Social Spam Detection

Benjamin Markines[1,2]* Ciro Cattuto[2] Filippo Menczer[1,2]

[1]School of Informatics, Indiana University, Bloomington, Indiana, USA
[2]Complex Networks Lagrange Laboratory, Institute for Scientific Interchange Foundation, Torino, Italy

ABSTRACT

The popularity of social bookmarking sites has made them prime targets for spammers. Many of these systems require an administrator's time and energy to manually filter or remove spam. Here we discuss the motivations of *social spam,* and present a study of automatic detection of spammers in a social tagging system. We identify and analyze six distinct features that address various properties of social spam, finding that each of these features provides for a helpful signal to discriminate spammers from legitimate users. These features are then used in various machine learning algorithms for classification, achieving over 98% accuracy in detecting social spammers with 2% false positives. These promising results provide a new baseline for future efforts on social spam. We make our dataset publicly available to the research community.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services; K.4.2 [**Computers and Society**]: Social Issues; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Algorithms, Design, Experimentation, Human Factors, Performance

Keywords

Social spam, tag, resource, post, annotations, tag similarity, Web 2.0

1. INTRODUCTION

With the transition to Web 2.0, various forms of social linking have gained considerable ground and have shifted the control over content as well as content structure toward the bottom. To date, the most successful form of social annotation has been collaborative tagging [12, 21, 10], to the point that tags have become the hallmark of Web 2.0 systems. As the Web becomes increasingly user-driven, tagging metadata is regarded as a first-class data source to harvest emergent semantics in social media, with the goals of improving navigation and search, learning ontologies, and making contact with more formal representations of content.

*Corresponding author. Email: bmarkine@cs.indiana.edu

AIRWeb '09, April 21, 2009 Madrid, Spain.

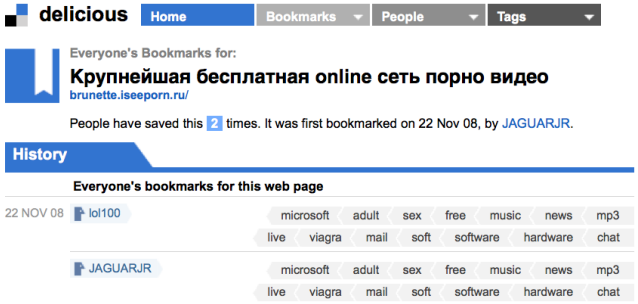

Figure 1: A spammer using two identities posts a Russian porn site on `delicious.com`. The spammer uses popular tags such as "music," "news" and "software," which are unrelated to each other and the site.

The success of tagging can be ascribed to its simplicity and open-ended nature: in a social tagging system any user can easily associate a free-form tag to any resource represented in the system. The type of resource depends on the specific system, and there are many popular systems for annotating almost any kind of media. The data structure that supports a tagging systems is a collaborative artifact known as "folksonomy," formally represented as a hyper-graph. In this view, nodes comprise users, resources and tags, and each annotation adds a hyper-edge to the graph, connecting a user, the annotated resource, and the chosen tag.

Since every user can easily add to the folksonomy, the structure of the graph is entirely user-driven, and a malicious user can exploit this control to make some content more prominent, drive user traffic to chosen targets, and in general to pollute the folksonomy. We refer to these kinds of exploitations of collaborative annotation systems as *social spam.* Identifying social spam automatically and efficiently is a key challenge in making social annotations viable for any given system and for the Web at large.

Here we restrict our analysis to *social bookmarking* systems. These systems are typically "broad folksonomies," that is, users provide annotations of content that is external to the bookmarking system (in contrast with systems like Flickr); this affords the aggregation of annotations from an entire community, and the definition of several socially-induced measures of content similarity [5, 19]. Furthermore, the success of social bookmarking systems and the large communities they bind make them an attractive target for spamming. Fig. 1 shows a typical example of social spam.

Contributions and Outline

Social spam is a relatively new research area and the literature is still sparse. After a formal definition of the data structures underly-

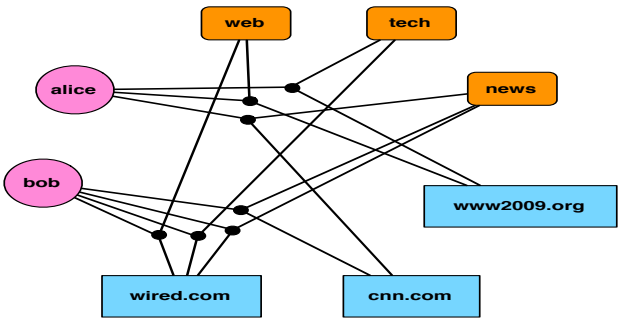

Figure 2: Example folksonomy. Two users (alice and bob) annotate three resources (cnn.com, www2009.org, wired.com) using three tags (news, web, tech). The triples (u, r, t) are represented as hyper-edges connecting a user, a resource and a tag. The 7 triples correspond to the following 4 posts: (alice, cnn.com, {news}), (alice, www2009.org, {web, tech}), (bob, cnn.com, {news}), (bob, wired.com, {news, web, tech}).

ing a folksonomy and some background on social tagging systems, we provide a brief review of related work in adversarial information retrieval and the recent shift of attention toward social spam (§ 2). In the remainder of the paper we make the following contributions:

- We discuss the main motivation and incentives behind social spam, relating it to the phenomena of click fraud and Web pollution (§ 3).
- We identify six features of collaborative tagging systems capturing various properties of social spam (§ 4).
- We analyze how the proposed features can be used to detect spammers, showing that each feature has predictive power for discriminating spammers and legitimate users (§ 5.2).
- We evaluate various supervised learning algorithms that use these features to detect spam, and show that the resulting classifiers achieve accuracy above 98% while maintaining a false positive rate near 2%. Further feature selection analysis reveals that the best performance is obtained using all six features. These promising results provide a baseline for future efforts on social spam detection (§ 5.3).
- We make our dataset publicly available to stimulate further open research on social spam detection.

2. DEFINITIONS AND BACKGROUND

A folksonomy F is a set of *triples*. The triple representation is widely adopted in the Semantic Web community [14]. Each triple (u, r, t) represents user u annotating resource r with tag t. A *post* $(u, r, (t_1, \ldots, t_n))$ can be represented as the set of triples $\{(u, r, t_1), \ldots, (u, r, t_n)\}$. Fig. 2 displays an example.

Lambiotte *et al.* introduced the tripartite graph representation for folksonomies [18, 21]. Hammond *et al.* reviewed some early social tagging systems [12]. Golder and Huberman provided a first quantitative analysis of `delicious.com` [10]. The network structure of folksonomies and the exposure of non-social behavior through network measures was investigated by Cattuto *et al.* [6]. We investigated the design of effective and efficient tag-tag and resource-resource similarity measures [20, 19].

Long before social tagging became popular, spam was a problem in other domains, first in email then in Web search [11, 1]. Unfortunately, the countermeasures that were developed for email and Web spam do not directly apply to social systems [13]. Recently, Caverlee *et al.* introduced a trust system for combating spam on social networking sites [7]. Benevenuto examined spam detection on the social media site `YouTube.com` [2]. Social annotations sites that enable voting have also been a target for spammers [3]. Koutrika *et al.* [16] conducted one of the first studies of spam detection in social bookmarking systems, the problem on which we focus here.

The dataset used in this paper has been the focus of other social spam detection efforts. Gkanogiannis *et al.* use a Rocchio-like method to maximize the discrimination between spammers and non-spammers using tags [9]. Chevalier *et al.* use features such as number of tags per post, length of tags, and number of tags with a special character for classification [8]. Kim and Hwang compute the mutual information between a tag and a user for classification with naive Bayes [15]. Krause *et al.* investigate several features based on account properties, number of account requests from a location (IP), average number of tags per post, tag blacklists, and other semantic features [17]. None of the six social spam features analyzed here are found in prior literature.

3. MOTIVATIONS OF SOCIAL SPAM

The first step toward the design of effective measures to detect and combat social spam is an understanding of the motivations behind it. Based on our experience as well as judging from past history of spam in other contexts, we argue that the most threatening motivation is financial gain. How can someone make money by abusing social tagging systems? This question has not yet been thoroughly explored. The spammer probably makes money when a user visits site X, and therefore the spammer needs to attract users to site X. Social spam is a cheap way to attract users. Other methods include email spam, search engine manipulation, and placing ads. The first is more expensive because there is already an infrastructure in place against email spam: filters, black lists, and so on. Search manipulation is more expensive because search engines have a financial interest in preventing rank manipulation and thus invest in spam detection algorithms. Finally, advertising has obvious monetary and disclosure costs. Social tagging systems are therefore a target of opportunity; an abuser can submit many spam annotations effectively, efficiently, cheaply, and anonymously.

Once the spammer has attracted users to site X, the easiest and most effective way to make a profit is to place ads. The widespread adoption, low entrance cost, and ease of use of advertising platforms and networks such as Google AdSense (`www.google.com/adsense`) and Yahoo APT (`apt.yahoo.com`) have created a market for online traffic and clicks. Abuse is therefore to be expected. All the spammer needs to do is create some content, place ads, and use social sites to attract traffic. Much of this can be done automatically, and therefore cheaply. For instance tools are available to identify a set of keywords that, if used to tag the target website, are likely to generate traffic to it. Such tools include Google Trends (`trends.google.com`) and Google Keyword Tool (`adwords.google.com/select/KeywordToolExternal`). It is important at this point to briefly discuss the relationship between social spam and click fraud.

Advertising networks and keyword tools are legitimate when used as intended. If a user tags with helpful keywords a legitimate site containing ads, this is not a case of spam. We consider social spam only those abusive uses of social tagging in which misleading tags are used, and/or a fraudulent or malicious site is tagged. Examples include phishing sites, pages that install malware, and so on. Click fraud may occur when a site uses ads in a way that is inconsistent with the terms of service of the ad provider, for instance simulating clicks to generate revenue or harm a competitor. However, there is a large gray area between the extreme of blatant click fraud (which an ad network would censure) and the extreme of legitimate use.

This gray area includes target sites with fake or plagiarized content, whose exclusive purpose is to draw profit from ads. Such sites may not violate the advertising terms of service. Yet a reasonable user would consider annotations that lead to them as social spam.

Content can be manipulated to attract users to ads. From the spammer's perspective, there is no need for text to be coherent or meaningful. The only important criterion is for content to attract visitors. To this end one would generate text that contains popular keywords and that looks genuine, at least to search engines' crawlers and content analysis algorithms. The expression "original content" is sometimes used to refer to text that *looks* original enough to trick a search engine into thinking it is a genuine page written to convey information — as opposed to its hidden, exclusive purpose of attracting clicks to revenue-generating ads.

There are at least three ways to generate "original content." One is to hire cheap labor. Another approach is automated generation of text via natural language techniques. The third approach is to plagiarize the content from legitimate sources such as Wikipedia — sources abound. There is already a marketplace for such services; questionable sites like `www.adsenseready.com` allow users to download pre-made "content-rich" sites on which to place ads.

In January 2006 a security mailing list circulated a message by Charles Mann, a writer who had authored a piece on click fraud for *Wired* magazine. Mann reported on a letter that he received in response to his article. The anonymous letter writer told his story as a former author of "original content." The story detailed how a disreputable company developed software to automatically create fake Web sites to capitalize on Google AdSense. In particular, the software would automatically: (1) find relevant keywords using suggestion tools such as those mentioned above; (2) register domain names based on the keywords; (3) create hosting accounts for those domain names; (4) create complete websites full of bogus content using keywords in generic sentences, with embedded ads; (5) upload these websites to the hosting accounts for the corresponding domain names; and (6) cross-link the websites to boost PageRank. The software allowed the creation of hundreds of sites per hour, each with hundreds of pages, generating significant revenues from advertising. This testimonial clearly demonstrates that there is both an incentive and an industry to generate online junk, and consequently to promote it through social spam.

As a demonstration of the ease with which revenue can be generated this way, we wrote a simple script that generates a dynamic websites for a fake "Gossip Search Engine" (`homer.informatics.indiana.edu/cgi-bin/gossip/search.cgi`). When the user submits a celebrity name, the site acts as if it had searched for news about the query. In reality, a fake news item is created on the fly with a grammar-based text generation algorithm (`dev.null.org/dadaengine`). Additional news and images are obtained from RSS feeds and search APIs. Although the page seems to provide original content, the information presented is either fake or stolen. The script simulates the experience of navigating through pages with different content by linking back to itself with keywords scraped from other sites. The ads on the generated pages are often relevant to the contextual information (celebrity names and gossip keywords). As a result the demo generates revenues.

It is important to ask, *Who is harmed by spammers who generate fake content?* The advertiser gains, because real users click on ads and thus visit the advertiser's page, which is the desired outcome. The intermediary gains its fees from the advertiser. The publisher (spammer) of course gains its cut from the clicked ads. And one might argue that if a user clicks, he/she was ultimately interested in the ad, thus no harm is done. Paradoxically it may seem there are no victims. This leads to a lack of incentives for the intermediary to curb this kind of abuse. In reality, when bogus content is generated to play the system, good information resources become diluted in junk and become harder to be found. Search engines and folksonomies direct traffic in the wrong directions. Information consumers end up with less relevant or valuable resources. Producers of relevant resources receive less cash as a reward (lower clickthrough rate) while producers of junk receive more cash. One way to describe this is *pollution.* Virtual junk pollutes the Web environment by adding noise. Everybody but the polluters pays a price for Web pollution: search engines work less well, users waste precious time and attention on junk sites, and honest publishers lose income. The polluter spoils the Web environment for everybody else.

The above discussion provides us with a clear financial motive for social spam. Understanding the incentives for social spam is essential to the design of effective countermeasures. In the remainder of this paper we describe a detection approach whose key ingredients are a set of features directly inspired by such insight.

4. FEATURES

The first issue to address in the design of a social spam detection system is what class of objects should be seen as potential candidates for spam labeling. Spam can be injected into a tagging system at three different levels. The traditional view is to classify pages or sites as spam based on their content, that is, resources that users of the system perceive as non relevant or "junk." The problem with this perspective is its subjectivity: what is spam to one person can be interesting to another. Secondly, we can focus on spam posts, i.e., on malicious associations between resources and tags. Associating a spam resource with many and/or popular tags creates pathways that lead to that resource. This type of spam pollutes the system by creating artificial links between resources and tags that would otherwise be unrelated. This kind of pollution thus affects the measures of tag and resource similarity that are grounded in the social annotations, altering recommendation, ranking and search. Finally, one can look at user accounts created with the goal of injecting foreign content into the system. Such accounts may or may not mix legitimate content with spam, in order to mask spamming activity. Flagging users as spammers is the approach taken by some social tagging system, such as BibSonomy. This approach is intuitive and easy from an administrator's point of view, but it uses a broad brush. It may be exceedingly strict if a user happens to post one bit of questionable content on otherwise legitimate annotations.

When detecting spam, one can focus on each of the different levels mentioned above, and design features that selectively target spam resources, spam posts (tag-resource associations), or spammer users. We think that the appropriate level of resolution for classifying spam is often that of a *post.* When a resource's content is hard to classify, observing how a user annotated it should reveal the intent. Consider for example Fig. 1: different users may disagree on whether the *resource* itself is spam, but any reasonable user would recognize the *posts* as spam based on the misleading tags. Conversely, appropriate tags might suggest legitimate annotation of a questionable resource. Moreover, spam labels can be "escalated" from posts to users when necessary, by aggregating across the posts of a user. Here we define six features designed to capture social spam. Two of the features operate at the post level, three at the resource level, and one at the user level. For each feature we discuss the underlying strategy, the technique used for computing it, and the complexity entailed by such computation.

4.1 TagSpam

A simple feature can be built on the notion that taggers share a prevalent vocabulary to annotate resources [23]. Spammers may

therefore use tags and tag combinations that are statistically unlikely to appear in legitimate posts. If a body of annotations labeled for spam is available, we can use it to estimate the probability that a given tag is associated with legitimate content. The support used to define this probability depends on the granularity of spam labels. In the present case we have labels on a per-user basis (cf. 5.1). Let U_t be the set of users with tag t ($U_t = \{u : (\exists r : (u, r, t) \in F)\}$) and $S_t \subset U_t$ the subset of these users labeled as spammers. We can then define the *TagSpam* feature via the probability that a tag t is used to spam, estimated by the fraction of users tagging with t who are spammers: $\Pr(t) = |S_t|/|U_t|$. These probabilities are then aggregated across a post's tags to obtain the *TagSpam* feature. Consider a user u who annotated resource r with tags $T(u, r) = \{t : (u, r, t) \in F\}$. We define the *TagSpam* feature as:

$$f_{TagSpam}(u, r) = \frac{1}{|T(u, r)|} \sum_{t \in T(u,r)} \Pr(t).$$

This feature can be computed in constant time for any post entering the system, assuming the number of tags in a post does not grow with the number of annotations in the system. However *TagSpam* suffers from a cold start problem, since a body of labeled annotations is needed to bootstrap the tag probabilities.

4.2 TagBlur

In spam posts, the spam resource is usually associated with a large number of popular tags that may be unrelated to the resource and are often semantically unrelated to one another. This happens because spammers gain from associating the spam resource with a number of high-frequency tags, regardless of their meaning in the context of the folksonomy. This is one of the malicious behaviors described by Koutrika *et al.* [16]. We define a new feature that captures this degree of unrelatedness of tags belonging to a post, i.e., the "semantic blur" of the post. The underlying assumption is that spam posts tend to lack semantic focus.

To define a measure of semantic blur we need to build on a notion of semantic similarity for tags. To this end, we draw upon our previous work on social similarity [20, 19], where a systematic characterization of measures for tag and resource similarity was carried out and validated against a manually-created taxonomy of concepts and resources. In this prior work we have shown that mutual information is one of the best measures of tag similarity (we omit the definition in this context for brevity, see [19]). Let the similarity $\sigma(t_1, t_2) \in [0, 1]$ be defined as the mutual information between two tags t_1 and t_2, computed on the basis of a set of annotations (cf. § 5.1) and normalized into the unit interval. The *TagBlur* feature is obtained from a measure of distance (dissimilarity) between tags, averaged across all the pairs of tags in a post:

$$f_{TagBlur}(u, r) = \frac{1}{Z} \sum_{t_1 \neq t_2 \in T(u,r)} \frac{1}{\sigma(t_1, t_2) + \epsilon} - \frac{1}{1 + \epsilon}$$

where Z is the number of tag pairs in $T(u, r)$, ϵ is a constant ensuring that the distance is defined (and large) when $\sigma = 0$, and the last term ensures that the distance is zero when $\sigma = 1$. The computation of *TagBlur* is quadratic in the number of tags per post, but can still be considered constant time if the number of tags in a post does not grow with the annotations in the system. This feature relies on the availability of a precomputed similarity for any two tags in the folksonomy, as is the case in the `GiveALink.org` system.

4.3 DomFp

Let us now focus on the *content* of annotated resources, by observing that Web pages in social spam often tend to have a similar document structure, possibly due to the fact that many of them are automatically generated by tools that craft web sites from predefined templates. We want to estimate the likelihood that the content of a tagged page is generated automatically, by measuring its structural similarity to other pages from a body of annotations manually labeled as spam. We first extract a *fingerprint* of a page's DOM structure (hence the feature name *DomFp*). We strip away all the content but the HTML 4.0 elements and then build a string by mapping HTML elements to symbols while preserving their order of appearance in the page.

Assuming a body of labeled social spam, we can build a set K of fingerprints associated with spam resources. Each fingerprint $k \in K$ will have an associated frequency denoting the number of times that fingerprint is encountered in spam. Based on this frequency we can estimate the probability $\Pr(k)$ that k is associated with spam resources. The likelihood that a resource r is spam is then estimated by measuring the similarity between its fingerprint $k(r)$ and the spam fingerprints. To measure the similarity between two fingerprints k_1 and k_2 we turn to the shingles method [4]. For each fingerprint sequence we build a set of shingles (10 symbols each), and define the fingerprint similarity $\sigma(k_1, k_2) \in [0, 1]$ as the Jaccard coefficient between the sets of shingles corresponding to the two fingerprints. Let us finally define the *DomFp* feature for resource r as the normalized weighted average:

$$f_{DomFp}(r) = \frac{\sum_{k \in K} \sigma(k(r), k) \cdot \Pr(k)}{\sum_{k \in K} \sigma(k(r), k)}.$$

We interpret *DomFp* as the likelihood that resource r is spam based on its document structure. This feature requires that each resource be crawled to extract its fingerprint. We also assume that a labeled spam collection is available and spam fingerprint probabilities are precomputed. For each resource it is necessary to compute its similarity to all spam fingerprints, with complexity that grows linearly with the size of the labeled spam collection.

4.4 NumAds

Another resource feature, *NumAds*, draws upon the idea that spammers often create pages for the sole purpose of serving ads (see § 3). Ads are typically served from external resources through javascript. Since Google AdSense is the most popular ad service, we focus on it to illustrate our idea, by simply counting the number of times the Google ad server `googlesyndication.com` appears in a Web page tagged by a user. Let $g(r)$ be the number of ads in page r. We compute the *NumAds* feature as $f_{NumAds}(r) = g(r)/g_{max}$, where g_{max} is a normalization constant. For evaluation purposes we set $g_{max} = 113$, which is the maximum value of $g(r)$ over all resources in our dataset (cf. § 5.1). Computing *NumAds* requires the complete download of a Web page.

4.5 Plagiarism

Our last resource feature, *Plagiarism*, shares with *DomFp* the goal of detecting automatically generated pages. Spammers can easily copy original content from all over the Web, as discussed in § 3. To estimate the odds that a page's content is not genuine, we look for authoritative pages that are likely sources of plagiarized content. We first extract a random sequence of 10 words from the page's content. This sequence is submitted as a phrase query (using double quotes) to the Yahoo search service API (`developer.yahoo.com/download`). We can measure *Plagiarism* by the number of results returned by the search engine, excluding the originating resource's URL. Let $y(r)$ be the number of search hits different from r matching the phrase query extracted from r. We then define $f_{Plagiarism}(r) = y(r)/y_{max}$, where y_{max} is a normalization con-

stant. For evaluation purposes we limited the results from Yahoo to 10, and set $y_{max} = 10$. So $f_{Plagiarism}(r) = 1$ if $y(r) \geq 10$.

Plagiarism is the most expensive of our features. The page must be downloaded to extract the random sequence of words, then the search engine must be queried for a total of two network requests per resource. While each of the resource features requires a download, the extra request and the query limits of the Yahoo search service impose a larger burden on the computation of *Plagiarism*.

4.6 ValidLinks

Our last feature, *ValidLinks*, is defined at the level of a user and focuses on the detection of user profiles created for spam purposes. Many of the resources posted by a spammer may have questionable content (for example copyright infringing material along with ads) and be taken offline when detected by the hosting service. Other examples include malware or phishing sites. Malicious users may temporarily set up a page to obtain sensitive information, and then post the malicious resource to a social site. Once enough data is collected or the site is taken down by the hosting company, the resource disappears. Finally, a spammer may become inactive for various reasons and leave broken links in the social site. To capture these situations, we define *ValidLinks* as the fraction of a user's posts with valid resources, $f_{ValidLinks}(u) = |V_u|/|R_u|$, where $R_u = \{r : (\exists t : (u, r, t) \in F)\}$ is the set of resources tagged by user u and $V_u \subset R_u$ is the subset of these resources whose links are valid. To determine the validity of a link we send an HTTP HEAD request. This must be done for each of a user's posts, making this feature expensive to compute for users with many resources.

5. EVALUATION

5.1 Dataset Description

To evaluate the proposed features, both individually based on their discrimination power and together in support of a classifier, we need labeled examples to build training and test sets. We turn to a public dataset released by `BibSonomy.org` as part of the ECML/PKDD 2008 Discovery Challenge on Spam Detection in Social Bookmarking Systems (`www.kde.cs.uni-kassel.de/ws/rsdc08`). The dataset contains all the annotations of 27,000 users, of which 25,000 are manually labeled as spammers and the remaining 2,000 as legitimate users. Manual classification by a trusted moderator is one of the methods described in [16]. In this particular case, the criterion is that if any one resource tagged by a user is judged to be spam, then the user is labeled as a spammer.

To perform our evaluation we sampled a subset of users from the complete dataset. The sample is random, except for two sources of intentional bias. One bias is to select an equal number of spammers and legitimate users. While such a ratio of spammers is not reflective of the original dataset, we have two reasons for this balance. First, from an evaluation perspective, spammers are so predominant in the BibSonomy dataset that a baseline classifier labeling all users as spammers would achieve over 92% accuracy, making it difficult to compare different features and algorithms. Second, in a realistic setting we expect that a social bookmarking site would have a spam defense mechanism in place, so that the density of spam in the system should not be so high. We also apply a bias such that users with more posts in their profiles have a higher probability of being sampled. The resulting dataset comprises of a total of 500 users, 250 of whom are labeled spammers, with all their annotations.

Three features — *TagSpam*, *TagBlur* and *DomFp* — rely on statistics from the folksonomy, that is, we need a training set to compute them. We could split the dataset into a training set for this purpose and a test set, however this would create an imbalance

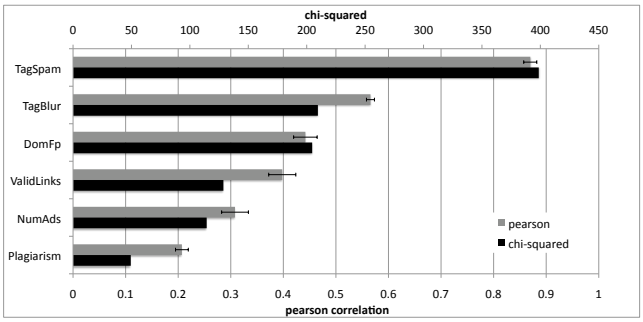

Figure 3: Discrimination power of our features with respect to spammers, as measured by Pearson's correlation and χ^2. For the former, error bars are computed by shuffling the contingency matrices.

with respect to the test set size for the other features. To keep the same test set size of 500 users for all features, we drew another independent sample from the original BibSonomy dataset using the same procedure. The annotations of these separate 500 users (again half spammers) were used to compute the values of *TagSpam*, *TagBlur*, and *DomFp* for the 500 users in our dataset. More precisely, to compute *TagBlur* we need at least one post with more than two tags, leaving us with 486 users in the original dataset with usable statistics. There are other feature requirements as well. Users without any valid resource links have *NumAds*, *DomFp*, and *Plagiarism* undefined. This yields 453 users with the *NumAds* feature defined. To extract a DOM fingerprint we need at least one crawled resource containing at least one W3C-standard HTML element, leaving 450 users with usable *DomFp* statistics. Only 446 users have at least one resource with text content allowing us to compute *Plagiarism*. Finally all 500 users have *TagSpam* and *ValidLinks* features defined.

5.2 Feature Analysis

The labels in our dataset are at the level of users, therefore we must apply user-level spam detection to be able to evaluate our features and algorithms. One of our proposed features, *ValidLinks*, is a user-level measure. The other five features defined in § 4, however, are computed at the level of posts or resources, and we must aggregate them across each user's posts for evaluation. We considered various aggregation methods: average, minimum, maximum, product (sum-log), and variance. Among these approaches, the simple average of feature values across the posts/resources of a user provides the best results in terms of user discrimination power, which we report below. In some individual feature cases other aggregation schemes performed as well as averaging, but none was as effective across all features as the simple average:

$$f(u) = \frac{1}{|P(u)|} \sum_{(u,r) \in P(u)} f(u, r)$$

where $f(u, r)$ if the value of feature f for post (u, r) (or just for resource r) and the set $P(u)$ of posts by user u is defined as $\{(u, r) : (\exists t : (u, r, t) \in F)\}$ (and similarly for resources).

To analyze the discriminating power of each feature in separating spammers from legitimate users, we first normalized each feature such that $f(u) \in [0, 1]$. We then divided the unit interval into 20 equal-size bins, building a contingency matrix $n(l, f)$ for the number of users with feature value $f(u) = f$ and spam label $l(u) = l$ ($l = 1$ for spammers, $l = 0$ otherwise). Finally we applied two standard discrimination functions, the χ^2 statistics and Pearson's correlation. Fig. 3 shows that both measures yield a consistent ranking of our six features by discrimination power. Not surprisingly, the three features that rely on statistics from the training set

are the most effective; *TagSpam* is the best predictor — spammers do tend to use certain "suspect" tags more than legitimate users. All features have a statistically significant correlation with the spam label, therefore all features are predictive. These results are robust; we repeated the analysis on a separate, independent sample of users and found the same correlations.

For each feature we also show in Fig. 4 the histograms of spammers and legitimate users from the contingency matrices. These distributions give a visual intuition for how well each feature discriminates spammers. While some features, such as *TagSpam*, clearly lend themselves to being used in linear discrimination classifiers, others such as *ValidLinks* may also be good discriminators but would require the use of nonlinear classifiers; the distributions of spammers and legitimate users are not so easily separated by a simple threshold. This is evident if we use each feature alone in conjunction with a threshold to detect spammers. When we rank users by the values of each feature we obtain the ROC curves and AUC values shown in Fig. 5. The ranking of the features by AUC is roughly consistent with the discrimination power of the features (Fig. 3) except for those features that require a nonlinear classifier, which underperform in this simple linear detection setting.

We note that our top feature, *TagSpam*, achieves alone an AUC of 0.99, which compares favorably with the best classifier from the ECML/PKDD 2008 Challenge on Spam Detection in Social Bookmarking Systems: the winner of the challenge scored an AUC of 0.98 [9]. This is very encouraging, especially when considering that we use a smaller and more balanced sample of the dataset.

5.3 Classification

Having found that all of the proposed six features have predictive power, we used various supervised learning algorithms to construct social spam detectors based on these features. We turned to the Weka software library ([22], `www.cs.waikato.ac.nz/ml/weka`) for off-the-shelf implementations of many established machine learning algorithms. To use all features we focused on the subset of 431 users (203 spammers) in our dataset for whom all six features are defined. We evaluated many of the classifiers in the Weka suite using default values for all parameters and 10-fold cross-validation. The top 30 classifiers and their performance are shown in Table 1. The best algorithm, additive logistic regression (LogitBoost), even without any tuning reaches an accuracy of almost 98% with a false positive rate below 2%. All classifiers perform very well, with accuracy over 96% and false positive rate below 5%. This attests to the effectiveness of our proposed features.

To explore the issue of feature selection and see if the accuracy can be further improved by tuning a detector's parameters, let us consider AdaBoost, a well-known and popular ensemble classifier whose performance is very close to the best (AdaBoostM1 in Table 1). For sake of comparison we also consider the linear support vector machine (SVM), another widely popular algorithm (SMO in Table 1). After tuning the parameters for both we could not improve on the default SVM, while AdaBoost's performance was enhanced by simply extending the number of iterations to 1000. AdaBoost thus achieved an accuracy of 98.4% (better than LogitBoost with default parameters), with a false positive rate of 2% and $F_1 = 0.98$ (comparable to LogitBoost). This is our best result.

The effect of feature selection is reported in Table 2 and displayed in Fig. 6. In the case of SVM we see a modest improvement in accuracy and decrease in false positive rate by using both *TagSpam* and *TagBlur*, but additional features do not help. Performance is actually hindered by the addition of the *ValidLinks* feature. This is understandable because as we have discussed above, this feature does not give a clean linear separation of spammers from legitimate users and the linear SVM is therefore unable to exploit this feature. AdaBoost, on the other hand, is able to combine evidence from all features and thus improve both accuracy and false positive rate by learning from as many features as are available.

Table 1: Top Weka classifiers, ranked by accuracy (fraction of users correctly classified). Also shown is the false positive rate (FP), related to precision and defined as the fraction of legitimate users who are wrongly classified as spammers. In a deployed social spam detection system it is more important that the false positive rate be kept low compared to the miss rate, because misclassification of a legitimate user is a more consequential mistake than missing a spammer. The F_1 measure is the harmonic mean of precision and recall. Recall is related to misses (undetected spammers). Each classifier uses default parameter values, is trained using all six features, and is validated with 10-fold cross-validation. Best scores are highlighted.

Weka Classifier	Accuracy	FP	F_1
LogitBoost	**97.91%**	.018	**.978**
LWL	97.68%	**.013**	.975
AdaBoostM1	97.68%	.018	.975
ConjunctiveRule	97.68%	**.013**	.975
DecisionTable	97.68%	.018	.975
DecisionStump	97.68%	**.013**	.975
RandomCommittee	97.45%	.018	.973
RandomForest	97.45%	.018	.973
Bagging	97.22%	.022	.970
NNge	97.22%	.022	.970
ADTree	97.22%	.026	.970
ClassificationViaRegression	96.98%	.031	.968
Decorate	96.98%	.018	.968
MultiBoostAB	96.98%	.026	.968
LMT	96.98%	.044	.969
REPTree	96.98%	.026	.968
RBFNetwork	96.75%	.031	.966
SMO	96.75%	.048	.966
JRip	96.75%	.039	.966
OneR	96.75%	.035	.966
PART	96.75%	.039	.966
J48	96.75%	.039	.966
BayesNet	96.52%	.039	.963
Logistic	96.29%	.044	.961
VotedPerceptron	96.29%	.053	.961
NaiveBayes	96.06%	.035	.958
NaiveBayesSimple	96.06%	.035	.958
NaiveBayesUpdateable	96.06%	.035	.958
MultilayerPerceptron	96.06%	.035	.958
SimpleLogistic	96.06%	.048	.959

Table 2: Performance of linear SVM and AdaBoost social spam detectors as we select features in the order of their discrimination power (Fig. 3). Best performance (highlighted) is when all features are used.

	SVM			AdaBoost		
Features	Accuracy	FP	F_1	Accuracy	FP	F_1
TagSpam	95.82%	.061	.957	94.66%	.048	.943
+ TagBlur	96.75%	.048	.966	96.06%	.044	.958
+ DomFp	96.75%	.048	.966	96.06%	.044	.958
+ ValidLinks	96.52%	.048	.964	96.75%	.026	.965
+ NumAds	96.52%	.048	.964	97.22%	.026	.970
+ Plagiarism	96.75%	.048	.966	**98.38%**	**.022**	**.983**

6. DISCUSSION AND CONCLUSION

Our discussion of the incentives of social spam has motivated the design of a number of novel features to detect spammers who abuse social bookmarking systems. These features all have strong dis-

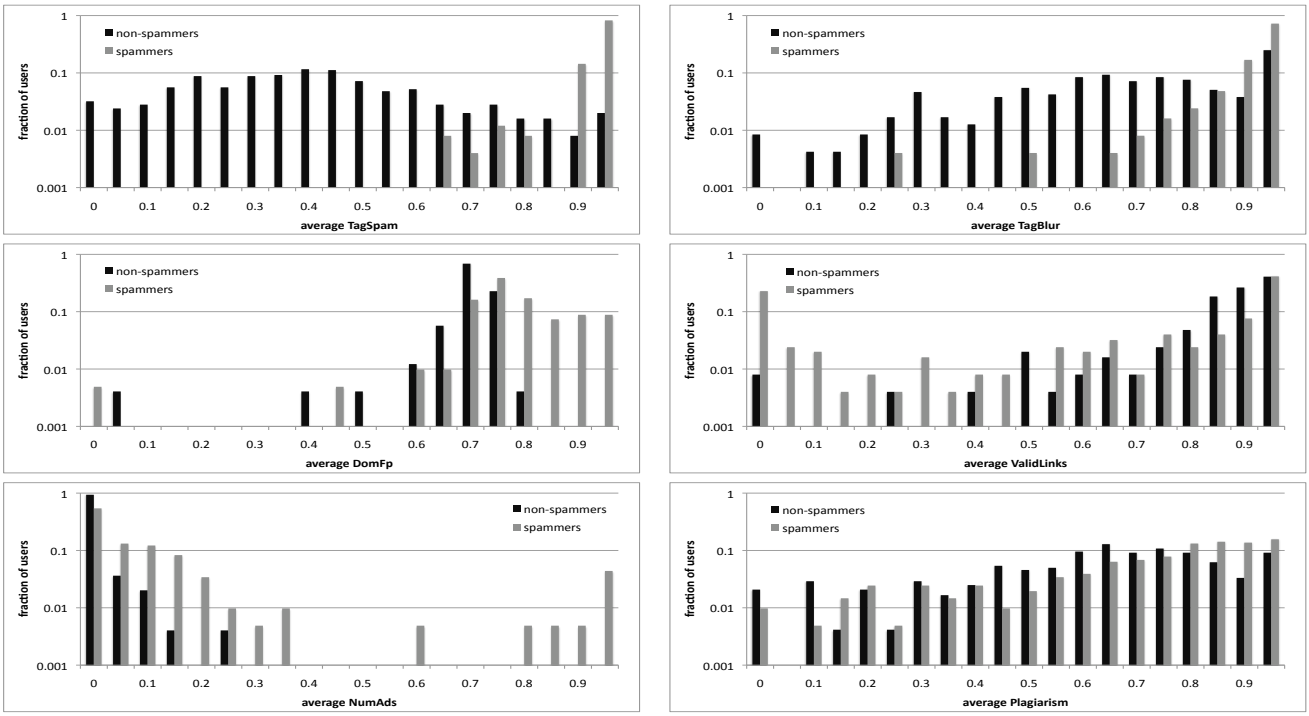

Figure 4: Distributions of the values of the six proposed features for spammers vs. legitimate users. Each feature's distribution is based on the subset of users for which the feature is defined (see text).

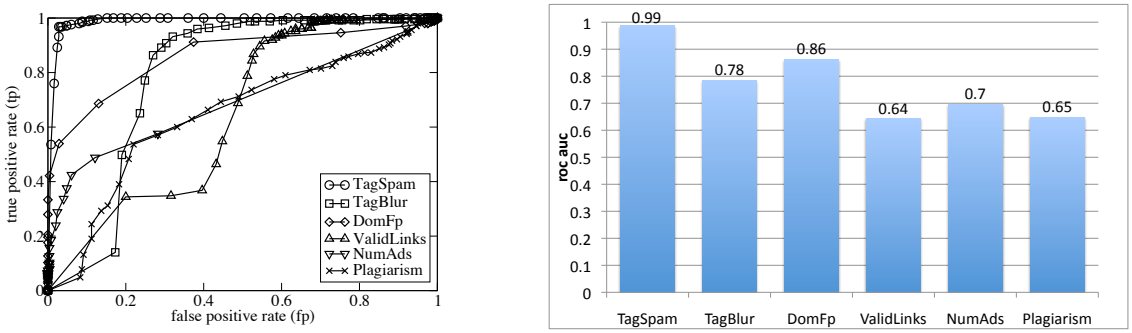

Figure 5: Left: ROC curves for each of the proposed features to detect spammers. A true positive is a correctly classified spammer, a false positive is a legitimate user incorrectly classified as a spammer. Right: Areas under the ROC curves (AUC) for the six features. Larger AUC denotes better detection trade-off between true and false positives for the linear classifier based on the corresponding feature.

criminating power in detecting spammers; using just one of them allows to correctly classify about 96% of users with a linear discriminant function, while combining all six features boosts the accuracy to over 98% with an ensemble classifier. At the same time the false positive rate can be kept below 5% with a single feature and pushed down to 2% combining all features. These promising results provide a new baseline for future efforts on social spam. A natural extension is to combine our features with those of Krause *et al.* [17] to see if performance can be further improved. To facilitate such efforts we have made our dataset freely available to the research community at `GiveALink.org/socialspam`.

In an effort to bring these findings to bear on contemporary tagging systems by making them more robust with respect to social spam, we are currently working on the integration of a spam detection system — such as described here — into `GiveALink.org`, a social annotation system that we develop and maintain for research purposes. Of course spam detection is just one of the measures that must be put in place to defend a social bookmarking system from spammers. Other measures include prevention (e.g. through captchas) and mitigation (e.g. through ranking algorithms that penalize tags used by spammers) [13].

When deploying a spam detection system in a live social tagging site it is not necessary to aggregate post-level features into user-level features, as was done here due to evaluation dataset constraints. We plan to experiment with the option of detecting individual spam posts rather than classifying users. The detection system can be used in many ways: to filter posts, to flag posts to a moderator, or to flag users, say when a significant portion of their posts is deemed to be spam. It remains to be seen whether the finer resolution of posts will lead to increased effectiveness (by decreasing false positives) or to an encouragement of non-social behavior.

From an efficiency/feasibility perspective, the *TagBlur* feature looks promising. While not quite as predictive as *TagSpam*, it relies on tag-tag similarity measures, which can be maintained up-to-date with incremental techniques [20, 19] and therefore are always available. Other features rely on the availability of infrastructure

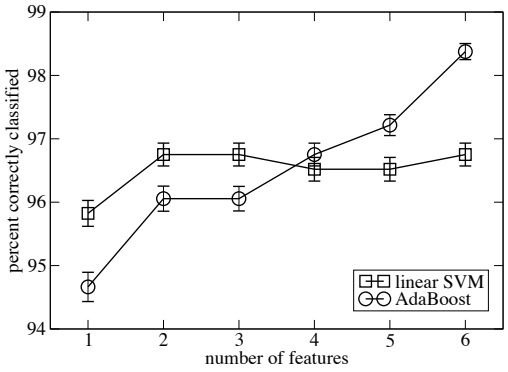

Figure 6: Accuracy of linear SVM and AdaBoost social spam detectors with features select in order of discrimination power (see Fig. 3). Error bars are based on root mean squared error reported by Weka.

to enable access to resource content (e.g. *DomFp*, *NumAds*) or the cooperation of a search engine (e.g. *Plagiarism*) and thus their feasibility depends on the circumstances of a particular social annotation system. Another efficiency issue, not explored here, is feature selection on the tags. The computation of features such as *TagSpam* and *TagBlur* might be greatly accelerated by focusing on a subset of the tags in the folksonomy, for example the most frequently used.

Bootstrap is an open issue. In the absence of spam labels, or until these may become available through a user feedback mechanism, we need spam assessments to compute the features that rely on tag spam statistics, such as *TagSpam* and *DomFp*. One approach we are exploring is to bootstrap the probabilities necessary to compute these features using the other features, which do not require supervision, on a sample of annotations. There are also indirect dependencies on labeled data, however. *TagBlur* relies on tag similarity, which is assumed to be computed on legitimate annotations. An abundance of spam in the folksonomy would bias the tag similarity values making *TagBlur* less effective. For example if "software" and "sex" co-occur often (due to spam posts), the system could wrongly conclude that these tags are related, missing posts such as those in Fig. 1. Therefore the similarity computations must exclude posts labeled as spam. In general, a deployed social spam detection system must be incrementally trained. As similarities are updated and spam labels collected in response to newly received annotations, the feature values of incoming posts that depend on these assessments need to be computed with the latest statistics to keep the detector fresh. A question for future research is whether after the initial bootstrap phase, unsupervised features such as *ValidLinks* or *Plagiarism* should continue to be used — along with user input — to update the supervised features, in semi-supervised fashion.

Given the current financial incentives for social spam, we have no doubt that this is but one early chapter in an escalating arms race to combat the emerging phenomena of Web 2.0 abuse.

Acknowledgments

We are indebted to G. Stumme, A. Hotho, D. Benz, B. Krause, N. Street, and the Networks & agents Network at the IU School of Informatics for helpful discussions. We gratefully acknowledge the BibSonomy team and the European Commission's TAGora project (`tagora-project.eu`) for access to the Bibsonomy spam dataset. Thanks to the ISI Foundation for support; the ideas presented here hatched while FM and BM were visiting ISI. This work is partly funded by NSF Award IIS-0811994.

7. REFERENCES

[1] J. Attenberg and T. Suel. Cleaning search results using term distance features. In *Proc. 4th Intl. Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 21–24, 2008.

[2] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, C. Zhang, and K. Ross. Identifying video spammers in online social networks. In *Proc. 4th Intl. Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 45–52, 2008.

[3] J. Bian, Y. Liu, E. Agichtein, and H. Zha. A few bad votes too many?: towards robust ranking in social media. In *Proc. 4th Intl. Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 53–60, 2008.

[4] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8–13):1157–1166, 1997. Proc 6th Intl. WWW Conf.

[5] C. Cattuto, D. Benz, A. Hotho, and G. Stumme. Semantic grounding of tag relatedness in social bookmarking systems. In *Proc. ISWC*, vol. 5318 of *LNCS*, pages 615–631, 2008.

[6] C. Cattuto, C. Schmitz, A. Baldassarri, V. D. P. Servedio, V. Loreto, A. Hotho, M. Grahl, and G. Stumme. Network properties of folksonomies. *AI Commun.*, 20(4):245–262, 2007.

[7] J. Caverlee, L. Liu, and S. Webb. Socialtrust: tamper-resilient trust establishment in online communities. In *Proc. 8th ACM/IEEE-CS Joint Conf. on Digital libraries (JCDL)*, pages 104–114, 2008.

[8] J. Chevalier and P. Gramme. RANK for spam detection ECML - Discovery Challenge. In *Proc. Europ. Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2008.

[9] A. Gkanogiannis and T. Kalamboukis. A novel supervised learning algorithm and its use for spam detection in social bookmarking systems. In *Proc. Europ. Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2008.

[10] S. Golder and B. A. Huberman. The structure of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, April 2006.

[11] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proc. 13th Intl. Conv Very Large Data Bases (VLDB)*, pages 576–587, 2004.

[12] T. Hammond, T. Hannay, B. Lund, and J. Scott. Social Bookmarking Tools (I): A General Review. *D-Lib Magazine*, 11(4), April 2005.

[13] P. Heymann, G. Koutrika, and H. Garcia-Molina. Fighting spam on social web sites: A survey of approaches and future challenges. *IEEE Internet Computing*, 11(6):36–45, 2007.

[14] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *The Semantic Web: Research and Applications*, vol. 4011 of *LNAI*, pages 411–426. Springer, 2006.

[15] C. Kim and K.-B. Hwang. Naive bayes classifier learning with feature selection for spam detection in social bookmarking. In *Proc. Europ. Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2008.

[16] G. Koutrika, F. A. Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina. Combating spam in tagging systems. In *Proc. 3rd Intl. Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 57–64, 2007.

[17] B. Krause, C. Schmitz, A. Hotho, and G. Stumme. The anti-social tagger: detecting spam in social bookmarking systems. In *Proc. 4th Intl. Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 61–68, 2008.

[18] R. Lambiotte and M. Ausloos. Collaborative tagging as a tripartite network. *LNCS*, 3993:1114, Dec 2005.

[19] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *Proc. 18th Intl. WWW Conf.*, 2009.

[20] B. Markines, H. Roinestad, and F. Menczer. Efficient assembly of social semantic networks. In *Proc. 19th ACM Conf. on Hypertext and Hypermedia (HT)*, pages 149–156, 2008.

[21] P. Mika. Ontologies are us: A unified model of social networks and semantics. In *Proc. ISWC*, vol. 3729 of *LNCS*, pages 522–536, 2005.

[22] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition, 2005.

[23] Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. In *Proc. WWW'06 Collaborative Web Tagging Workshop*, 2006.