs for a live URL experiment. An experiment of similar nature has been performed previously in [19] with a balanced distribution of phishing and legitimate URLs.

However, in our experiment we feed our classifiers two types of URL datasets. We construct two training and testing datasets of **balanced** (approx. 1:1) and **unbalanced** (approx. 7:1) distributions of legitimate to phishing URLs. Both the training datasets comprise of a mixture of recent and older phishing URLs and we test on more current phishing URLs. For grabbing the legitimate URLs, we start at the top 100 sites on DMOZ and then travel to each legit website and grab the URLs internal to the site, this procedure was repeated until we reached at most 10 websites from the same start page. We grab the phishing URLs from PhishTank dataset. A description of the URL sources is given in Table 2.

Spam URL and Domain datasets: For the spam domain experiment we extracted the spam domains from the blacklists provided by the jwSpamSpy [2] used by [5], the dataset has 917 spam domains and 2,000 legitimate URL domains extracted from the links of the webpages visited randomly after starting at the top 100 sites on DMOZ. We call this dataset **SpamDom** Dataset.

We could not find a publicly available spam URL dataset so we extracted spam URLs from the LingSpam Corpus [1] of spam email messages to build a dataset of 409 Spam URLs and the 2,000 legitimate URLs extracted randomly by scraping the web after starting at the top 100 sited on DMOZ. These datasets will be made publicly available. This dataset is referred to as **SpamURL** dataset. The datasets were built with an unbalanced ratio of legitimate to spam links so that they can emulate a real world setting where legitimate instances are more common than malicious ones.

3. ANALYSIS AND RESULTS

Defining a positive occurrence to be a malicious URL, we consider a false positive as a benign URL that was classified as malicious. We chose to include false positive rate in our results because it has specific interest to the problem of phishing detection. When any software program is attempting to detect phishing attacks it is important that it has a very low false positive rate, because each false positive occurrence in a real system might mean an important message never reaches its recipient.

Classifier Codes: We used an open source Online Learning Library [22] from Google projects, which provides stand-

Table 1: Dataset Description for Static URL detection

URL Dataset	Legitimate (# & Source)	Phishing (# & Source)
DMOZ	11,275 from DMOZ Open Directory Project	11,271 from PhishTank (02/12/2014).
APWG100	126,511 randomly chosen from DMOZ	128,767 URLs from APWG

Table 2: Dataset Description for Live URL detection

URL Dataset	Legitimate (# & Source)	Phishing (# & Source)
Balanced Training	35000 (Top 100 on DMOZ and random web scraping, April 2016)	12475 from PhishTank (May 01 to June 30, 2016) and 18395 from APWG
Balanced Testing	2000 (Top 100 on DMOZ and random web scraping, April 2016)	1881 from PhishTank (July 01 to July 21, 2016).
Unbalanced Training	87375 (Top 100 on DMOZ and random web scraping, April 2016)	12482 from PhishTank (May 01 to June 30, 2016)
Unbalanced Testing	13200 (Top 100 on DMOZ and random web scraping, April 2016)	1881 from PhishTank (July 01 to July 21,2016).

alone programs for learning and predicting using the algorithms: Perceptron, Averaged Perceptron, Passive Aggressive, Passive-Aggressive I and Passive-Aggressive II written on a C++ code base. We used [17] for running the Adaptive Regularization of Weights online learning algorithm. For all the experiments in Sections 3.1 and 3.2, we run the online learners with a retraining option of 20 iterations on the shuffled training data instances (shuffling was enabled to emulate the real-world scenario).

Performance metrics: We run our classifiers to study the accuracy values of the online learners on the datasets described in Section 2.3. *Accuracy* is an intuitively understandable performance metric. In data classification applications, accuracy is given as the number of correct classifications out of the total number of test cases. With the usual notations, TP for True Positives, FP for False Positives, FN for False Negatives and TN for True Negatives, we have:

$Accuracy = \frac{(TP+TN)}{(TP+FN+FP+TN)}$

However, as stated in [30], in a practical real-world scenario the cost of classification is not entirely dependent on the accuracy of the classifier. A weak classifier can perform very well in terms of accuracy when tested on an unbalanced dataset. The authors, therefore, propose the following two cost sensitive measures in [30]: (I) Weighted sum of sensitivity and specificity and (II) Weighted cost of learning

On calculating the cost-sensitive metrics on the balanced datasets (like DMOZ, APWG100, Balanced Live Testing) we observed a nature similar to the behavior of the accuracy metric of the classifier. We therefore do not present our observations of this metric on balanced datasets separately in this paper. Instead, we present the classifier performance on Unbalanced Live datasets in terms of the cost-sensitive metric, i.e. *the weighted sum of sensitivity and specificity* and analyze the classifier performance based on the nature of the metric. With the usual notation mentioned earlier, we have:

$Sum_{csoal} = \eta_p \times Sensitivity + \eta_n \times Specificity$

$Sensitivity = TruePositiveRate\ (TPR) = \frac{TP}{TP+FN}$

$Specificity = TrueNegativeRate\ (TNR) = \frac{TN}{TN+FP}$

where $0 \geq \eta_p$, $\eta_n \leq 1$ and $\eta_p + \eta_n = 1$. We set $\eta_p = \eta_n = 1/2$.

3.1 Classification of Static URL Datasets

Section A: Online Learning Algorithms: In this section, we test the performance of our classifier on 4 'static' phishing URL datasets (APWG100 and DMOZ) described in Section 2.3.

For an unbiased classification, we split the entire dataset and 70% of the instances are used for training while the remaining 30% is used for testing purposes. We report the performance based on the accuracy of each classifier on the datasets as described in Table 1. Figures 1 and 2 report the accuracy of the classifiers on the datasets.

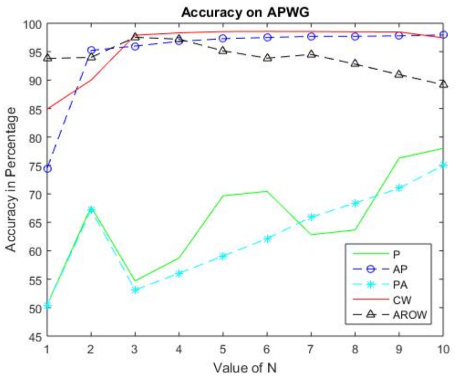

Figure 1: Accuracy on APWG100

For Figure 1, we observe that the simple perceptron and passive-aggressive classifier performance does not go beyond an average value of approximately 75%. The other three online learners, AROW, AP and CW perform considerably higher and have close to 99% accuracy.

For Figure 2, CW has a higher accuracy than all the other classifiers In Figure 2 we observe AROW as the second best classifier. All the other classifiers have an accuracy ranging from 75% to 85%.

Training and Testing times We prefer online learning algorithms not only for their effectiveness, and robustness in the case of AROW and CW, as seen from the results, but also for their fast training and testing times. Table 3 reports the times that these algorithms take to build the model (*training time*) and also test the model on the dataset (*testing time*).

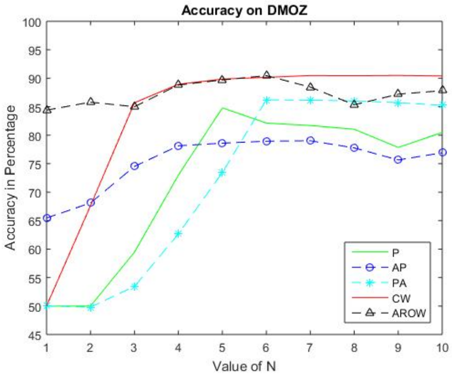

Figure 2: Accuracy on DMOZ

We report the training and testing times using the 10-gram features extracted from the APWG100 Dataset. The number of distinct features present in the dataset was: 27601162.

Table 3: Training and testing times on 10-grams extracted from APWG100

Online Algorithm	Training Time (s)	Testing Time (s)
P	232.66	61.61
AP	223.36	64.60
PA	222.22	60.72
CW	449.59	61.21
AROW	1265.07	21.44

Section B: Batch Learning Algorithms

We used WEKA V.3.8 [12] library to construct our batch learner models. We built the models based on decision tree based algorithms like J48 and RandomForest and also evaluated a rule-based learner like PART, which were found to be effective in [26] for classification of phishing URLs. We refer the reader to [26] or any good machine learning text for a description of these three classifiers.

Specifications: We used WEKA's default implementation of the RandomForest class which incorporates ten unlimited depth trees, with a feature selection of $log\ P + 1$, where P is the total number of features. For J48, we used the WEKA 3.8 default pruning threshold and parameters. Similarly, for PART we used the default parameters assigned by WEKA.

Even after pruning,[2] we could only run files containing up to bigrams on our machine. Table 4 reports the results on DMOZ datasets. Because WEKA was taking enormous time, we had to dispense with the usual 10-fold cross-validation and did a 5-fold cross validation instead of using separate training and testing datasets.

From the above results, we observe that all the online learning algorithms are much faster (the difference in languages, viz., C++ and Java, cannot explain the total difference) and the best-performing, AROW and CW, are at least as accurate. This was previously observed in [4] for other problems; we prove it here for phishing URL classification.

Table 4: Batch Learning Algorithms on DMOZ dataset

Algorithm	N-gram	Accuracy	Training Time(s)	Testing Time(s)
J48	Unigrams	94.27%	41.6	5.41
PART	Unigrams	97.45%	384.3	14.32
RandomForest	Unigrams	98.89%	200.1	10.27
J48	Bigrams	98.48%	762.34	23.7
PART	Bigrams	99.36%	2778.68	31.27
RandomForest	Bigrams	99.83%	1654.59	24.35

3.2 Classification of Live URL Datasets

The essence of an online learner lies in the fact that it builds a classification model in an incremental manner when fed with a number of URL training data in an 'online learning' fashion. In real world situation, the URLs are indeed fed to a classifier in a sequential manner as is also explained by the authors in [19] and [30]. In this section, we trained our model and tested on the various sets of URLs as explained in Table 2. The two sets vary in the ratio of the legitimate to phishing URLs present in the data. Here also, for an unbiased classification, we split the entire dataset and 70% of the instances are used for training while the remaining 30% is used for testing purposes.

Live URL feed in a Balanced (1:1) setup: We report the accuracy on the balanced live URL feed in Figure 3. The results reported in the figures demonstrate that AROW and CW are are the best performers on this dataset, achieving high accuracy (approx. 99.98%)

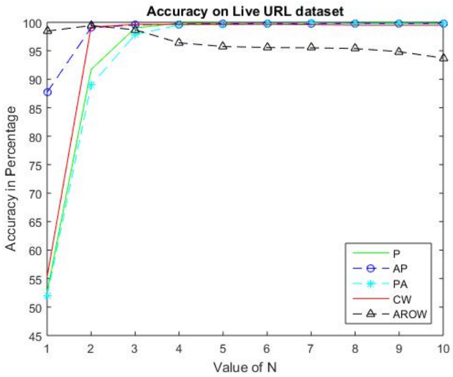

Figure 3: Accuracy on Balanced Live URL Dataset

Live URL feed in an Unbalanced (7:1) setup: We report the accuracy and the cost-sensitive parameter of weighted sum on the balanced live url feed in Figure 4.

For the unbalanced setup, the other online classifiers reported very low accuracy values (approx. 23% to 50%). We have reported here the best results which were achieved by CW and AROW in Figure 4. We chose to go up to a value of N=20 i.e. extracting 20 grams from a URL address to observed the classifier performance oh higher grams. This was done exclusively for this dataset due to extremely low accuracy values of the classifiers on the lower grams.

On-the-fly learning versus Retraining: The main purpose of this study is evaluation of online or on-the-fly learning when compared to retraining on input data. While the online method of learning is apt for balanced live URL data and static balanced and static unbalanced datasets, our experiments show that these may not be apt for unbalanced data which is supplied live to the URL classifier. We there-

[2] **Pruning rate**, K = The percentage of instance in which an n-gram must occur in order to be kept. For example, if there are 1000 data instances, and K = 1, then any N-gram that occurs in less than 10 URLs will be removed. In our experiment, we used K = 3.

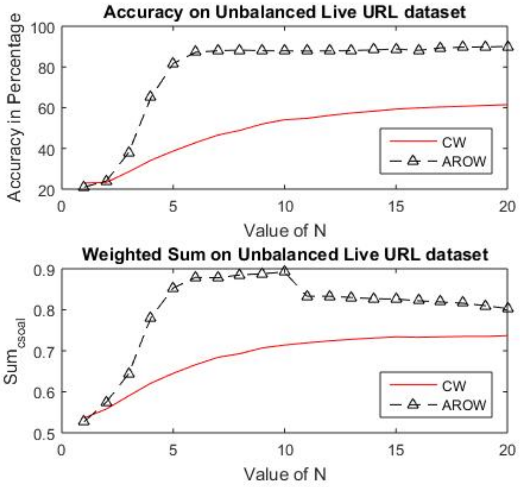

Figure 4: Accuracy on Unbalanced Live URL Dataset

fore present a subsequent comparison of the URL classifiers (AP and CW) on the Unbalanced live data when they are tested after training on-the-fly and also after retraining on the training dataset for 20 iterations. Figures 5 and 6 respectively give the accuracy and cost sensitive sum or cost of classification for N gram values of N=1 to 10 extracted from the input data for the two training modes (i=0 and i=20) on the data. One reason to choose AP and CW was the on-the-fly training option provided by the library used in our study.

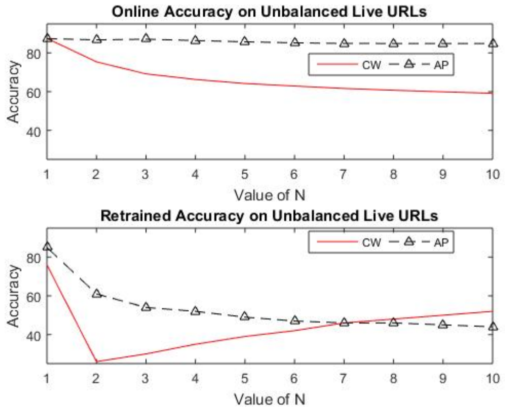

Figure 5: (a) Accuracy for On-the-fly training (b) Accuracy with retraining of online learners

Thus we observe that with high iterations and higher grams the accuracy drops but the misclassification cost decreases with a little retraining. Thus, there exists a trade-off between the classifier performance and the suitable method of training. Therefore, it somehow seems suitable to train on the data online for an optimum and fast performance. Learning on the training data for a few iterations for better performance on test data can be possible but not a better option.

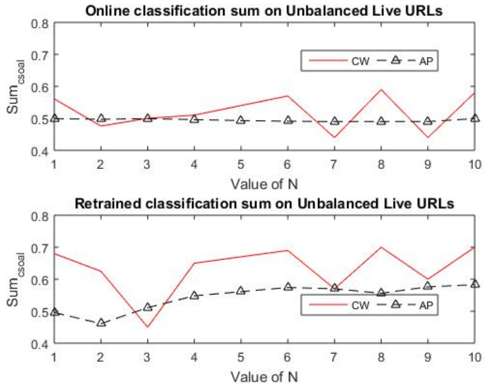

Figure 6: (a) Sum_{csoal} for On-the-fly training (b) Sum_{csoal} with retraining of online learners

3.3 Comparing Phishing and Spam Domains and URLs

How well can online learning algorithms work on different types of malicious, e.g., spam or sites hosting malware, attacks? We also performed experiments on domains and URLs that are not phishing in nature. For this experiment on domains and URLs, we used two different types of datasets: the PhishDom and the SpamDom Datasets. The classifier was tested on 18,254 domains extracted from phishing URLs (PhishDom Dataset) collected from Huawei and used in [28]. The experiments have been conducted by training the online learning algorithms for 20 iterations on the training data and tested on unseen test data (70%-30% split of the entire dataset). Figure 7 reports the accuracies of our classifier on the SpamDom dataset. Figure 8 reports the accuracies of our classifier on the PhishDom dataset. CW and AROW perform well on all two domain datasets. Curiously, all online algorithms work well on the SpamDom dataset due to the retraining on the data. We repeated our analysis for the SpamURL dataset and Figure 9 reports the accuracies on the dataset. Better performances were observed also for the SpamURL datasets.

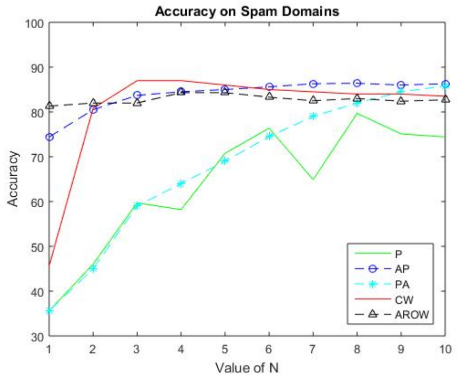

Figure 7: Accuracy on SpamDom dataset

We also trained the Spam URLs on the fly (zero iterations) and ran our classifiers on the test URL dataset. The best performance on Spam domains and Spam URLs was observed by the AP and CW algorithms (these learn-

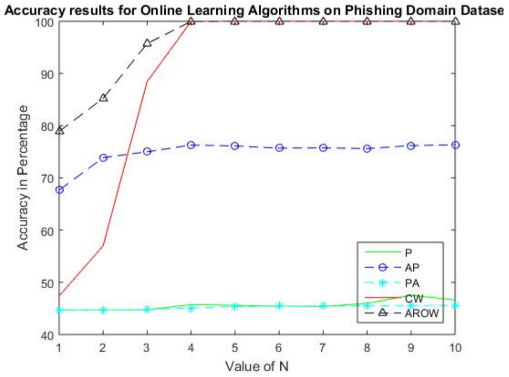

Figure 8: Accuracy on PhishDom dataset

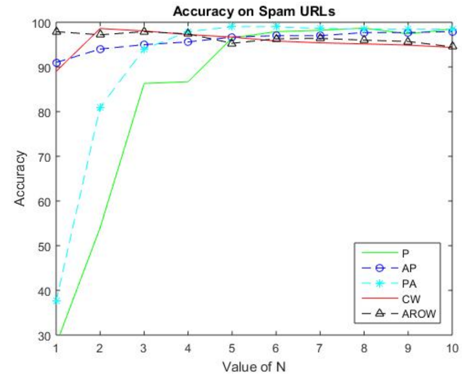

Figure 9: Accuracy on SpamURL dataset

ers were the ones which supported an on-the-fly training option). This is demonstrated in the Figure 10.

3.3.1 Cross-dataset validation across the domains and URLs

For this experiment, we reduced the sizes of the two domain data sets: SpamDom (S) and PhishDom (P), so that they are comparable in their sizes (shrinking them to have approximately 2000 URLs of type legitimate and malicious each). We run the best classifier (AROW) with uni-, bi- and tri-grams extracted from the SpamDom and PhishDom datasets and we observe the classification accuracy achieved when it was tested on a different dataset *without any re-training*. Table 5 reports the accuracies. In this and the following table the first row reports N values and the first column reports the ordered pairs (i, j) where i = training dataset and j is the testing dataset. It is clear that training on spam domains does not help with detecting phishing domains, the other direction is better, i.e., training on phishing domains does help somewhat in detecting spam domains.

Similarly, we run AROW on the first 3 values of N grams extracted from the Spam URL (SU) and Phishing URL (PU) datasets and we observe the classification accuracy achieved when it was tested on a different dataset *without re-training*. The phishing URL dataset used in this experiment are 1000 legitimate and phishing URLs selected randomly from the dataset described and used in prior research [28] on phishing URL analysis. Table 6 reports the accuracy values.

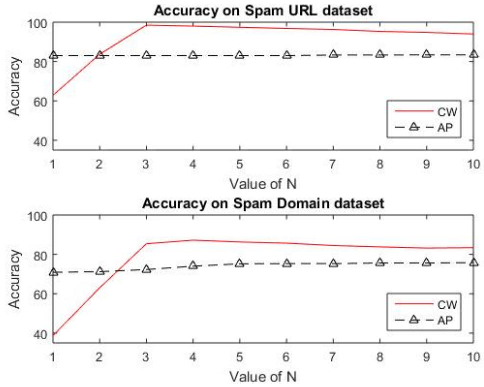

Figure 10: Accuracy for On-the-fly learning on SpamURL and SpamDom

Table 5: AROW accuracies on cross-natured Domains

Ordered Pairs	1	2	3
(P, P)	79.62%	87.98%	98.75%
(P, S)	78.53%	86.65%	88.55%
(S, S)	99.81%	99.81%	97.67%
(S, P)	50.68%	51.20%	51.65%

Table 6: AROW accuracies on cross-natured Spam and Phishing URLs

Ordered Pairs	1	2	3
(PU, PU)	84.57%	91.32%	90.67%
(PU, SU)	74.15%	66.23%	64.80%
(SU, PU)	60.94%	64.34%	61.64%
(SU, SU)	73.25%	77.65%	76.25%

4. WHY DO N-GRAMS HELP?

The nature and construction of the phishing URLs are essentially different than from those of legitimate URLs because there are certain letters, word fragments and characters in the URLs. In this section, we delve deeper into analyzing the *Balanced Live Training dataset* that we have used in our work as described in Table 2 in Section 2.3. This experiment has been done primarily with the intent of (i) finding the major differences in the patterns of a phishing URL from its legitimate counterpart, and (ii) finding if there are common patterns that recur in phishing URLs.

These patterns if present will reinforce the classifier which can predict the URL nature prematurely even before performing the conventional time-consuming steps of feature extraction and running the classification algorithm.

Presence of Server Side Extensions: We observed the presence of 170 types of extensions (like program executables and encoded scripts for example exe, pl, cmd, r, js, etc.) in the two types of URLs and report the top six extensions which are more common in phishing URLs with respect to legitimate counterparts. We also report the common sections of the URL these extensions were used in: for example, if these scripts were used more in the ‘query’ part or the ‘path’ section of the URL. The various parts of the URL can be explained by the following example:

```
Original URL: https://video.google.co.uk:80/videoplay?
    docid=-7246927&hl=en#00h02m30s
Parts of the URL:
(a)Protocol: ‘https://’
(b)Subdomain: ‘video’
(c)Domain: ‘google.co.uk’
(d)Port #: ‘80’
(e)Path: ‘videoplay’
(f)Query: ‘docid’
(g)Parameters: ‘-7246927&hl=en’
(h)Fragment: ‘00h02m30s’
```

In Table 7, we explain the observations for the extracted extensions from the two types of URL datasets. The table reports the occurrence frequencies and the URL sections where the extensions occur.

Table 7: File extension Analysis in URLs

Extensions	Legitimate	Phishing
.php	1244 (path, query)	10416 (path, query, param and fragment)
.pl	82 (path, query)	576 (path, query)
.cmd	4 (path)	1157 (path, query)
.r	3 (path)	25 (query)
.x	9 (path)	121 (query)
.dll	11 (path)	98 (path, query)

Inference: These server side extensions are used by attackers to launch an executable, Java and Perl scripts, or dynamic linked library files on the compromised or the victim's system. These files or extensions may act as signs that the URL is a potential phishing link. The ratio analysis further confirms that these attributes set apart a phishing link from a legitimate one. And the use of character n-grams captures these differences in use of extensions as features for analyzing the URLs.

Distribution of Characters and Digits: A sly technique used by the phishers for blacklist evasion is the redirection of the victims to a page hosted on an automatically and randomly generated URL that replicates all the malicious content hosted on a previous phishing URL. This is a technique adopted by phishing kits and very effective in ‘phishing’ unsuspecting users. Moreover, URLs generated by using this technique is also very hard to detect using a blacklist approach as the phishing kit would only generate a new random URL to host the malicious webpage.

The main idea for generating the random webpages can be summarized by the following PHP code [13]:

```
$random=rand(0,100000000000);
$md5=md5("$random");
$base=base64_encode($md5);
$dst=md5("$base");
$src="source";
recursive_copy( $src, $dst );
header("location:$dst");
```

Even reporting such URLs is pretty useless as the users normally report the randomly generated phishing URL locations rather than reporting the fixed entry page. Also reporting an entry page is rendered useless as the users maybe redirected to the random location on clicking it [13].

These new and unique obfuscation techniques used to fool the user into mistaking a phishing URL for a genuine one, perturbs the character distribution of phishing URLs from legitimate URLs.

Calculation of Shannon Entropy: To demonstrate the randomness factor in URLs we use Shannon Entropy as a measure: higher the entropy, higher is the randomness of the instance under consideration. We calculate the entropy measure of each legitimate and phishing URL separately. The Table 8 demonstrates the mean and standard deviation of the entropy measures for the phishing and legitimate URLs in the Balanced Live Training Data.

Table 8: Mean and SD Entropy values for Balanced Training Data

	Legitimate URLs	Phishing URLs
Average	4.082	4.517
Standard Deviation	0.432	0.366

Inference: Thus this can be used as a measure to infer that legitimate URLs are mostly regular in nature and therefore the values are low. This helps predict a correlation of randomness with the URL class. The phishing URLs have a more random nature than their legitimate counterparts. This randomness in URL nature can be used as a potential attribute to predict the URL class effectively.

5. SECURITY ANALYSIS

The phishers may adapt their attack vectors and come up with ways to counter the effectiveness of our classification methods. The robustness of the online learning classifiers lies in the fact that they can adapt with changing strategies of the phisher, which helps them classify the true nature of any URL. The features that we have used are the character N-grams of URLs. So, one possible way in which the phisher may try to work around our classifier is by using domain names which are very similar to the original legitimate domain names, for their phishing websites. For example, ‘Login’ or ‘loGin’ in place of ‘login’. For this reason, we recommend to combine our classifiers with an edit distance feature. In a quick experiment with an edit distance feature, we found that keeping a small list of popular targets (few hundreds) is sufficient to give an accuracy of approximately 85% with batch learning algorithms in WEKA. So we expect that with online learning algorithms, especially AROW, higher accuracies can be obtained. Note that a popular phishing technique of adding strings to a legitimate domain with or without the @ symbol, which is interpreted in a special way by the browser, will fail since the distribution of N-grams is disturbed significantly from that of a legitimate one unless the total length of the added strings is small, which again implies a small edit distance.

6. RELATED RESEARCH

We refer to [15] for a broader perspective on phishing. We now compare our work with previous research on phishing website detection based on URLs since this is the focus of our paper. We also mention related work on the use of character N-grams as features in the malicious email ecosystem.

Phishing URL Detection. Note that we do not use any host-based features, e.g., as used by [20], but still our results are competitive or better. In this sense, our work proves that it is possible to do as good or better in terms of phishing detection using *URL features alone.* The online learning algorithms P, PA, CW, and Logistic Regression with stochastic gradient were used by them. Here, we see that AROW usually outperforms P, PA and CW algorithms. Using the features extracted by [20], researchers conducted unbalanced dataset experiments in [30] with sum and cost metrics, but they do not conduct any live experiment, nor do they compare their work with any baseline. Researchers in [28] report a Logistic Regression algorithm to achieve a reasonably accurate model, but their approach requires *manual* extraction of some of the features, whereas our features and classifiers perform at a higher level of accuracy with completely automatic feature extraction. Our methods outperform also the previous work of [26] on the same datasets, albeit our feature set of N-grams is larger than theirs for $N > 2$. Our results are also competitive with or better than those in [19] and [3]. However, the datasets from the latter two groups could not be provided to us when we requested, due to the limitation of their data-sharing agreements. In [24], the authors detect phishing patterns based on the fundamental characteristics of a phishing website. The authors extract features based on the URL, the website content and the behavioral differences between phishing and legitimate websites. However, this can be quite time consuming, risky and inefficient since visiting the phishing websites for content related feature extraction and evaluation may be harmful.

After this work was completed, we discovered that the AROW algorithm was previously used to study phishing URLs in [18], where the authors show a drop of 1% in accuracy when binary lexical features of [20] and a variety of other hand-selected lexical features are used for detection as opposed to a full feature set consisting of Whois, Team Cymru (network and geolocation) and lexical features. Their accuracy was around 90%. But our N-gram features on larger datasets give significantly higher accuracy and our work studies other dimensions (e.g., live experiment and analysis) that they do not consider. The authors of [25] have proposed a real-time spam URL filtering service which provides filtering of "malicious" URLs by deploying the system online. It uses a linear regression algorithm for fast URL analysis and gets around 91% accuracy with 0.87% false positives. But, as mentioned by the authors, they have their classifier trained on datasets that "lack comprehensive ground truth," since they rely on blacklist feeds and their spam-trap. The authors of the paper [5] propose a large number of features (around 53) belonging to six classes: Lexical, Link Popularity, Webpage Content, DNS, DNS Fluxiness and Network. These are then used to batch classify URLs (32000 "malicious" and 40000 benign) which belong to phishing, spam and legitimate emails, websites and also to Malware producing websites. The diverse features will incur a time penalty and their datasets are smaller, less diverse.

Character N-grams. Character N-grams as features have been investigated for spam *messages* in [16], where Support Vector Machine was used as the only classifier. The proposed method gave a precision of 99.60% with a recall of around 98%. However, we find that a majority of spam messages *do not contain URLs* (only 227 out of 481 contain URLs which is 47.19% only), and we find that the two classes of URLs are quite different. Moreover, in our classification technique, we use a range of online learning algorithms to classify the URLs.

In an unpublished draft paper [10], the author extracts character N-grams from their own private dataset of 250,000 benign and "malicious" URLs each and report the top 25 N-grams (N= 1 to 6). However, they do not define precisely what they mean by the term malicious URLs. They also refrain from performing any sort of classification of the URLs, which we have done in our work. Another work which makes use of *word* N-grams is [21], but their use of N-grams is for building Markov models for domain name generation. The authors' focus is on predicting "malicious" domain names. It would be interesting future work to perhaps integrate their predictor into our models.

7. CONCLUSIONS

In this paper, we performed a comprehensive study of phishing URLs. One of our primary goals was to build fast, robust and accurate URL classifiers for detecting the nature of the URLs. In this work, we use a single class of features that subsumes other lexical features: character based N-grams from URLs and evaluate a range of online learning algorithms. We achieved very high accuracies with the Adaptive Regularization of Weight and Confidence Weighted vectors. The time taken to evaluate the model on the set of 90,000 URLs took 2 seconds for the set of unigrams, 6 seconds for the set of bigrams, and to 30 seconds for the set of all trigrams. The increase in time seems to be reasonable, because even for the set of all trigrams we were able to process 3000 URLs per second. This is more than fast enough to be built into a real-time system, such as a web browser.

We conducted a number of experiments to study the robustness of our features and learning algorithms. We also ran our classifiers on the two types of experimental datasets as well as on extremely balanced and unbalanced datasets in a live URL feed setup to test their robustness. Our classifiers apart from being robust and fast, are also hard to beat: due to the virtue of their online nature, they can quickly learn and update the models with the mutations in the URL formats made by the phisher to annul the detection algorithms used. Thus our online learning classifiers exhibit two important qualities essential for exemplary classification purposes: robustness and speed.

Acknowledgments

This research was funded by the National Science Foundation (NSF) grants DUE 1241772, CNS 1319212 and DGE 1433817. We would also like to thank our colleagues from University of Houston, specially Keith Dyer, Luis Felipe Moraes and Daniel Lee who provided insight and expertise that greatly assisted the research. We would like to thank the reviewers for their comments and insights that greatly improved the manuscript.

8. REFERENCES

[1] Ling spam corpus. http://csmining.org/index.php/ling-spam-datasets.html.

[2] Spam block lists. http://www.joewein.de/sw/blacklist.htm\#use.

[3] A. Blum, B. Wardman, T. Solorio, and G. Warner. Lexical feature based phishing url detection using online learning. In *AISec*, pages 54–60, 2010.

[4] L. Bottou and Y. LeCun. Large scale online learning. *Advances in Neural Information Processing Systems 16, Cambridge, MA: MIT Press*, pages 217 – 224, 2004.

[5] H. Choi, B. B. Zhu, and H. Lee. Detecting malicious web links and identifying their attack types. *USENIX*, 2011.

[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006.

[7] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. *Proceedings of the 22nd Neural Information Processing Systems*, 2009.

[8] H. Daumé III. A course in machine learning: The perceptron. http://ciml.info, 2012.

[9] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. *Proceedings of the 25 th International Conference on Machine Learning*, 2008.

[10] M. Geide. N-gram character sequence analysis of benign vs. malicious domains/urls. http://research.zscaler.com/2010/04/fun-with-n-gram-analysis.html.

[11] B. Gyawali, T. Solorio, M. M. y Gómez, B. Wardman, and G. Warner. Evaluating a semisupervised approach to phishing url identification in a realistic scenario. In *CEAS*, pages 176–183, 2011.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[13] X. Han, N. Kheir, and D. Balzarotti. Phisheye: Live monitoring of sandboxed phishing kits. In *Proceedings of the 23rd ACM conference on Computer and communications security (CCS)*, CCS '16. ACM, October 2016.

[14] J. Hong. The state of phishing attacks. *Commun. ACM*, 55(1):74–81, 2012.

[15] M. Jakobsson and S. Myers. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft.* Wiley-Interscience,, Hoboken, NJ, USA, 2007.

[16] I. Kanaris, K. Kanaris, and E. Stamatatos. Spam detection using character n-grams. *SETN*, 2006.

[17] T. Kiso. Arowpp. https://github.com/tetsuok/arowpp, 2011.

[18] A. Le, A. Markopoulou, and M. Faloutsos. Phishdef: Url names say it all. In *INFOCOM*, pages 191–195, 2011.

[19] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[20] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Learning to detect malicious urls. *ACM TIST*, 2(3):30, 2011.

[21] S. Marchal, J. François, R. State, and T. Engel. Proactive discovery of phishing related domain names. In *Research in Attacks, Intrusions, and Defenses. Springer, vol. 7462, pp.190-209*, Sep 2012.

[22] D. Okanohara. Online learning library. https://code.google.com/p/oll/, 2010.

[23] S. Shaleve-Shwartz. Online learning: Theory, algorithms, and applications. 2007.

[24] T. Thakur and R. Verma. Catching classical and hijack-based phishing attacks. In *International Conference on Information Systems Security*, pages 318–337. Springer, 2014.

[25] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and evolution of a real-time url spam filtering service. In *IEEE Symposium on Security and Privacy*, 2011.

[26] R. Verma and K. Dyer. On the character of phishing urls: Accurate and robust statistical learning classifiers. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 111–122. ACM, 2015.

[27] R. Verma, N. Shashidhar, and N. Hossain. Detecting phishing emails the natural language way. In *European Symposium on Research in Computer Security*, pages 824–841. Springer, 2012.

[28] J. Zhang and Y. Wang. A real-time automatic detection of phishing urls. In *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on*, pages 1212–1216, Dec 2012.

[29] Y. Zhang, J. Hong, and L. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proc. 16th int'l conf. on World Wide Web*, pages 639–648. ACM, 2007.

[30] P. Zhao and S. C. Hoi. Cost-sensitive online active learning with application to malicious url detection. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 919–927. ACM, 2013.