INFORMATION ASSURANCE: DETECTION OF WEB SPAM ATTACKS IN SOCIAL MEDIA

Pang-Ning Tan, Feilong Chen, and Anil K Jain

Department of Computer Science & Engineering, Michigan State University

East Lansing, MI 48823-1226

ABSTRACT

As online social media applications continue to gain its popularity, concerns about the rapid proliferation of Web spam has grown in recent years. These applications allow spammers to submit links anonymously, diverting unsuspected users to spam Web sites. This paper presents a novel co-classification framework to simultaneously detect Web spam and spammers in social media Web sites based on their content and link-based features. Using data from two real-world applications, we empirically showed that the proposed co-classification framework is more effective that learning to classify the Web spam and spammers independently. We also investigate the efficacy of combining data from multiple social media applications to improve Web spam detection.

1. INTRODUCTION

With the emergence of social media applications such as blogs, social bookmarking, and wikis, concerns about the alarming increase of Web spam has grown in recent years. For example, out of a sample of more than ninety thousand spam URLs extracted from a benchmark email spam corpus (Webb et al., 2006), our study showed that about 7% of them were posted at digg.com, a social news Web site while 18% of them were posted at delicious.com, a social bookmarking Web site. These social media applications allow users to post interesting news stories or submit links to their favorite Web sites. Unfortunately, spammers may take advantage of this capability by creating bogus user accounts and submitting links that direct users to spam Web sites. Worse still, some of these Web sites may trick unsuspected users into divulging their personal information or allow malicious code to be injected to the user's browser. To alleviate such Web spam attacks, it is critical to develop effective techniques that can automatically detect Web spam in social media applications.

While there has been extensive research on detecting spam on the World Wide Web, spam detection in social media is still in its infancy. Early works have focused on detecting spam in blogs (Ishida, 2008; Lin et al., 2008), social bookmarking systems (Krause et al., 2008), and online review forums (Jindal and Liu, 2008). Figure 1 illustrates the conceptual difference between spam detection on the World Wide Web and in social media applications. The former assumes that the data is composed of a single, homogeneous network consisting of nodes of the same type (Web pages) while the latter is a multi-graph network containing nodes of different types (users and their submitted URLs). Spam detection on social media applications can therefore be decomposed into two sub-problems—detecting spam URLs and the spammers who are responsible for posting them.

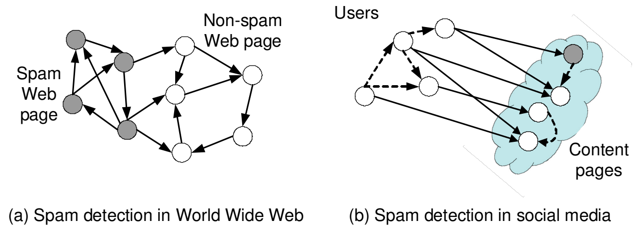
Spam Web page
Non-spam Web page
Users
Content pages
(a) Spam detection in World Wide Web
(b) Spam detection in social media

Figure 1. Comparison between spam detection in the World Wide Web (where the network consists of hyperlinked Web pages) and spam detection in social media (where the network consists of users and their shared social media content).

There are many types of features that can be used for Web spam detection in social media. For example, content-based features can be derived from the set of tags assigned by users to the URLs they have submitted. Link-based features can also be constructed from the links between users, links between URLs, or links between users and their submitted URLs. However, integrating such diverse features into a Web spam detection algorithm is not a trivial task. First, existing classifiers such as support vector machine (SVM) are not designed to handle both content-based and link-based features. Second, the links are often noisy due to the fact that some legitimate users may inadvertently link to spam URLs while some spammers may deliberately add links to legitimate Web sites to evade detection.

This paper presents a robust framework to effectively detect Web spam and spammers in social media Web sites. Our framework extends the least-square support vector machine (LS-SVM) classifier to handle data that contains both link-based and content-based features. The framework was developed based on the following two assumptions: (1) Spam URLs are more likely to be posted by spammers than non-spammers and (2) Spammers are more likely to link with other spammers than to non-spammers. We formalize these assumptions as graph-based constraints and develop a co-classification algorithm to learn a pair of classifiers that simultaneously detect Web spam and spammers at a social media Web site. Experimental results suggest that the co-classification approach is more effective than learning to classify the Web spam and spammers independently.

Our co-classification framework can be extended to nonlinear models using the kernel trick and adapted to a semi-supervised learning setting. Finally, we also investigate the effectiveness of augmenting data from multiple social media applications to improve Web spam detection using a combination of co-training with the co-classification approach.

2. PRELIMINARIES

We begin with a brief discussion of the terminology and notations used in this paper. The terminology is applicable to most social media applications that allow users to submit links to their favorite Web sites:

- **User**: A registered visitor of the social media Web site. Let **U** be the set of all users.
- **URL**: A Web address submitted by users. Let **B** be the set of all submitted URLs.
- **Tag**: A keyword assigned by a user to a submitted URL. Each URL can be associated with zero or more tags. Let **T** be the set of all tags.
- **Post**: A 4-tuple (b, u, t, τ), where $b \in \mathbf{B}$, $u \in \mathbf{U}$, $t \subseteq \mathbf{T}$, and τ is the timestamp at which the user posted the URL. We denote the set of all posts, also known as the posting history, by Π. Let $\mathbf{E}_b = \{(u, b) \mid u \in \mathbf{U}, b \in \mathbf{B}; \exists t, \tau: (b,u,t,\tau) \in \Pi\}$ be the set of all user-URL pairs in which user u submitted link to URL b.
- **"Friendship" relation:** A directional link between users. If a user u accepts a "friendship" invitation[1] from user v, then u becomes a friend of v. Let $\mathbf{E}_u = \{(u, v) \mid u, v \in \mathbf{U}\}$ be the set of all user pairs in which u is a friend of v. The friendship relation may not be reciprocal, i.e., $(u, v) \in \mathbf{E}_u$ does not imply $(v, u) \in \mathbf{E}_u$
- **URL Popularity:** The number of users who posted a URL, i.e., $\boldsymbol{\sigma}(\boldsymbol{b}) = \sum_{u} \boldsymbol{E}_b(\boldsymbol{u},\boldsymbol{b})$.
- **User Popularity:** The number of friends associated with a user, i.e., $\boldsymbol{\sigma}(\boldsymbol{u}) = \sum_{v} \boldsymbol{E}_u(\boldsymbol{v},\boldsymbol{u})$.

[1] For twitter.com and social bookmarking Web sites such as delicious.com, $\mathbf{E}_u$ corresponds to a "follower" or "fan" relation, in which $(u, v) \in \mathbf{E}_u$ if u is a follower (fan) of v.

We model the social media data as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \mathbf{B} \cup \mathbf{U}$ is the set of nodes (URLs and users) and $\mathbf{E} = \mathbf{E}_b \cup \mathbf{E}_u$ is the set of links. Our proposed framework is based on the following two assumptions:

Spam-Link Assumption: If $(u,b) \in E_b$ and b is a spam URL, then u is more likely to be a spammer than a non-spammer. This is because we do not expect non-spammers to post spam URLs (although spammers may post links to non-spam URLs to evade detection).

Spammer-Link Assumption: If $(u,v) \in E_u$ and v is a spammer, then u is also likely to be a spammer. This is because non-spammers are likely to decline invitation to be friends with spammers (though spammers are likely to accept invitations from non-spammers).

As will be shown later in Section 4, these assumptions are enforced as graph regularization constraints in our proposed co-classification framework.

3. SPAM IN SOCIAL MEDIA WEB SITES

This section presents our analysis on the prevalence of Web spam at two popular social media Web sites, delicious.com and digg.com. The former is a social bookmarking Web site that allows users to bookmark their favorite URLs, add tags to each bookmark, and share them with other users. The latter is a social news Web site, in which users may post links to interesting news stories they found on the Internet and vote on the stories submitted by other users. For the remainder of this paper, we use the term URL interchangeably to refer to either a bookmark on delicious.com or a submitted news story on digg.com.

We created a list of spam URLs by extracting the links embedded in a benchmark email spam corpus (Webb et. al. 2006). We choose this data set because of its broader coverage of the Web, unlike other benchmark Web spam data sets such as (Castillo et. al. 2005), which is limited to Web sites hosted at the .uk domain. Our rationale for considering the embedded links in email spam messages as Web spam is that, since the Web sites are involved in email spamming, they are likely to participate in other self-promoting activities including Web spam. We further verified that the extracted links are mostly spam Web sites by checking their safety ratings at McAfee SiteAdvisor and then manually examining the ones that were not flagged as Web spam. From the list of 94,198 spam Web sites that were extracted, we found 6,420 (~7%) of them posted at digg.com and 16,537 (~18%) of them posted at delicious.com. These results suggest that spam is quite prevalent in social media applications, and thus, must be eliminated to improve the quality of information posted on these Web sites.

The social news Web site digg.com also provides additional counter-measures to safeguard against the promotion of Web spam by allowing users to "vote down" or "bury" uninteresting posts. Even though such measures help to reduce Web spam, they are not entirely full proof because spammers may create several bogus user accounts and collude with each other to promote ("vote up" or "dig") their spam Web sites.

The problem is even more acute at delicious.com, in which nearly one-third of the spam URLs were found to be bookmarked by at least 20 users and about 23% of them were bookmarked by at least 30 users. Some of the spam URLs were as popular as the non-spam URLs listed at http://delicious.com/popular/. An example of a popular spam URL at delicious.com was the "Airset" spam, which was identified Brian Dear[2]. He noted some of the characteristics of the Airset spam include: (1) all the bookmarks were directed to the same URL, (2) all the bookmarks were assigned the same keyword tag EVDB, and (3) the majority of the users who bookmarked the spam URL posted no other URLs. While such an unusual pattern can be potentially used as a signature for detecting Web spam, it is not exhaustive enough to cover all types of Web spam as some of the more experienced spammers may post links to both legitimate and spam Web sites to obfuscate their activities.

To illustrate the difficulty in identifying Web spam and spammers, consider the plots shown in Figure 2 and Figure 3. In this study, we consider a user to be a spammer if the number of submitted spam bookmarks exceeds some threshold, λ. Figure 2 compares the user popularity for spammers against non-spammers. As previously mentioned, user popularity refers to the number of "friends" associated with a user. At delicious.com, this measure is given by the number of "fans" added to the user's network in order to allow them to keep track of the user's bookmarks. Since there are more non-spammers than spammers, we randomly sample an equal number of non-spammers from the data set and plot their popularity distribution. Figure 2 shows that the popularity of non-spammers follows a power law distribution. A similar distribution was observed for spammers, except the number of friends for the most popular spammers is much lower than the number of friends for the most popular non-spammers. In terms of the number of URLs submitted by spammers and non-spammers, again, the shape of the distributions resembles each other, as shown in **Figure 3**.

[2] A discussion of the Airset spam can be found at http: //www.brianstorms.com/archives/000575.html.

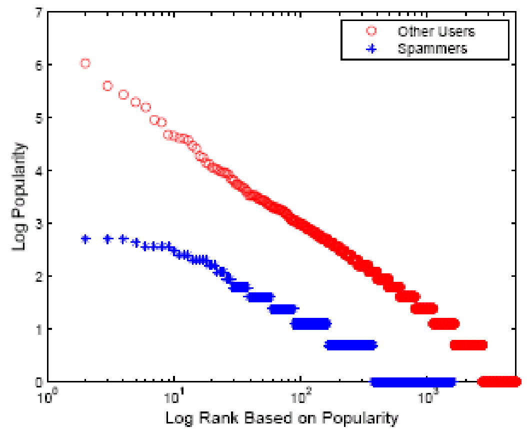

Figure 2. Distribution of user popularity for spammers versus other users at delicious.com. The horizontal axis corresponds to the rank of user's popularity.

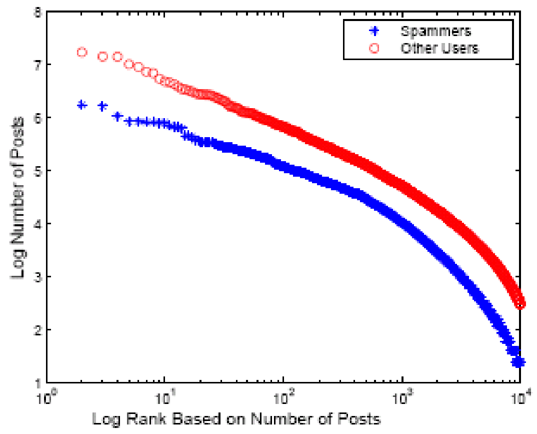

Figure 3. Distribution of number of posts submitted by spammers and other users. The horizontal axis corresponds to the rank of their number of posts.

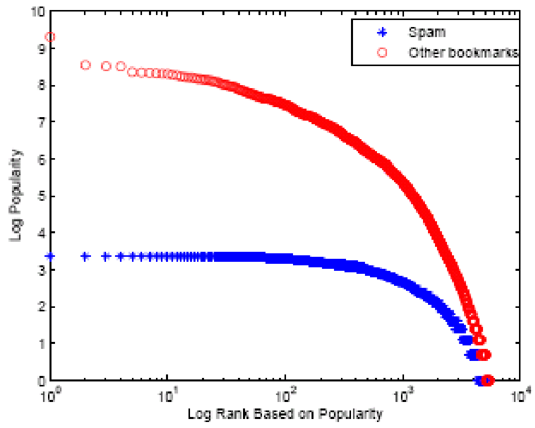

Figure 4. Distribution of bookmark popularity for Distribution of bookmark popularity for spam and non-spam bookmarks. The horizontal axis corresponds to the rank of bookmark's popularity.

We also analyzed the popularity of spam URLs compared to non-spam at delicious.com. As can be seen from **Figure 4**, the shape of their distributions is very similar, though, the most popular spam URLs are bookmarked by less number of users compared to the most popular non-spam URL. Neither curve appears to follow a power law distribution.

The analysis presented in this section suggests that the amount of Web spam in social media Web sites is quite prevalent and their popularity distributions resemble those of non-spam URLs and users. Therefore, it is not possible to set a minimum threshold to filter the spammers and spam URLs without misclassifying the non-spammers and non-spam URLs. This makes it difficult to identify spammers or spam URLs based on their popularity measures alone. The analysis here also suggests the need to incorporate other features, such as the tags used and the links between users to improve performance of the Web spam and spammer detection tasks.

4. A CO-CLASSIFICATION FRAMEWORK FOR DETECTING WEB SPAM AND SPAMMERS

Our proposed co-classification framework is designed to train a pair of classifiers for detecting Web spam and spammers. The co-classification problem can be formally defined as follows. Let $V_b = \{(\vec{x}_i^{(b)}, y_i^{(b)})\}$ be the set of labeled URLs, where $\vec{x}_i^{(b)}$ denote the content-based feature vector a URL and $y_i^{(b)}$ is the class label. We denote a spam URL as $y_i^{(b)} = +1$ and $y_i^{(b)} = -1$ for non-spam URL. Similarly, let $V_u = \{(\vec{x}_i^{(u)}, y_i^{(u)})\}$ be the set of labeled users where $\vec{x}_i^{(u)}$ denote the content-based feature vector a user (derived from the set of tags used by the user) and $y_i^{(u)}$ is the class label. Again, we denote a spammer as $y_i^{(u)} = +1$ and $y_i^{(u)} = -1$ for non-spammer.

DEFINITION 1: *Given the sets of labeled URLs* V_b, *labeled users* V_u, *user-user links* $E_u = \{(\vec{x}_i^{(u)}, \vec{x}_j^{(u)})\}$, *and user-URL links* $E_b = \{(\vec{x}_i^{(u)}, x_j^{(b)})\}$, *the goal of Web spam and spammer detection tasks is to learn a pair of classifiers: (1)* $f_{URL}(\vec{x}^{(b)})$ *to map a URL* $\vec{x}^{(b)}$ *to its class label* $y^{(b)} \in \{+1,-1\}$ *and (2)* $f_{user}(\vec{x}^{(u)})$ *to map a user* $\vec{x}^{(u)}$ *to its class label* $y^{(b)} \in \{+1,-1\}$.

The classifier used in this study is an extension of the least-square support vector machine (LS-SVM) classifier (Suykens et. al. 2002), which we have modified to handle multi-graph network data. Unlike regular support vector machine (SVM), which requires solving a quadratic programming problem involving inequality constraints, LS-SVM involves equality constraints and can be reduced to solving a system of linear equations. Previous studies have also shown that the generalization performance of LS-SVM is comparable to regular SVM (Gestel et al., 2004; Zhang and Peng, 2004). Furthermore, when the data points are linearly independent, it has been theoretically shown that LS-SVM is equivalent to hard margin SVM using the Mahalanobis distance measure (Ye and Xiong, 2007).

4.1. LS-SVM Classifier

The linear LS-SVM classifier learns a hypothesis $f(\vec{x}) = sign(\vec{w}^T \vec{x} + b)$ for classifying a feature vector $\vec{x}$ (Suykens et. al. 2002). The classifier is trained to minimize the following objective function,

$$\tfrac{1}{2}\|\vec{w}\|_2^2 + \lambda \sum_{i=1}^{l} e_i^2$$

subject to the following constraint:

$$\forall i:\ y_i[\vec{w}^T \vec{x}_i + b] = 1 - e_i$$

where $\boldsymbol{w}$ and $\boldsymbol{b}$ are the model parameters, l is the training set size, and $\boldsymbol{e}_i$ is the prediction error of the ith training point. The constraint optimization problem can be cast into the following problem using the Lagrangian formulation:

$$L(\vec{w}, b, \vec{e}, \vec{\partial}) = \tfrac{1}{2}\|\vec{w}\|_2^2 + \lambda_1 \sum_{i=1}^{l} e_i^2 - \sum_{i=1}^{l} \partial_i \left(y_i[\vec{w}^T \vec{x}_i + b] - 1 + e_i \right) \quad (1)$$

where $\{\partial_i\}$ is the set of Lagrange multipliers.

4.2. Proposed Co-classification Framework

In principle, we may use Equation (1) to construct a pair of LS-SVM classifiers for detecting Web spam and spammers. However, such classifiers do not incorporate the link-based features and must be trained independently using only the content-based features for URLs and users. Instead of building the classifiers independently, our co-classification framework utilizes the link information in E_b and E_u to ensure that the hypotheses $f_{URL}(\vec{x}^{(b)})$ and $f_{user}(\vec{x}^{(u)})$ are consistent with each other.

More specifically, our framework extends the original LS-SVM formulation in two ways. First, it enforces the following constraint on the class labels between pairs of linked nodes in the user networks:

$$\forall (i,j) \in E_u,\ y_j^{(u)} > 0:\ \vec{w}_u^T \vec{x}_i^{(u)} + b_u \ge \vec{w}_u^T \vec{x}_j^{(u)} + b_u$$

The preceding constraint states that users who are friends of known spammers must also be spammers, which is consistent with the Spammer-Link assumption described in Section 2. By adding this constraint to our framework, it will penalize models that predict users who are linked to spammers as non-spammers. Second, it imposes an additional constraint on the links between users and the URLs they have posted:

$$\forall (i,j) \in E_b,\ y_j^{(b)} > 0:\ \vec{w}_u^T \vec{x}_i^{(u)} + b_u \geq \vec{w}_b^T \vec{x}_j^{(b)} + b_b$$

This constraint ensures that our co-classification framework would penalize models in which non-spammers are allowed to submit many spam pages, which is consistent with the Spam-Link assumption given in Section 2.

Putting them altogether, our co-classification framework is designed to learn the classifiers f_{user} and f_{URL} simultaneously by minimizing the following objective function:

$$\begin{aligned} L_{co-class} = {} & \tfrac{1}{2}\|\vec{w}_b\|_2^2 + \gamma_1 \sum_{i=1}^{l} e_i^2 + \tfrac{1}{2}\|\vec{w}_u\|_2^2 + \gamma_2 \sum_{i=1}^{l} \xi_i^2 \\ & - \sum_{i=1}^{l} \partial_i \{ y_i^{(b)} [\vec{w}_b^T \vec{x}_i^{(b)} + b_b] - 1 + e_i \} \\ & - \sum_{i=1}^{l} \beta_i \{ y_i^{(u)} [\vec{w}_u^T \vec{x}_i^{(u)} + b_u] - 1 + \xi_i \} \\ & - \gamma_3 \sum_{\substack{(i,j)\in E_u \\ y_j^{(u)} > 0}} \delta_{ij} \left(\vec{w}_u^T \vec{x}_i^{(u)} - \vec{w}_u^T \vec{x}_j^{(u)} \right) \\ & - \gamma_4 \sum_{\substack{(i,j)\in E \\ y_j^{(b)} > 0}} \eta_{ij} \left(\vec{w}_u^T \vec{x}_i^{(u)} + b_u - \vec{w}_b^T \vec{x}_j^{(b)} - b_b \right) \end{aligned} \tag{2}$$

where $\vec{w}_b$, $\vec{w}_u$, b_b, b_u, $\vec{w}_b$, $\vec{e}$, $\vec{\xi}$, $\vec{\partial}$ and $\vec{\beta}$ are the model parameters to be estimated from training data and $\gamma_{i=1,2,3,4}$ are the user-specified parameters. For our experiments, we set $\gamma_1 = \gamma_2 = 1$ whereas γ_3 and γ_4 are estimated from the data via cross validation. Note that the last two terms in Equation (2) correspond to the graph regularization constraints associated with the Spam-Link and Spammer-Link assumptions. The solution to the objective function is obtained by taking the partial derivative of the Lagrangian L with respect to each of the model parameters and setting them to zero. It can be shown that the model parameters are obtained by solving a system of linear equations. Due to lack space, we refer the readers to (Chen et al., 2009) for a full derivation of the model parameters.

The objective function given in (2) can be solved in a supervised or semi-supervised learning setting, depending on the nodes used to express the graph regularization constraints in E_b and E_u. In a supervised learning setting, E_b and E_u involve only URLs and users that belong to the labeled training data, V_b and V_u . For semi-supervised learning, the constraints due to E_b and E_u in (3) include unlabeled URLs (U_b) and unlabeled users (U_u) as well. We have implemented both supervised (Sup-Co-Class) and semi-supervised (Semi-Co-Class) co-classification algorithms in this study.

Once the model parameters have been estimated, a user $\vec{x}_{test}^{(u)}$ and a bookmark $\vec{x}_{test}^{(b)}$ can be classified as follows:

$$f_{user}\left(\vec{x}_{test}^{(u)}\right) = \vec{w}_u^T \vec{x}_{test}^{(u)} + b_u \ ,\quad f_{URL}\left(\vec{x}_{test}^{(b)}\right) = \vec{w}_b^T \vec{x}_{test}^{(b)} + b_b .$$

The sign of the function value (positive or negative) will be used as the predicted class.

Finally, although the proposed co-classification framework assumes a linear classifier, it can be extended to non-linear models using the well-known kernel trick,

$$\begin{aligned} f_u\left(\vec{x}_{test}^{(u)}\right) = {} & b_u + \sum_{i=1}^{l_u} \beta_i y_i^{(b)} \vec{x}_i^{(u)} \cdot \vec{x}_{test}^{(u)} \\ & + \gamma_3 \sum_{\substack{(i,j)\in E_u \\ y_j > 0}} \delta_{ij} \left[\vec{x}_i^{(u)} - \vec{x}_j^{(u)} \right] \cdot \vec{x}_{test}^{(u)} \\ & + \gamma_4 \sum_{\substack{(i,j)\in E_b \\ y_j > 0}} \eta_{ij} \vec{x}_i^{(u)} \cdot \vec{x}_{test}^{(u)} \end{aligned} \tag{3}$$

$$\begin{aligned} f_b\left(\vec{x}_{test}^{(b)}\right) = {} & \sum_{i=1}^{l_u} \partial_i y_i^{(b)} \vec{x}_i^{(b)} \cdot \vec{x}_{test}^{(b)} \\ & + \gamma_4 \sum_{\substack{(i,j)\in E_b \\ y_j > 0}} \eta_{ij} \vec{x}_j^{(b)} \cdot \vec{x}_{test}^{(b)} + b_b \end{aligned}$$

Since the classification step above (as well as the training step) involves only dot product operations between the labeled training examples and the unlabeled test example, these operations can be replaced by their corresponding Mercel kernels for nonlinear classifiers.

Algorithm 1 summarizes the high-level overview of our Co-Class algorithm. The *LinearSolver* function optimizes the objective function by solving the system of linear equations. The Classify function takes as input the model parameters and unlabeled examples to generate their predicted class labels.

```
Input: $\mathcal{V}_b, \mathcal{V}_u, \mathcal{U}_b, \mathcal{U}_u, E_b, E_u, \gamma$
Output: $\{y_i^{(u)} | i = l_u + 1 \cdots n_u\}, \{y_j^{(b)} | b = l_b + 1 \cdots n_b\}$.
Method:
1. $(\alpha, \beta, b_u, b_b) \leftarrow$ LinearSolver($\mathcal{V}_b, \mathcal{V}_u, E_b, E_u, \gamma$)
2. $\mathbf{y}^{(u)} \leftarrow$ Classify($\mathcal{U}_u, \mathcal{V}_u, \beta, b_u$)
3. $\mathbf{y}^{(b)} \leftarrow$ Classify($\mathcal{U}_b, \mathcal{V}_b, \alpha, b_b$)
```

Algorithm 1. Co-Classification Algorithm

4.3. Experimental Evaluation

We applied the proposed framework to a synthetic dataset and two real-world datasets collected from the delicious.com and digg.com Web sites. The real-world data sets each contains 20,000 users and 20,000 URLs. The Web spam URLs are extracted from the benchmark email spam dataset (Webb et. al. 2006).

The synthetic data set is generated as follows. First, the number of content-based features for users and URLs are set to 100. The values of each feature are generated randomly from a uniform distribution in the range between [0, 1]. We randomly label 20% of the data as spam and the remaining 80% as non-spam. For the user data set, we randomly select k features as the salient features for identifying Web spam. The values for each of the k salient features in the spam class are then perturbed by adding a random number generated from a uniform distribution in the range between [0, 0.5]. The values of the k salient features for non-spam data are left unperturbed. A similar strategy is also applied to generate the salient features for URL data. The link structures E_u and E_b are created by randomly assigning links according to the Spam-Link and Spammer-Link assumptions. To simulate E_u, a spammer is a friend of another spammer with probability 0.5 and a friend of a non-spammer with probability 0.2. In contrast, a non-spammer is a friend of a spammer or non-spammer with probabilities 0.02 and 0.25, respectively. Similarly, the link structure of E_b is generated by assuming that a spammer will post a spam URL with probability 0.5 and a non-spam URL with probability 0.2. On the other hand, a non-spammer will post a spam URL with probability 0.02 and a non- spam URL with probability 0.25.

For comparison purposes, we use SVM-light (Joachims, 1999) to build a pair of support vector machine classifiers on users and bookmarks independently. The performance of the classifiers is evaluated using the following measures:

$$precision = tp/(tp + fp)$$
$$recall = tp/(tp + fn)$$
$$F_1 = 2 \cdot precision \cdot recall/(precision + recall)$$

where tp, fp, and fn are the true positive, false positive and false negative, respectively. Precision measures the proportion of examples predicted as Web spam (or spammer) that were correctly classified by the classifier while recall measures the percentage of Web spam (or spammer) examples that were correctly predicted by the classifier. The F_1 measure summarizes both precision and recall into a single number by taking their harmonic mean.

First we report the experimental results for the synthetic data set. These results are obtained after repeating the experiment twenty times, each time using a different dataset simulated by our synthetic data generator. For this experiment, we also varied the number of salient features from 4 to 9 (out of 100 features).

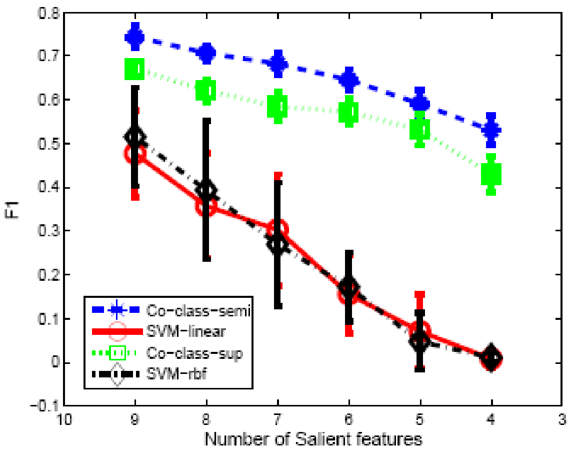

Figure 5. Comparison of F_1 measure for classifying users in the synthetic data.

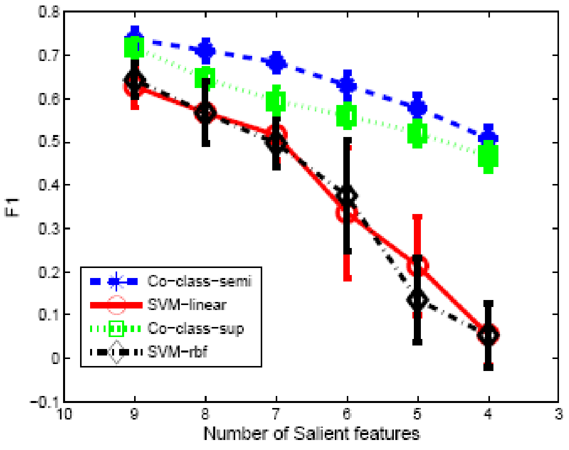

Figure 6. Comparison of F_1 measure for classifying URLs in the synthetic data.

Figure 5 compares the F_1-measure for classifying users using linear SVM and nonlinear SVM with RBF kernel against both supervised and semi-supervised co-classification algorithms. The figure shows that our proposed algorithms consistently outperform both linear and nonlinear SVM. In particular, when the number of salient features decreases, the improvement shown by our algorithms become even more pronounced. This is because, as the number of salient features decreases, it is more difficult to separate the two classes, thus causing the performance of SVM to degrade signficantly. However, our supervised and semi-supervised Co-Class algorithms still manage to maintain a high F_1-measure because they utilize information from the URL class labels to train the model. A similar conclusion can be drawn from the results of URL classification shown in Figure 6. There is no significant difference between the performance of linear and nonlinear SVM classifiers on this data set. However, the performance of semi-supervised Co-Class algorithm is relatively better than its supervised counterpart. This is because the semi-supervised Co-Class algorithm takes advantage of the link structures in E_u and E_b to propagate the labeled information to neighboring unlabeled users and URLs.

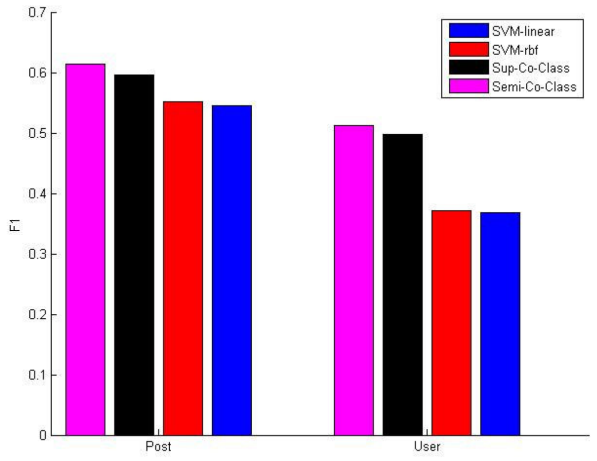

Figure 7. Results on delicious.com dataset

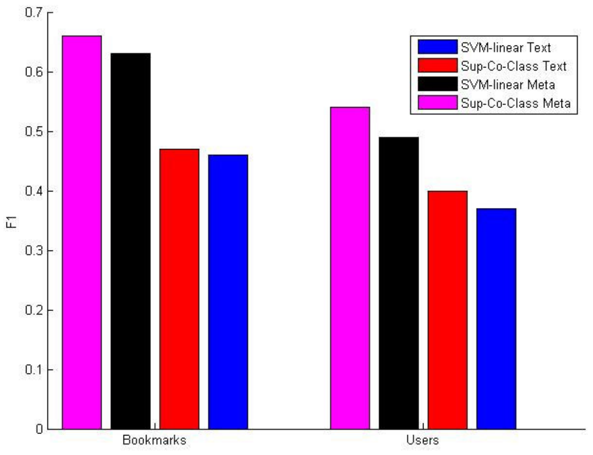

Figure 8. Results on digg.com dataset

Figure 7 and Figure 8 show the results for detecting Web spam and spammers on the delicious.com and digg.com social media Web sites. The results again showed that both supervised and semi-supervised co-classification algorithms outperformed the linear and nonlinear SVM classifiers trained to classify the users or URLs independently. The semi-supervised co-classification algorithm is also more effective than the supervised version.

5. CO-CLASSIFICATION ON MULTIPLE SOCIAL MEDIA WEB SITES

We have further extended the framework to combine social media data from multiple domains (for instance, delicious.com and digg.com) using a co-training approach. This approach was motivated by the observation that there are common URLs posted on both social media Web sites.

5.1. Co-Training with Co-Classification

Co-training (Blum et. al., 1998) is a semi-supervised learning technique that assumes each labeled or unlabeled example can be represented by two different sets of features. Each feature set provides a complementary view of the data example. Ideally, we want the two feature sets to be conditionally independent given the class and that each feature set contains sufficient information to correctly predict the class label of a given example.

Note that the problem described in this section is somewhat different than traditional co-training. First, there are only a small fraction of the URLs that have two sets of features, i.e., posted on both Web sites. The remaining URLs have only one set of features. Second, the co-training problem investigated in this study involves co-classifying heterogeneous nodes in multiple networks, whereas traditional co-training is applied mostly to non-network data.

Our proposed co-training with co-classification approach first learns a pair of classifiers for each domain source (digg.com and delicious.com) using their shared labeled examples (i.e., URLs posted on both delicious.com and digg.com Web sites). It then selects the test examples with highest confidence in their predictions to be augmented to the labeled training data. This process is repeated until the algorithm converges.

5.2. Experimental Results

We evaluated the performance of our co-training with co-classification algorithm using the delicious.com and digg.com datasets described in Section 4. After checking the submitted URLs, we found about 8% of the URLs are common to both Web sites. In order to evaluate the effectiveness of using the common URLs in improving classification performance, we gradually increased the amount of common URLs in the training set from 4% to 6% and eventually 8%.

The experimental results given in Figure 9 showed that the performance of the co-training with co-classification approach, denoted as Co-Co-Class, improved while the amount of common URLs increased. Furthermore, its F_1 measure is higher than that for the SVM classifier and co-classification algorithm trained on a single domain only. Nevertheless, the percentage of improvement is not as significant as that observed in Figure 7 and Figure 8.

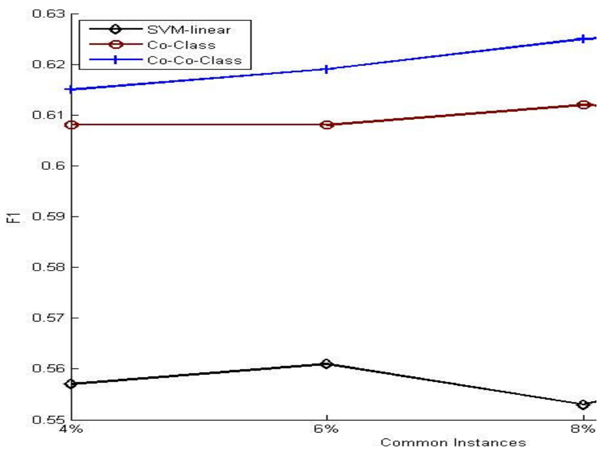

Figure 9. Comparison between Co-class on single domain and Co-class (with co-training) on network data from multiple domains.

6. CONCLUSIONS

This paper focuses on the problem of Web spam and spammer detection in social media Web sites. Unlike search engine spam, Web spam from social bookmarking and social news aggregator Web sites are potentially damaging because it may direct users to malicious Web sites that compromise browser security. To overcome this problem, we presented a co-classification framework that simultaneously trains classifiers for detecting Web spam and spammers. We demonstrated that such a strategy is more effective than learning each task independently. We further extended the framework to combine information from multiple social media Web sites, in order to achieve better classification accuracy.

ACKNOWLEDGMENT

This material is based upon work supported in part by the U.S. Army Research Office under grant number W911NF-09-I-0566 and the Office of Naval Research under grant number N00014-09-1-0663.

REFERENCES

Blum, A. and Mitchell, T., 1998: Combining labeled and unlabeled data with co-training. In Proc. *COLT*.

Castillo, C., Donato, D., Beccheti, L., Boldi, P., Santini, M., and Vigna, S., 2006: A reference collection for web spam. SIGIR Forum, 40(2).

Chen, F., Tan, P.-N., and Jain, A., 2009: A Co-classification Framework for Detecting Web Spam and Spammers in Social Media Web sites. In Proc of CIKM.

Gestel, T., Suykens, J., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., Moor, B.D., and Vandewalle, J., 2004: Benchmarking least square support vector machine classifiers. Machine Learning, 54(1): 5–32.

Ishida, K., 2008: Extracting spam blogs with co-citation clusters. In Proc of WWW '08.

Jindal, N. and Liu, B., 2008: Opinion spam and analysis. In Proc. of WSDM.

Joachims, T., 1999: Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (eds.), MIT Press.

Krause, B., Schmitz, C., Hotho, A., and Stumme, G., 2008: The anti-social tagger: detecting spam in social bookmarking systems. In Proc. of the 4th Int'l Workshop on Adversarial Information Retrieval on the Web.

Lin, Y., Sundaram, H., Chi, Y., Tatemura, J., and Tseng, B. L., 2008: Detecting splogs via temporal dynamics using self-similarity analysis. ACM Trans. Web, 2(1): 1–35.

Suykens, J., Gestel, T., Brabanter, J., Moor, B.D., and Vandewalle, J., 2003: Least Square Support Vector Machines, World Scientific Publishing, Singapore.

Webb, S., Caverlee, J., and Pu, C., 2006: Introducing the Webb spam corpus: Using email spam to identify web spam automatically. In Proc. of CEAS '06.

Ye, J., and Xiong, T., 2007: SVM versus least square SVM. In Proc of the Eleventh Int'l Conf on Artificial Intelligence and Statistics

Zhang, P. and Peng, J., 2004: SVM vs regularized least squares classification. In Proc of ICPR.