A Domain-Agnostic Approach to Spam-URL Detection via Redirects

Heeyoung Kwon Mirza Basim Baig Leman Akoglu*

Stony Brook University, Computer Science, {heekwon,mbaig}@cs.stonybrook.edu
*Carnegie Mellon University, H. John Heinz III College, lakoglu@cs.cmu.edu

Abstract. Web services like social networks, video streaming sites, etc. draw numerous viewers daily. This popularity makes them attractive targets for spammers to distribute hyperlinks to malicious content. In this work we propose a new approach for detecting spam URLs on the Web. Our key idea is to leverage the properties of URL redirections widely deployed by spammers. We combine the redirect chains into a *redirection graph* that reveals the underlying infrastructure in which the spammers operate, and design our method to build on key characteristics closely associated with the modus operandi of the spammers. Different from previous work, our approach exhibits three key characteristics; (1) *domain-independence*, which enables it to generalize across different Web services, (2) *adversarial robustness*, which incurs difficulty, risk, or cost on spammers to evade as it is tightly coupled with their operational behavior, and (3) *semi-supervised detection*, which uses only a few labeled examples to produce competitive results thanks to its effective usage of the redundancy in spammers' operations. Evaluation on large Twitter datasets shows that we achieve above 0.96 recall and 0.70 precision with false positive rate below 0.07 with only 1% of labeled data.

1 Introduction

Web services are ubiquitous: social networks (e.g. Facebook, Twitter), review sites (e.g., Yelp, Amazon), video streaming sites (e.g. YouTube, Hulu), blogs, forums, etc. draw billions of viewers daily. The widespread adoption of these services makes them attractive for spammers to distribute harmful content (scam, phishing, malware, etc.) through links they post on these sites to such content. As a result, detecting and filtering malicious content effectively becomes crucial for the quality and trustiness of the Web.

IP blacklisting—a popular solution for social network operators and URL shortening services—has been found to provide false positive rates ranging between 0.5 to 26.9%, and false negative rates between 40.2 to 98.1% [16, 17], which is quite inaccurate. Blacklisting is also quite slow to keep up with the speed and scale that Web services are being consumed today. Alternative solutions focus on identifying suspicious *accounts* operated by spammers that behave in automated or fraudulent ways [2, 8, 18]. These, however, have limited ability to detect spam disributed through *compromised* accounts. In fact, 97% of accounts participating in spam campaigns on Facebook [4] and 86% on Twitter [5] have been found

to involve compromised accounts. Moreover, they incur detection delays as they require a history of mis-activity committed by an account. Thus, it is essential to build solutions that can make fine-grained, i.e., *URL-level decisions*, which could enable services to filter individual posts rather than shutting down user accounts. It is also desirable to have solutions that are *generalizable* to different kinds of Web services, i.e. that spot spam URLs regardless of the context, platform, or domain in which they appear.

We propose a general and robust solution for detecting malicious URLs. Our key realization is the widespread usage of *redirect chains* by spammers to distribute spam on the Web [6, 21]. Our main contributions are as follows.

- We develop a new graph-based approach for spotting malicious URLs that appear on the Web. Our method leverages the underlying redirection network used by the spammers. In particular, we build a graph, called the Redirect Chain Graph (RCG), based on the redirect paths of the URLs and use its structural properties to design and extract indicative features of spam.
- Our features fall under three main groups (resource sharing, heterogeneity, and flexibility) and capture the very nature of spammers' operational behaviors. These are hard to alter by the spammers without incurring monetary or management cost. As such, our features have higher *adversarially robustness*.
- Our approach relies solely on the redirection infrastructure and does not use any domain-specific information, which makes it *context/content-agnostic*. As such, it can detect spam URLs in various domains, including URLs shared on any online site, URLs returned as online search results, and so on.
- In a fully supervised setting, our approach performs extremely well.When compared to context-aware supervised detection that uses user account and post content features, our context-free features perform equally well, despite ignoring all domain-specific information.
- Finally, we propose a *semi-supervised* method, designed for more realistic scenarios where labeled data is scarce. By carefully exploiting the redundancy present in spammers' infrastructures, our proposed method requires only a few labeled examples to achieve desirably high performance to be applicable in the real-world.

In contrast, numerous existing methods, such as [9, 11, 12, 19, 20], either (*i*) utilize easy-to-evade information (low robustness), (*ii*) rely on context-dependent information (low generality), and/or (*iii*) require large collections of labeled data for training (low applicability in practice).

2 Redirection Infrastructure

Many studies have shown the pervasive use of redirects by spammers [6, 21, 1, 9, 11]. In this section, we introduce the interconnected architecture of redirect chains, which provides the main motivation for our graph-based approach.

Definition 1 (Redirect Chain). *A redirect chain C consists of an ordered set of URLs, $C = \{U_1, U_2, \ldots, U_l\}$, starting with an initial URL U_1, followed by URLs automatically and conditionally redirected in a sequence, and landing on a final URL U_l. $l = |C|$ denotes the length of the chain.*

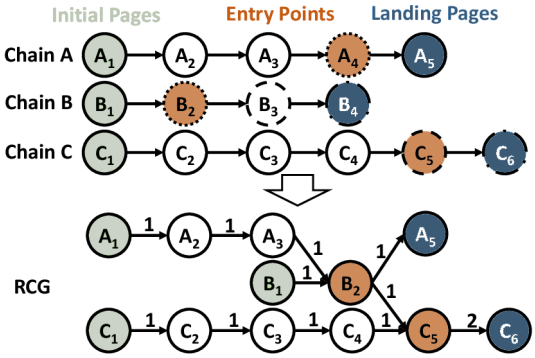

Fig. 1. 3 example redirect chains and their RCG. Chains may contain the same URL(s) (e.g., A_4 & B_2, B_3 & C_5, B_4 & C_6), yielding the interconnected network RCG.

Initial URL (often shortened by e.g. `bit.ly`) is the one displayed to users on a site, whereas landing page is where the user ends up after clicking the initial URL (cf. Figure 1).

As data preprocessing, we group the domain names of the URLs that appear on the same IP. For example, if `http://123.com/hi.html`, `http://xyz.com/hi.html`, and `http://xyz.com/ hello.html` are all co-located at the same-IP address, then we replace the first two URLs with `http://[123.com,xyz.com]/hi.html` and the third with `http://[123.com,xyz.com]/hello.html`. This grouping helps us unify malicious URLs that use several domain names so as to bypass blacklisting. Moreover, a (grouped) URL may be located at multiple different IPs, a list of which we also store. As such, each URL is associated with a list of domain names as well as a list of IPs.

Our key motivation for a graph-based solution is due to the following observation: (malicious) redirect chains deployed by spam campaigns contain several URLs in common, i.e. *shared* across chains, creating a network structure as shown in Figure 1. In other words, the (malicious) redirection infrastructure of spammers is highly *inter-connected*.

Provided that the redirect chains are likely to share several URLs, it is beneficial to study them collectively, rather than individually. As such, we merge the redirect chains of URLs[1] to create the Redirect Chain Graph (RCG).

Definition 2 (Redirect Chain Graph (RCG)). *Given a set of redirect chains $\mathcal{C} = \{C_1, \ldots, C_m\}$, we decompose each chain $C_i = \{U_{i,1}, \ldots, U_{i,l_i}\}$ into a set of directed edges between consecutive URLs, $E_i = \{e(U_{i,j}, U_{i,j+1}) \mid j \in \{1, \ldots, l_i - 1\}\}$. The RCG $= (V, E)$ then consists of all the edges across chains, where $E = \biguplus E_i$ and $V = \bigcup C_i$ for $1 \leq i \leq m$.*

Note that the RCG nodes are the unique set of URLs across all chains, whereas the edges are allowed to reappear (hence the multiset addition $\biguplus$). As such, RCG is a directed and weighted graph, where edge weights depict the

[1] Note that what is posted on a Web service are the initial URLs. We run a crawler to go through the redirects to extract the chains.

number of chains that a redirection step appears in. Notice for example the edge weight 2 in Figure 1.

Finally we introduce the *entry-point URLs*. Those are the nodes with large in-weight in the RCG, considered as "directors"—they are central pages that aggregate user traffic and direct them to one of several malicious pages (sort of routers). As such, entry-points are critical in functionality but hard to identify without aggregate graph analysis—the entry point does not serve the actual spam, as such it is more difficult to spot and shut down. We characterize each chain by its entry-point URL, as defined below.

Definition 3 (Entry-point URL). *Given a redirect chain* $C = \{U_1, U_2, \ldots, U_l\}$*, let* w_j *denote URL* j*'s in-weight in the RCG. The entry-point of* C *is the URL* U_k *with the largest in-weight, where* $w_k = \max\{w_1, \ldots, w_l\}$*.*

3 Feature Design using Redirects

There exists a vast body of work that use information derived from the user account that a URL originated from, the URL itself or its page content [8, 9, 13, 20]. We choose not to use such information for the reasons we discuss below.

Rationale to exclude account and URL&page-content information: First reason is to ensure a general solution. Such features are derived from meta data on the specific site the URL appears in (e.g., number of followers). Relying on contextual information would make it hard to cross Web service boundaries due to potentially disparate contextual information across sites. Second, context-aware solutions require personally identifiable information from user accounts, which may not be desirable due to privacy concerns.

As for content, spammers can use feedback from classifiers to fine-tune their URLs and page content in an attempt to evade detection by the classifiers, e.g., by spoofing sufficient benign features with high weights as studied in adversarial classification [3, 10]. For example, they can avoid using spam terms, adjust URL length and character distribution, and modify links and plugins to imitate non-spam pages and URLs, while remaining sufficiently effective in eliciting response from the target users.

Our features (Table 1) fall into three main categories, characterizing spammer operations that reflect (1) shared resources, (2) heterogeneity, and (3) flexibility.

1) Shared resources-driven features. Spammers would ideally deliver each copy of malicious content through a dedicated independent channel, such that if a server fails or is shut down, it has minimal effect. Avoiding to reuse components in their infrastructure (domain names, servers, etc.), however, would increase their costs and limit profits. As such, spammers often reuse their underlying hosting infrastructure for significant periods [1, 13].

The first type of sharing occurs due to the same URLs being reused across different redirect chains. As discussed in §2, nodes with large in-weight in the RCG (e.g., the entry points) are those URLs that are reused to route traffic for many redirect paths. As such, for each given URL[2], we identify the connected

[2] Note that the given URLs are the observed ones posted on the Web, also referred to as the initial URLs in this work.

component of the RCG it resides in and extract features based on structural graph properties, e.g., in-weight of its entry point URL, average in-degree and in-weight of nodes in its chain, density of its RCG component, etc. We also treat the RCG component as a tree, rooted or "hung" at the entry point URL. This tree is obtained by a breadth-first search traversal of the RCG component starting at the entry-URL. Intuitively, a small number of unique entry points in a large RCG is suspicious; implying a few URLs shared among many chains. Tree-based features such as level width and horizontal imbalance capture this, as few entry points cause large fan-out.

A second type of sharing occurs due to the same servers hosting many different domain names. To evade and stay ahead of domain blacklisting, spammers run through many domain names. To reduce operating costs, they host them on the same server (IP address), all serving the same malicious content. We leverage this domain co-location property based on domain counts both in landing URLs of the RCG component as well as in all the URLs in the redirect chain of a URL.

2) Heterogeneity-driven features: The operational infrastructure of spammers consists of a variety of heterogeneous agents, including various compromised servers and bot machines from various geo-locations, besides their own hosting servers. This kind of heterogeneity arises naturally and is crucial for their operations. First, it would require high maintenance to ensure all compromised machines are of a single type or all reside in close geo-locations. Moreover it would be risky if everything resided on one machine or all machines were at the same geo-location, as the infrastructure would have a few failure points. Based on this insight, we design features that quantify infrastructure heterogeneity.

These features mainly leverage geo-spatial and domain name heterogeneity. For example, given the sequence of URLs in a redirect chain, we quantify the total distance in km's traversed. Similarly, we count the number of transfers between different continents, countries, and IPs. We also count the cross-domain hops—contrary to a hodgepodge of IPs and domain names in spam redirect chains, benign sites have the opposite incentive to keep visitors within their own domain. Compromised IPs or sites would also come from various kinds of top-level domains (TLDs), such as `.edu`, `.org`, `.com`, etc., therefore we also keep a count of transfers between different TLDs.

3) Flexibility-driven features: Finally, we derive features from operational properties that allow spammers flexibility, through which their maintenance overhead or expenses are reduced.

To have the advantage of luring as many users as possible, spammers use multiple different initial URLs (even though they redirect to the same malicious content) to make their posts look different. We capture this by keeping count of the initial URLs in the RCG component of a given URL, as well as the total number of domain names that they host.

Using multiple landing URLs (serving the same content), on the other hand, provides redundancy; if a landing page goes down, others can still distribute malicious content. Another way that spammers achieve redundancy is by having copies of the same URLs across multiple different IPs, which we capture through features associated with the number of IP addresses that URLs appear in.

Table 1. Features introduced (3 categories). RC: redirect chain, CC: connected component of RCG a RC resides in. TREE: BFS-tree of CC, rooted at entry-URL. TLD: top-level domain. Node degrees & edge weights are based on RCG.

Feature Name	**Description**
Shared resources-driven (17 features)	
EntryURLiw	In-weight (freq.) of entry-point URL
EntryURLid	In-degree of entry-point URL
AvgURLiw	Mean in-weight of URLs in RC
AvgURLid	Mean in-degree of URLs in RC
ChainWeight	Total weight of edges in RC
CCsize	Number of nodes in CC
CCdensity	Edge density of CC
MaxRCLen	Max. length of RCs in CC
MinRCLen	Min. length of RCs in CC
TreeHeight	Height of TREE (root: entry-URL)
MaxLevelWidth	Max. node count at TREE levels
ImbalanceH	Horizontal imbalance of TREE
ImbalanceV	Vertical imbalance of TREE
MaxLdURLDom	Max. domain count of CC landing URLs
AvgLdURLDom	Mean domain count of CC landing URLs
MaxURLDom	Max domain name count per URL in RC
AvgURLDom	Mean domain name count per URL in RC
Heterogeneity-driven (12 features)	
GeoDist	Total geo-distance (km's) of hops in RC
MaxGeoDist	Max. geo-distance (km's) across hops in RC
XContinentHops	Number of cross-continent hops in RC
CntContinent	Number of unique continents in RC
XCountryHops	Number of cross-country hops in RC
CntCountry	Number of unique countries in RC
XIPHops	Number of cross-IP hops in RC
CntIP	Number of unique IPs in RC
XDomainHops	Number of cross-domain hops in RC
CntDomain	Number of unique domains in RC
XTLD	Number of cross-TLD hops in RC
CntTLD	Number of unique TLDs in RC
Flexibility-driven (10 features)	
ChainLen	Length (#URLs) of RC
EntryURLDist	Distance from initial to entry URL in RC
CntInitURL	Number of initial URLs in RCG
CntInitURLDom	Total domain name count in initial URLs
CntLdURL	Number of final landing URLs in RCG
MaxIPperURL	Max. IP count each URL in RC appears in
AvgIPperURL	Mean IP count each URL in RC appears in
MaxIPperLdURL	Max. IP count landing URLs in CC appear in
AvgIPperLdURL	Mean IP count landing URLs in CC appear in
RatioCheapTLD	Fraction of non-.com/.mil/etc. URLs in RC

Using long redirect chains helps with dynamicity and selectivity, which is hard to evade for spammers, if they want to be flexible in how they replace machines and how they choose who to spam. Specifically, a series of redirects provides them with the flexibility to modify intermediate steps (plug-in & plug-out), as well as the flexibility to hide malicious, "bullet-proof" landing URLs behind layers of redirection. The location of entry point URLs also plays a key role—since these pages have to conditionally redirect visitors to different landing URLs, suspicious entry point URLs are often located early in the chains.

Another case of flexibility is related to top-level domain (TLD) names. Spammers tend not to invest on trustworthy but costly TLDs such as `.com` and `.net`, or try to compromise often bullet-proof TLDs such as `.mil` and `.gov`—especially given that most URLs are only for redirection purposes and not for delivering content. As a result, they resort to acquiring or attacking cheap TLDs.

4 Spam Detection

After crawling the redirect chains, constructing the RCG, and extracting for each URL in the dataset the 39 features as listed in Table 1, our next step is spam detection. In this work, we study both supervised and semi-supervised detection, with a note that the latter presents a more realistic scenario.

Supervised Detection. When a large body of labeled URLs is available, one can build classifiers. In this work, we analyze the performance of our feature categories and characterize the most discriminative ones. In addition, we extract context-based features from user accounts and keywords appearing alongside the URLs to build context-aware classifiers, which we compare to our models.

Note that acquiring labels for each URL is quite time consuming, as annotators need to set up virtual sandboxes and analyze the URL, landing page content, behaviors a click triggers in their system, etc. Moreover, since spam is rare as compared to normal URL traffic, a reasonably large number of URLs needs to be labeled to ensure representative amount of spam labels in the training data.

Semi-supervised Detection. Due to the challenges with supervised detection, we design an approach that utilizes only a small set of labeled examples. Our method achieves comparable performance to fully supervised methods. As such, it is both applicable under the most realistic scenarios where labeled data is scarce as well as desirably effective in detecting spam.

In particular, we leverage the user–URL graph to formulate the problem as a network-based classification task, which we solve using label propagation based inference. More formally, we consider the bipartite graph $G = (N, E)$ in which n user nodes $U = \{u_1, \ldots, u_n\}$ are connected to m URL nodes $V = \{v_1, \ldots, v_m\}$, $N = U \cup V$, through 'post' relations in E. To define a classification task on this network, we utilize pairwise Markov Random Fields (MRFs) [7]. An MRF model consists of an undirected graph where each node i is associated with a random variable Y_i that can be in one of a finite number of states (i.e., class labels). In our case, the domain of labels for URLs is $\mathcal{L}_V = \{spam, benign\}$ and it is $\mathcal{L}_U = \{spammer, non\text{-}spammer\}$ for users. In pairwise MRFs, the label of a node is assumed to be dependent only on its neighbors and independent of other nodes in the graph. As such, the joint probability of labels is written as a product of individual and pairwise factors, respectively parameterized over the nodes and the edges;

$$P(\mathbf{y}) = \frac{1}{Z} \prod_{Y_i \in N} \phi_i(y_i) \prod_{(Y_i, Y_j) \in E} \psi_{ij}(y_i, y_j) \tag{1}$$

where $\mathbf{y}$ denotes an assignment of labels to all nodes, and y_i refers to node i's assigned label. Individual factors $\phi : \mathcal{L} \rightarrow \mathbb{R}^+$ are called *prior* potentials, and represent class probabilities for each node initialized based on prior knowledge. Pairwise factors $\psi : \mathcal{L}_U \times \mathcal{L}_V \rightarrow \mathbb{R}^+$ are called *compatibility* potentials, and capture the likelihood of a node labeled y_i to be connected to a node with y_j.

As we consider a semi-supervised setting, only a small set of the URL labels is available. For the known spam URLs we set the priors as $\phi_i(spam) = 1 - \epsilon$ and $\phi_i(benign) = \epsilon$, and vice versa for the known benign URLs. To set the priors

for the unknown URLs, we learn a classifier using the available labeled data and employ it to assign class probabilities, i.e. priors, to the unknown URLs in the graph. For the users we set unbiased priors, i.e., $\phi_i(spammer) = 0.5$ and $\phi_i(non\text{-}spammer) = 0.5$, as we do not want to rely on any context-specific information (e.g., profile data such as ratio of followers to followees) to estimate such priors.

On the other hand, we instantiate the compatibility potentials so as to enforce homophily among connected nodes. Homophily captures the insight that URLs posted by spammers are spam and those shared by regular users are benign, with high probability, where ψ_{ij}'s are set as follows.

ψ_{ij}	**URLs**	
Users	*spam*	*benign*
spammer	$1-\epsilon$	ϵ
non-spammer	ϵ	$1-\epsilon$

We note that ϵ's in ϕ_i's for URLs with known labels account for the uncertainty in the labels associated with annotator agreement. ϵ's in ψ_{ij}'s capture the slight probability that non-spammers unknowingly can post spam URLs (e.g., retweet) and that spammers can post benign URLs for camouflage.

Provided the model parameters, the classification task is to infer the best assignment $\mathbf{y}$ to the nodes such that the joint probability in Eq. (1) is maximized. This is a combinatorially hard problem that is intractable for large graphs. Therefore, we use an approximate inference algorithm called Loopy Belief Propagation (LBP) [22]. LBP is an iterative algorithm where connected nodes exchange messages. A message m_{ij} captures the *belief* of i about j, specifically the probability distribution over the labels of j. Intuitively, it is what i 'believes' j's label probabilities are, given the current label distribution and the priors of i. The key idea is that after certain number of iterations of message passes between the nodes, the 'conversations' likely come to a consensus, which determines the marginal class probabilities of all the unknown variables. Although convergence is not theoretically guaranteed, LBP converges quickly in practice [15].

5 Experiments

5.1 Data Description

In this work we detect spam links posted on Twitter. Using the Twitter Streaming API, we collected 15,828,532 Twitter posts by 1,080,466 unique users during a period from May 2–September 10, 2014 . This interval captures major world events such as the World Cup and the ongoing search for the Malaysia Airlines Flight 370. Those serve as attractive means to spread spam, where e.g., users are lured to click a malicious link that supposedly points to a video that shows the four goals that Germany scored against Brazil in six minutes during the semi-finals, but instead triggers a drive-by-download exploit. We identified 3,871,911 (initial) URLs from 3,764,395 (≈24%) of the posts that contained links, i.e., a small fraction of posts contained multiple URLs.

We built a crawler and extracted the redirection chains for all the URLs. The chain lengths vary from 1 to 46, with more than 99% being less than 6. We

combined the redirect chains into RCG, a unified (weighted, directed) graph, which contains 4,874,256 nodes and 3,839,633 edges.

To construct a labeled URL set, we used a crawler to first identify a set of suspended Twitter users. In particular, if a user profile page input to our crawler automatically redirected to `http://www.twitter.com/suspended`, we label the account as a malicious one. This provided us with 88,147 suspended users. After removing these, we sampled another 1,000 users for human labeling. Five annotators were provided with the links to the profile pages of these users. Each annotator labeled each user by manually analyzing their tweets, number of followers/followees, temporal behavior, etc. At the end, a user is labeled by majority voting, which provided us with 216 spam users and 784 non-spammers.

We labeled URLs using the labeled users, where URLs inherit the majority label of the users who posted them (labeling URLs through users posting them is extremely pure: 99.06% have majority fraction 100%, i.e., all spammer or none). As a result, we obtained 459,822 labeled URLs, out of which 191,726 are spam.

For supervised detection, we built a balanced dataset (50% spam) by considering the 191,726 chains from the unique spam URLs, and randomly sampled the same number of URLs from those labeled as benign for a total of 383,452 chains. For semi-supervised detection, where we leverage the user–URL bipartite graph, we constructed a graph with 784 users from each class and all the URLs shared by those users for a total of 315,120 ground-truth URLs, with 16% consisting of spam. We experimented with 1% or 5% of this labeled set as input to our semi-supervised approach. Note that our method not only leverages a much smaller labeled set, but also works in an imbalanced setting as in practice.

5.2 Detection Results

We use two metrics to compare the detection methods; (1) average precision, which is the area under the precision–recall (PR) plot, denoted as AP, and (2) area under the ROC curve (false-positive vs. true-positive rate), denoted as AUC.

Supervised Detection. Our experiments with linear SVM, Logistic Regression, and Decision Tree (DT) show that non-linear DT significantly outperforms both. This suggests that the decision boundary of our task is complex. We use the DT model in the remaining supervised detection experiments.

Feature contribution analysis. To investigate the importance of individual features, we quantify their discriminative power. In particular, we use the sum of information gains weighted by the number of samples split by each feature at the internal tree nodes [14] based on the DT model trained on the entire dataset.

Figure 2 shows the ranking of features by the aforementioned importance score. We note that (*i*) the GeoDist feature is considerably the most informative one, and that the scores drop quickly. This suggests that in practice only a small subset of all the features could be enough to build accurate models. We also notice that (*ii*) the majority (5/10, 11/20) of the informative ones are from the shared-resources-driven features (green bars), which are mainly derived based on the RCG structure. Moreover, (*iii*) the top features come from a mix of all three feature categories, suggesting that they carry non-redundant information.

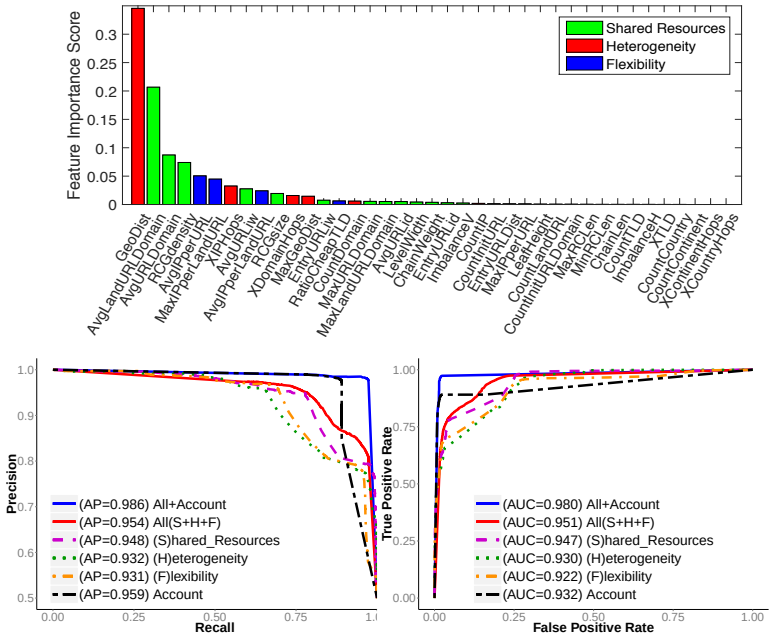

Fig. 3. Supervised detection. Context-free features achieve competitive performance.

In Figure 3 we demonstrate the performances achieved by individual feature categories. In agreement with observation (*ii*) above, (S)hared-resources-driven features perform slightly better than other categories. In addition, potentially due to (*iii*) that different feature groups carry non-redundant signals, using all the features holistically (from all S+H+F categories) yields the best result.

Context-free vs. Context-aware detection. Next we ask: how do context-driven features perform? To investigate this question, we build models (a) based solely on context-based features and (b) integrating them with our original set.

The context-based features are mainly derived from the account that posted the URL and the post content, including the account's age, number of hashtags, number of mentioned other users, the follower–followee ratio, total number of posts, fraction of posts containing a URL, and the keywords used in the posts (including hashtags and @mentions).

In Figure 3, we observe that the context-based model (labeled Account) is slightly better than our model with All features in terms of AP and slightly worse with respect to AUC. We conclude that our features are equally discriminative, even when no domain-specific or potentially missing, hidden, or private information is used.

Semi-supervised Detection. Under this setting, we randomly sample 1% or 5% of the URLs and reveal their labels (results are averaged over 10 runs). We then under-sample the majority class to obtain a balanced dataset, on which we train a classifier to assign class priors to the unlabeled URLs.

Semi-supervised results are given in Figure 4. First, we investigate the performance of semi-supervised classifiers alone (without the LBP on the user–URL

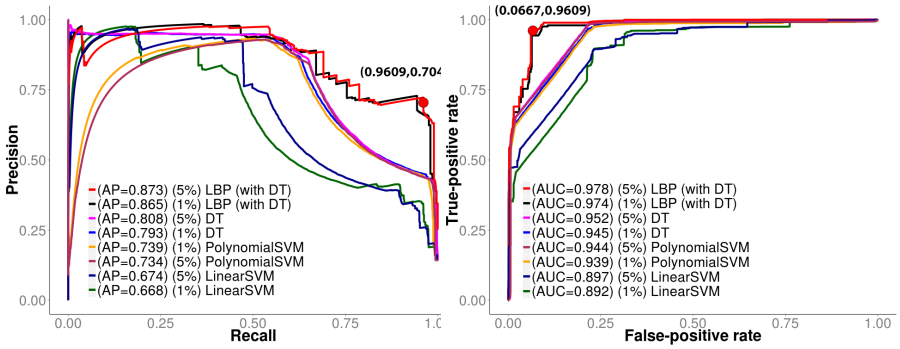

Fig. 4. Semi-supervised detection. LBP (with DT) achieves competitive performance. Red circles depicts values at classification threshold 0.5.

graph), namely linear SVM, polynomial SVM, and the decision tree (DT) classifiers. As before, both non-linear classifiers outperform linear SVM, where DT is superior to polynomial SVM.

Next, we employ our semi-supervised method. In particular, we leverage the user–URL graph in which we initialize the class priors of (1% or 5%) labeled URLs as $(1-\epsilon, \epsilon)$ for spam URLs, and vice versa for the benign. The class priors of unlabeled URLs are set to the class probabilities from DT, and of the users as $(0.5, 0.5)$, i.e. unbiased. As we see in Figure 4, incorporating relational information through LBP significantly improves the detection performance. Perhaps more importantly, the performance is desirably high; at a small false positive rate of 0.0667, recall and precision are above 0.95 and 0.70, respectively.

6 Conclusion

We considered the problem of detecting spam URLs that appear on the Web in various contexts. Our main goal has been to build an effective solution that is at the same time (*i*) context-free, (*ii*) adversarially robust, and (*iii*) semi-supervised; such that it is generalizable across Web service boundaries, costly to evade by spammers, and applicable in the face of label scarcity, respectively.

To achieve these goals, we utilize the URL redirect chains and the underlying network that they form to design three categories of domain-agnostic features. Our features are closely tied to the operational characteristics of the spammers, particularly related to their (1) reusing and sharing of resources, (2) heterogeneous hosting infrastructure, and (3) flexibility. Intuitively, evading detection by changing their behavior would incur considerable monetary or management cost upon the spammers. Evaluations on a large Twitter collection with millions of URL posts show that our context-free features yield quite similar performance against context-aware features. Moreover, our semi-supervised detection algorithm produces competitive results, at above 0.96 recall and 0.70 precision with false positive rate below 0.07, even with very limited supervision.

We publicly share our Twitter URL data collection (including ground truth labels, redirect chains, and the RCG) as well as our redirect chain crawler at `http://bit.ly/2jvdiFI`.

Acknowledgments

This research is sponsored by NSF CAREER 1452425 and IIS 1408287, DARPA Transparent Computing Program under Contract No. FA8650-15-C-7561, and ARO Young Investigator Program under Contract No. W911NF-14-1-0029. Any conclusions expressed in this material are of the authors and do not necessarily reflect the views, expressed or implied, of the funding parties.

References

1. D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing internet scam hosting infrastructure. In *Usenix Security*, 2007.
2. F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on Twitter. In *CEAS*, 2010.
3. N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *KDD*, pages 99–108, 2004.
4. H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao. Detecting and characterizing social spam campaigns. In *IMC*, 2010.
5. C. Grier, K. Thomas, V. Paxson, and C. M. Zhang. @spam: the underground on 140 characters or less. In *CCS*, pages 27–37, 2010.
6. Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *AIRWeb*, 2005.
7. R. Kindermann and J. L. Snell. *MRFs and Their Applications*. 1980.
8. K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots + machine learning. In *SIGIR*, 2010.
9. S. Lee and J. Kim. WarningBird: Detecting Suspicious URLs in Twitter Stream. In *NDSS*, 2012.
10. D. Lowd and C. Meek. Adversarial learning. In *KDD*, pages 641–647, 2005.
11. L. Lu, R. Perdisci, and W. Lee. SURF: detecting and measuring search poisoning. In *CCS*, pages 467–476, 2011.
12. J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *KDD*, pages 1245–1254, 2009.
13. J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *ICML*, 2009.
14. P. G. Neville. Decision Trees for Predictive Modeling. *SAS Institute Inc.*, 1999.
15. S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW*, 2007.
16. A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *CCS*, 2007.
17. S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based blacklists. In *Malicious & Unwanted Softw.* IEEE, 2008.
18. G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *ACSAC*, pages 1–9, 2010.
19. G. Stringhini, C. Kruegel, and G. Vigna. Shady paths: leveraging surfing crowds to detect malicious web pages. In *CCS*, pages 133–144, 2013.
20. K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *IEEE Symp. on Sec. and Priv.*, 2011.
21. B. Wu and B. D. D. 0001. Cloaking and redirection: A preliminary study. In *AIRWeb*, pages 7–16, 2005.
22. J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In Exploring AI in the new millennium. 2003.