Classifying Spam using URLs

Di Ai
Computer Science
Stanford University
Stanford, CA
diai@stanford.edu

CS 229 Project, Autumn 2018

***Abstract*—This project implements support vector machine and random forest models to create a spam classifier using primarily the url string as features. Because search engines have limited crawl resources, being able to identify a spam url without relying on page content will result in significant resource savings in addition to a reduction of spam for the user. This work implements various features and compares the performance of multiple classifiers in detecting spam.**

***Keywords*—spam, url, classification, search engine**

I. Introduction

Spam is a large and growing problem that is becoming increasingly difficult to tackle at scale. With the proliferation of devices that are connected to the internet, the amount of search queries increases each day, with Google currently processing over 3.5 billion queries per day. There are significant monetary incentives to achieving a high rank in search results, especially for queries with commercial intent like “hotel” or “plane tickets”. As a result, spammers have tried a variety of techniques to rank highly in search results. Search engines have limited resources and cannot crawl every page on the web. Assuming a fixed crawl throughput, for each spam page that is crawled, a page that could have been useful to users will not be crawled. Thus, an algorithm that is able to identify a spam web page before crawling the page would be very useful. Not only would this save precious crawl resources by allowing the crawler to bypass spam urls, but it would also remove the possibility of that spam url from later being served to user queries, thereby improving the quality of search results.

The input to our algorithm is a url string along with metadata such as registered domain holder, first seen date, and traffic. We then use a SVM or Random Forest to output a prediction of spam/not spam.

II. Related work

Previous work on this topic has involved content analysis of the page itself [1]. These typically involve creating features from the html structure of the page, links, and anchor text, such as number of words on page, average word-length, and number of words in title. Other methods involve looking at the amount and percentage of hidden content (not visible to a user) on a page.

Other work attempt to classify web spam into buckets, such as link spam, redirection, cloaking, and keyword stuffing [2]. While splitting spam into more specific buckets will likely lead to improvements in classifier ability, this paper will focus on building a general classifier for all types of spam.

Another approach is to first determine what are important features in terms of ranking in a search engine and then find which features are likely to be used by spammers [3]. The downside to this approach is that it is infeasible to enumerate every ranking element and thus important features may be missed.

While relying on the page content and links increase the amount of data available for spam classification, there are strong motivations for being able to classify spam prior to crawling a page. This paper explores using the url string as the primary feature in spam classification.

III. Dataset and Features

A. Overview

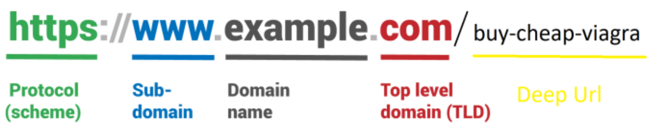
While there are a variety of features that one can use to classify if a web page is spam, this project aims to use only the url and limited metadata information to classify if web pages are spam/not spam. This choice was made for performance reasons, as scraping html from web pages is resource intensive and not useful since the page must have already been crawled. In the context of a search engine, it is often very useful to be able to detect if a given url is spam prior to a page being crawled. This way, urls that are likely to be spam can be deprioritized during crawling and those resources can be used to crawl more useful pages that are less likely to be spam.

B. Data

The dataset consists of 40,000 manually rated urls with 50% labeled as spam and 50% as not spam collected over the course of two months. Additional information that would have been available at crawl time was also included, such as first seen dates, whois contact data, and site traffic. Data is representative of the types of spam that appeared in Google Search in September and October 2018. All numerical features were scaled and standardized to have mean 0 and variance 1. Categorical variables like whois name and tld suffix were one-hot encoded with a minimum cutoff frequency threshold.

C. Features

Each url was broken down into domain, top-level tld, and deep url sections. For example, in the URL http://www.pcwebopedia.com/index.html, the domain name is pcwebopedia.com, the top-level tld is .com, and the deep url is index.html. Any characters preceding the domain name were discarded, such as http://www., except for sub-domains like sub1.pcwebopedia.com/.



Each deep url was split into words by using various characters as delimiters, such as _, ., :, =, ;, and /. The full regex is shown below:

' |\.|\/|\/\/|:|-|_|%|\?|=|;|<|>|~|\$|&|\+'

Words were kept if they were more than 2 characters long, otherwise they were ignored. A multinomial event model was fitted on these words with Laplace smoothing and a dictionary limited to words that appeared with a frequency of at least 0.1% times the size of the dataset. The output probabilities from this classifier was used as a feature.

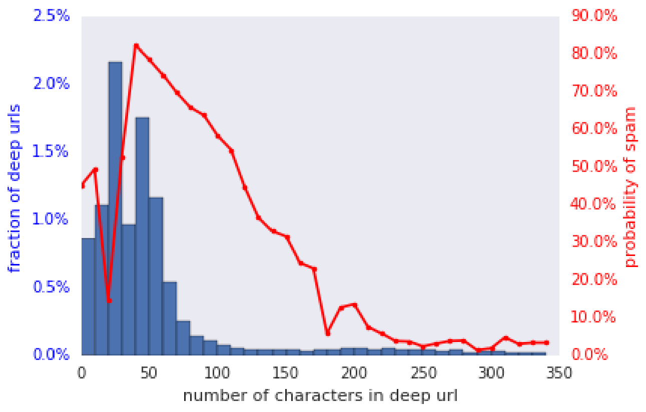
In a multinomial event model, we assume that the way a url is generated is via a random process in which spam/not spam is first determined according to p(y). Then, the spammer composes the url by generating words, x_i, from some multinomial distribution $p(x_i|y)$. All words are assumed to be chosen independently and from the same distribution. Note that the distribution according to which a word is generated does not depend on its position within the url string. The likelihood function and maximum likelihood estimates of the parameters with Laplace smoothing are given below, where |V| is the size of the vocabulary dictionary:

$$\begin{aligned}\mathcal{L}(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) &= \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}) \\ &= \prod_{i=1}^{m} \left(\prod_{j=1}^{n_i} p(x_j^{(i)}|y; \phi_{k|y=0}, \phi_{k|y=1}) \right) p(y^{(i)}; \phi_y).\end{aligned}$$

$$\begin{aligned}\phi_{k|y=1} &= \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i + |V|} \\ \phi_{k|y=0} &= \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} n_i + |V|}.\end{aligned}$$

This model had an accuracy of 83.2% and the top 5 indicative words for spam urls were: 'shgbcz', 'bdouwo1', 'yboyiw5k', 'yulsef', 'qoricksjw'.

Character counts of both domain name and deep url were used as features. The idea here is that longer urls, in general, are more likely to be spam, although this is a weak relationship that doesn't necessarily hold true for this specific dataset, as illustrated by the graph below.



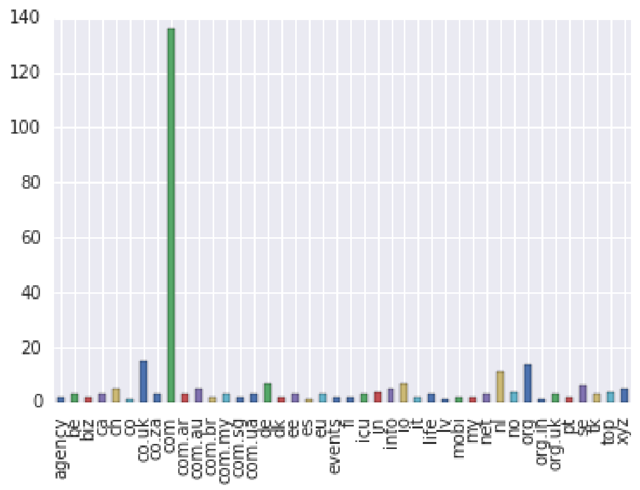
Additional features include the first seen date of the url and the first seen date of the site. This helps to capture any trend in time of various types of spam. For example, during Black Friday and other peak shopping holidays, we might expect higher amounts of spam related to keywords such as iPhone, laptop, or TVs. The difference between the first seen date of the url and the first seen date of the site was also used as a predictor, with the idea being that urls with a large difference may be more likely to be hacked.

The ratio of web traffic to the url vs. the site was used as a predictor. If a url has a lot more traffic than the corresponding site home page, then it indicates that something suspicious is occurring. The url may have been hacked or contain a forum that has been overrun by comment spam. I added +1 to the denominator (site traffic) for each observation to avoid NaN issues from dividing 0 by 0.

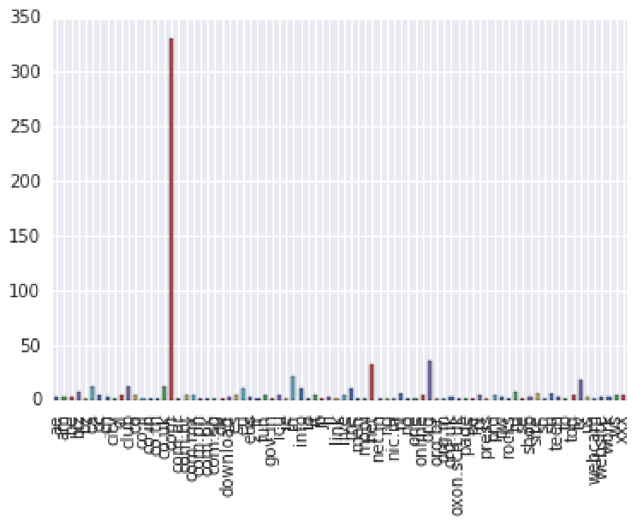
When available, WhoIs data was also obtained for each url consisting of the registrant name, previous DNS and current DNS. A change in DNS would indicate a change in site ownership which could indicate that a site has fallen into the hands of a spammer.

Top-level tld was encoded as a categorical variable with the following distributions for spam and not spam urls. While .com is by far the most common tld in the data, we can see that the relative frequencies of other tlds varies across the spam vs. not spam datasets for a sample of 2000 spam and not spam urls.

TLD distribution for Spam Urls



TLD distribution for Not Spam Urls



IV. METHODS

SVM with both linear and Gaussian kernels (RBF) were fitted on the training data. The support vector machine attempts to find a separating hyperplane that separates the data with the maximum geometric margin. However, since not every dataset is linearly separable, we introduce slack variables to allow the model to make some mistakes with a cost C. This results in the following optimization problem.

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1,\ldots,m \\ & \xi_i \geq 0, \quad i = 1,\ldots,m. \end{aligned}$$

Compared to the linear SVM, RBF SVM allows for a richer set of features as the Gaussian kernel represents an infinite dimensional mapping. While this may seem computationally expensive, by using the kernel trick, we are able to avoid blowing up the required time complexity. Specifically, we replace the dot product between x's with the Gaussian kernel.

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}\right)^T x + b \\ &= \sum_{i=1}^{m}\alpha_i y^{(i)}\langle x^{(i)}, x\rangle + b. \end{aligned}$$

$$K(x,z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)$$

We also use random forest to fit the data. Random forests are an ensemble learning method constructed from an agglomeration of decision trees. A decision tree makes binary splits of the data using the features and cutoff points that leads to greatest possible reduction in the loss in a greedy manner. Random forests aggregate a large number of decision trees using a method called bagging (bootstrap aggregation) and outputs the class that is the mode of the individual trees. The bootstrap method is first used to create random sub-samples of the dataset with replacement. Then, decision trees are fit on each sub-sample and their output averaged over all decision trees (for classification problems, you take the mode instead of average). Typically, random forests limit both the max depth of the trees and the number of features available for prediction for each tree to improve performance. This leads to greater variance reduction in the overall estimator because the pairwise correlation, ρ, between each tree is lower.

$$\text{Var}(\hat{T}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

We used the Gini loss, which is shown below.

$$G = \sum_{k=1}^{K}\hat{p}_{mk}(1-\hat{p}_{mk})$$

V. RESULTS

10% of the dataset (2000 examples with roughly 50% spam/not spam) was set aside as the test dataset. The primary evaluation metric is accuracy. However, in real world implementations the more important metric is probably precision, since failing to crawl a legitimate page results in significant loss to the business or site owner. On the other hand, crawling a spam page, while not ideal, is not as severe as a mistake, since the spam can still be suppressed or removed at serving time. Results are shown below for a variety of models and parameters.

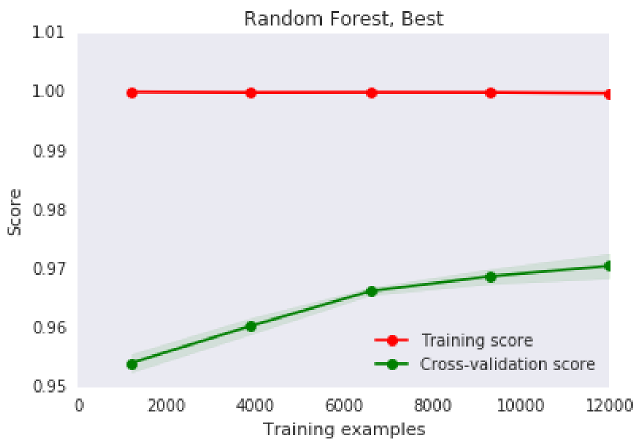
TABLE I. RESULTS

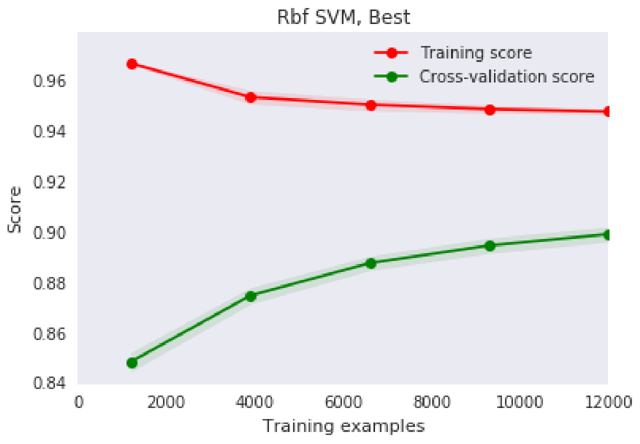
Model	Metrics		
	Cutoff*	***Train Accuracy***	***Test Accuracy***
Linear SVM, Cost = 1	50	88.70%	88.00%
Linear SVM, Cost = 10	50	89.40%	89.20%
Rbf SVM, Cost = 1, Gamma = 1	50	94.50%	91.15%
Rbf SVM, Cost = 1, Gamma = 0.01	50	87.37%	87.05%
Rbf SVM, Cost = 1, Gamma = 100	50	99.30%	81.45%
Rbf SVM, Cost = 10, Gamma = 1	50	96.81%	89.85%
Rbf SVM, Cost = 100, Gamma = 100	50	99.70%	81.90%
Rbf SVM, Cost = 1, Gamma = 1	5	94.41%	90.40%
Linear SVM, Cost = 1	5	89.55%	88.60%
Random Forest	50	99.70%	96.85%
Random Forest, Max Depth = 10	50	95.42%	95.20%
Random Forest	5	99.79%	96.50%
Random Forest, Max Depth = 10	5	94.36%	94.50%
Random Forest, Best	50	99.98%	97.25%

* Categorical variables were one hot encoded with a cutoff frequency threshold.

Varying the cutoff frequency threshold did not seem to have a large effect on the accuracy of the models. Any categorical variable that was one-hot encoded had the value dropped if it did not occur at least the cutoff frequency threshold number of times in the training dataset. Increasing the threshold from 5 to 50 resulted in the number of feature variables dropping from 406 to 81. While it is surprising that reducing the number of features by 5x does not degrade model performance, this may be due to the size of the test data, which was one-tenth the size of the training data. A combination of not enough training data (on 5 examples) and low prevalence frequency in the test dataset likely reduced the usefulness of the majority of categorical features.

The two models highlighted in red represent the models with the best, tuned hyperparameters as found by grid search for SVM and random search for Random Forest. The optimal SVM had hyperparameters of cost = 1 and gamma = 1. The optimal Random Forest model had hyperparameters of min samples leaf = 1, max features = sqrt(number of features), min samples split = 2, number of trees = 1400, bootstrap = False, and max depth = 3. Because of how Random Forests are constructed, they tend to be resistant to overfitting. However, the plot of test error vs. training error over training time seems to suggest that the model has high variance and could benefit from more training data. This is true for the Rbf SVM as well. My hypothesis is that given the vast variety of spam, 18,000 training examples (12,000 in the plots below which use 3-fold cross-validation) is simply not a large enough sample to adequately cover all possible variations of spam.





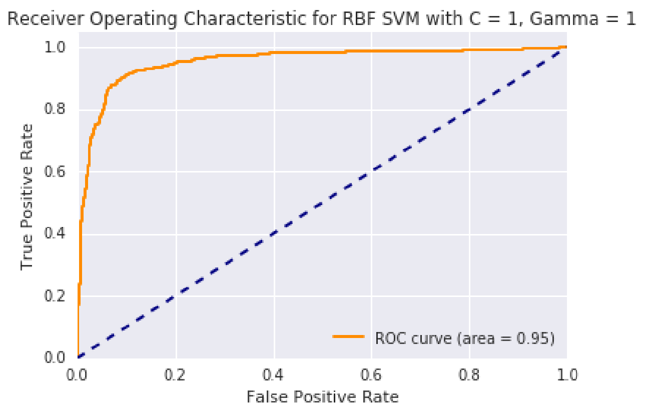
The confusion matrices for the best Rbf SVM and Random Forest models show that both ended up with a fairly balanced false positive rate and false negative rate, making roughly equal number of mistakes on both sides. The ROC curve clearly favors the Random Forest model over the Rbf SVM. However, for a model that would be used in production, we would likely increase the cutoff threshold for a positive label from a probability of 0.5 to something higher to reduce the false positive rate. A false positive, i.e. predicting a url is spam when it is in fact not spam, is a more serious error than a false negative, i.e. predicting a url is not spam when it is in fact spam. False positives create more damage to a search engine's reputation, especially if the false positive is on a prominent site, and harm real businesses, businesses that a search engine relies upon in a symbiotic relationship. In addition, false negatives at this stage can also be caught later at indexing or serving time. Hence, we would probably want to trade off a lower recall and higher false negative rate in exchange for a lower false positive rate.

TABLE II. CONFUSION MATRIX

N = 2000	RBF SVM, Best	
	Predicted Not Spam	*Predicted Spam*
Actual Not Spam	903	90
Actual Spam	87	920

TABLE III. CONFUSION MATRIX

N = 2000	Random Forest, Best	
	Predicted Not Spam	*Predicted Spam*
Actual Not Spam	912	31
Actual Spam	27	1030



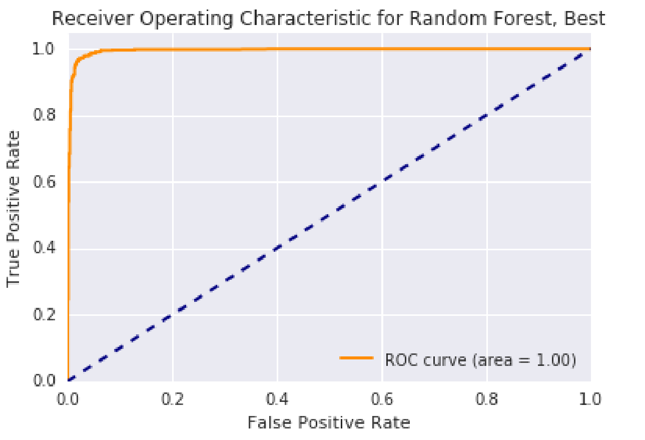


TABLE IV. ABLATION STUDY

Model	Metrics			
	Accuracy	*Precision*	*Recall*	*AUC*
SVM RBF	91.2%	91.1%	91.4%	91.1%
Remove Naïve Bayes Prob feature	87.5%	84.6%	91.2%	87.5%
Remove url length feature	91.0%	91.5%	90.6%	91.0%
Remove diff between url and site first seen date	90.5%	91.0%	90.0%	90.5%
Remove ratio of url traffic to site traffic	91.2%	91.3%	91.2%	91.2%

An ablation study was also performed to see which feature had the most impact on the model. Note that only single features were removed and the effects are not cumulative. We can see that the naïve bayes output probabilities (see Features section for description of how these were calculated) were the most important feature. This is corroborated by the feature importance ranking from the Random Forest model, shown below. Surprisingly, the suffix_net feature was the second most important feature, indicating that spammers favor .net domains significantly differently than non-spammers.

```
Feature ranking:
1. feature 8 (0.334062) <- naive_bayes_probs
2. feature 10 (0.126054) <- suffix_net
3. feature 3 (0.062638) <- site_traffic
4. feature 0 (0.050510) <- url_first_seen
5. feature 5 (0.050321) <- url_no_host_length
6. feature 1 (0.046668) <- site_first_seen
7. feature 6 (0.043675) <- domain_length
8. feature 4 (0.043049) <- diff_url_site_first_seen
```

VI. CONCLUSION

We have presented a variety of models in an attempt to identify spam urls prior to crawling the page. Random Forest with Gini loss performed the best out of the models tested, with 97.25% accuracy on the test data. Since spammers frequently alter their techniques, we expect model performance to degrade over time if the model is not retrained with new data. Gathering more data across a variety of spam types and time periods would help the model generalize to more types of spam. Training specific models for each spam type or geography/language would also likely result in performance gains.

The most important feature was the naïve bayes output probabilities, which was created by splitting the deep url portion into words and fitting a multinomial event model with Laplace smoothing. Since there is likely significant headroom to improve this feature, future work would involve more sophisticated feature extraction methods on the deep url portion. For example, using a bag of words model or n-grams would likely result in significant improvements. In addition, deep learning methods such as neural nets could also be explored given more time and computational power.

REFERENCES

[1] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. Detecting spam web pages through content analysis. In Proceedings of the 15th international conference on World Wide Web (WWW '06). ACM, New York, NY, USA, 83-92. DOI: https://doi.org/10.1145/1135777.1135794

[2] Gyongyi, Zoltan and Garcia-Molina, Hector (2005) Web Spam Taxonomy. In: First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb 2005), May 10-14, 2005, Chiba, Japan.

[3] Egele, M., Kolbitsch, C. & Platzer, C. J Comput Virol (2011) 7: 51. https://doi.org/10.1007/s11416-009-0132-6

[4] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of machine learning research 12.Oct (2011): 2825-2830.