Practical Detection of Spammers and Content Promoters in Online Video Sharing Systems

Fabrício Benevenuto[†], Tiago Rodrigues[‡], Adriano Veloso[‡],
Jussara Almeida[‡], Marcos Gonçalves[‡] and Virgílio Almeida[‡]
[†]Computer Science Department, Federal University of Ouro Preto, Brazil
[‡]Computer Science Department, Federal University of Minas Gerais, Brazil
{*fabricio, tiagorm, adrianov, jussara, mgoncalv, virgilio*}*@dcc.ufmg.br*

***Abstract*—A number of online video sharing systems, out of which YouTube is the most popular, provides features that allow users to post a video as a response to a discussion topic. These features open opportunities for users to introduce polluted content, or simply pollution, into the system. For instance, *spammers* may post an unrelated video as response to a popular one, aiming at increasing the likelihood of the *response* being viewed by a larger number of users. Moreover, *content promoters* may try to gain visibility to a specific video by posting a large number of (potentially unrelated) responses to boost the rank of the *responded video*, making it appear in the top lists maintained by the system. Content pollution may jeopardize the trust of users on the system, thus compromising its success in promoting social interactions. In spite of that, the available literature is very limited in providing a deep understanding of this problem.**

In this paper, we address the issue of detecting video spammers and promoters. Towards that end, we first manually build a test collection of real YouTube users, classifying them as spammers, promoters, and legitimate users. Using our test collection, we provide a characterization of content, individual, and social attributes that help distinguish each user class. We then investigate the feasibility of using supervised classification algorithms to automatically detect spammers and promoters, and assess their effectiveness in our test collection. While our classification approach succeeds at separating spammers and promoters from legitimate users, the high cost of manually labeling vast amounts of examples compromise its full potential in realistic scenarios. For this reason, we further propose an active learning approach that automatically chooses a set of examples to label, which is likely to provide the highest amount of information, drastically reducing the amount of required training data while maintaining comparable classification effectiveness.

I. Introduction

With Internet video sharing sites gaining popularity at a dazzling speed, the Web is being transformed into a major channel for the delivery of multimedia content. Online video social networks, out of which YouTube is the most popular, are distributing videos at a massive scale. It has been reported that the amount of content uploaded to YouTube in 60 days is equivalent to the content that would have been broadcasted for 60 years, without interruption, by NBC, CBS and ABC altogether [2]. Moreover, YouTube has reportedly served over 100 million users only in January 2009 [1], with a video upload rate equivalent to 10 hours per minute [3].

By allowing users to publicize and share their independently generated content, online video social networks become susceptible to different types of non-cooperative (e.g., malicious and opportunistic) user actions. Particularly, these systems usually offer three basic mechanisms for video retrieval: (1) a search system, (2) ranked lists of top videos, and (3) social links connecting users and/or videos. Although appealing as mechanisms to facilitate content location and enrich online interaction, these mechanisms open opportunities for users to introduce polluted content into the system. As an example, video search systems can be fooled by malicious attacks in which users post their videos with several popular tags [32]. Opportunistic behavior on the other two mechanisms for video retrieval can be exemplified by observing a YouTube feature that allows users to post a video as a response to a video topic. Some users, which we call *spammers*, post unrelated videos as responses to popular video topics aiming at increasing the likelihood of the *responses* being viewed by a larger number of users. Other users, to whom we refer as *promoters*, may try to gain visibility towards a specific video by posting a large number of (potentially unrelated) responses to boost the rank of the *video topic* among the most responded videos, making it appear in the top lists maintained by YouTube. Promoters and spammers are driven by several goals, such as spread advertise to generate sales, disseminate pornography, or simply compromise system reputation.

Polluted content may compromise user patience and satisfaction with the system since users cannot easily identify the pollution before watching at least a segment of it, which also consumes system resources, especially bandwidth. Additionally, promoters may further negatively impact system mechanisms related to content distribution, since promoted videos that quickly reach high rankings are strong candidates to be kept in caches or in content distribution networks [13].

In this paper, we address the issue of detecting video spammers and promoters adopting a 5-step approach. First, we crawled a large user dataset from YouTube site, containing more than 260 thousand users. Second, we sampled our user dataset to create a labeled test collection consisting of 829 users, which were "manually" classified as legitimate, spammers and promoters. Our sampling was performed so as to capture different profiles of users in each category. Third, we analyzed a variety of video, individual and social attributes that reflect the behavior of our sampled users, aiming at drawing some insights into their relative discriminatory power in distinguishing legitimate users, promoters and spammers.

Fourth, using the same set of attributes, which are based on the user's profile, the user's social behavior in the system, and the videos posted by the user as well as her target (responded) videos, we investigated the feasibility of applying supervised learning methods for identifying the two envisioned types of polluters. We consider two state-of-the-art supervised classification algorithms, namely, Support Vector Machines (SVM) [31] and Lazy Associative Classification (LAC) [44]. We evaluated both algorithms over our test collection, finding that both techniques can effectively identify the majority of the promoters and spammers.

However, despite effective, supervised solutions usually rely on manually labeled training data to learn patterns capable of identifying specific behaviors (e.g., spamming). Manually labeling large amounts of training data, specifically in the case of video sharing systems, is very costly. For instance, in order to manually create our user test collection with 829 users, volunteers had to watch around 20,000 videos (including video responses and video topics). Thus, the high cost of manually labeling vast amounts of examples compromise its full potential on a practical and realistic scenario.

For this reason, we propose an active learning approach which automatically chooses a set of examples to be labeled that is likely to provide the highest amount of information. This approach allows us to drastically reduce the labeling effort (in up to 81%) while maintaining a similar classification effectiveness, making our algorithm very suitable for practical scenarios.

The rest of the paper is organized as follows. Next section discusses related work. Section III describes our crawling strategy and the test collection built from the crawled dataset. Section IV investigates a set of user attributes and their ability to distinguish promoters, spammers and legitimate users. The supervised classification algorithms used as basis for our polluter detection methods along with a novel active learning approach are presented in Section V. Section VI provides an experimental evaluation of effectiveness of these learning methods on our test collection. Finally, Section VII offers conclusions and directions for future work.

II. Related Work

Content pollution has been observed in various applications, including e-mail [24], Web search engines [19], blogs [41], Peer-to-Peer systems [5], [35], microblogs [25], [15], and social networks [21]. Thus, a number of detection and combating strategies have been proposed [12], [27], [47], [36], [16], [6]. Several of them rely on extracting pieces of evidence from *textual descriptions* of the content, treating the text corpus as a set of objects with associated attributes, and applying a classification technique to detect spam [28].

Another interesting approach to prevent spam consists of white-listing users so that each user specifies a list of users from whom they are willing to receive content. "RE" [22] is a white-listing system for email based on social links that allows emails between friends and friends-of-friends to bypass standard spam filters. Socially-connected users provide secure attestations for each others' email messages while keeping users' contacts private. More recently, Mislove *et al.* [39] proposed Ostra, a mechanism that imposes an upfront cost to senders for each communication.

In another form of attack, opportunistic users associate unrelated metadata information to objects of the system aiming at improving the visibility of the content [8]. The main problem caused by this type of attack is related to content retrieval, as most information retrieval mechanisms rely mainly on metadata. If a multimedia content is not well described by its metadata (i.e. title, description, tags), it cannot be found in search results or appear in lists of related content. Additionally, recommendation mechanisms based on metadata may suggest undesirable videos. Lastly, users may access content by mistake, consuming extra resources. To tackle this problem, Koutrika *et al.* investigated strategies to detect spam in tag-based systems [32]. The authors proposed metrics to evaluate the level of pollution in a tag system. Through simulation, they analyzed several "what if" questions regarding the fraction of malicious users in the system, the number of unrelated tags created, etc. As conclusion, they showed that a trust moderator can help reducing tag spam, but it may take a significant effort in order to have a positive impact. More recently, Gao *et al.* [21] characterized spam on Facebook and proposed a clustering-based strategy to identify spammers.

Our proposal is complementary to these efforts for two reasons. First, it aims at detecting *users* who disseminate video pollution, instead of classifying the content itself. Content-based classification would require combining multiple pieces of evidence extracted from textual descriptions of the video (e.g., tags, title) and from the video content itself, which, in turn, would require more sophisticated multimedia information retrieval methods that are robust to the typically low quality of user-generated videos [10]. Instead, we use attributes that capture the feedback of users with respect to each other or to their contributions to the system (e.g., number of views received), exploiting their interactions through video responses. Second, we use a machine learning approach that explores the characteristics of pre-classified users to create models able to identify spammers and promoters. There have been other efforts that sucessfully rely on machine learning to detect spam. As examples, Castillo *et al.* [12] proposed a framework to detect Web spamming that explores social network metrics extracted from the Web graph. Benevenuto *et al.* [6] approached the problem of detecting spammers on Twitter, whereas Lee *et al.* [34] created honeypots (i.e., fake accounts that might be followed/contacted by spammers) to identify and study spammers on MySpace and Twitter.

Despite these previous studies, to the best of our knowledge, we are the first to tackle the problem of detecting spammers and content promoters in online video sharing systems. In previous studies, we analyzed the properties of the social network created by video response interactions in YouTube, finding evidence of pollution [9], [8]. We approached the problem by creating the test collection presented in this work and applying a supervised classification strategy to detect spammers and promoters [7]. In comparison with our previous work [7], we here present a much more thorough investigation of classification approaches to detect spammers and promoters,

proposing an active learning algorithm that greatly reduces the required amount of training data without significant losses in classification effectiveness.

Finally, our study is also complementary to other studies of the properties of video sharing systems. As examples, Cha *et al.* performed an in-depth analysis of popularity distribution, popularity evolution, and content characteristics of YouTube and of a popular Korean service [13]. Gill *et al.* [23] characterized YouTube traffic collected from an university campus network, comparing its properties with those previously reported for other workloads, whereas Figueiredo *et al.* [20] characterized the growth patterns and the sources of popularity of YouTube videos.

III. User Test Collection

In order to evaluate our approaches to detect video spammers and promoters in online video sharing systems, we need a test collection of users pre-classified into the target categories: spammers, promoters and, legitimate users. However, no such collection is publicly available for any video sharing system, thus requiring us to build one.

Before presenting the steps taken to build our user test collection, we introduce some notations and definitions. We say a YouTube video is a *responded video* or a *video topic* if it has at least one video response. Similarly, we say a YouTube user is a *responsive user* if she has posted at least one video response, whereas a *responded user* is someone who posted at least one responded video. Moreover, we define as *spammer* a user who posts at least *one* video response that is considered unrelated to the responded video (i.e., a spam). Examples of video spams are: (i) an advertisement of a product or website completely unrelated to the subject of the responded video, and (ii) pornographic advertisement posted as response to videos that are popular among teenagers. A *promoter* is defined as a user who posts a large number of video responses to a *responded video*, aiming at promoting the *video topic*. As an example, we found promoters in our dataset who posted a long sequence (e.g., 100) of (unrelated) video responses, often without content (0 second) to a single video. A user that is neither a spammer nor a promoter is considered legitimate. The term *polluter* is used to refer to either a spammer or a promoter.

We build our user test collection by first crawling YouTube, one of the most popular social video sharing systems [1] (Section III-A). Next, we carefully select and manually classify a subset of the crawled users (Section III-B).

A. Crawling YouTube

Our strategy consists of collecting a sample of users who participate in interactions through video responses, i.e, who post or receive video responses. These interactions can be represented by a *video response user graph* $G = (X, Y)$, where X is the union of all users who posted or received video responses until a certain instant of time, and (x_1, x_2) is a directed arc in Y if user $x_1 \in X$ has responded to a video contributed by user $x_2 \in X$. In order to sample YouTube data, we designed a distributed crawling framework similar to the one presented in [14]. Our distributed crawler is composed of a master node and a number of slave nodes. The master node maintains a centralized list of identifiers of users to be crawled, which is initialized with a set of seeds. The master is also responsible for coordinating the operation of the slaves, sending non-overlapping subsets of this list to them, thus preventing redundant crawling. The slaves, after obtaining the user identifiers from the master, crawl YouTube following Algorithm 1, returning all new users collected to the master, and waiting, at the end, for a signal from it. The master, in turn, eliminates duplicates or previously crawled users from the received lists, and, in case there are still uncrawled users, starts a new round of crawling by sending new user identifiers to the slaves. Otherwise, the master sends a signal for termination to the slaves and stops execution. The sampling starts from a set of 88 seeds, consisting of the owners of the top-100 most responded videos of all time, provided by YouTube. We ran our crawler using 10 Linux boxes (1 master and 9 slaves) located at the Federal University of Minas Gerais (*Universidade Federal de Minas Gerais*) in Brazil.

Algorithm 1 Video Response Crawler (run by slave nodes)

```
   Input: A list L of users received from master node
 1: for each user U in L do
 2:     Collect U's information and video list;
 3:     for each video V in U's video list do
 4:         Collect information of V;
 5:         if V is a responded video then
 6:             Collect information of V's video responses;
 7:             Insert the responsive users in list of new users NL;
 8:         end if
 9:         if V is a video response then
10:             Insert the responded user in list of new users NL;
11:         end if
12:     end for
13: end for
14: Return NL to the master node;
```

The crawler ran for one week (01/11-18, 2008), gathering a total of **264,460** users, **381,616** responded videos and **701,950** video responses. The crawler followed links of responded videos and video responses, gathering information on various attributes of their contributors (users), including attributes of all responded videos and video responses posted by them. Particularly, for each video that was crawled, we collected a number of pieces of information, including video identifier, video owner (i.e., contributor) identifier, title, category, description, tags, upload time, video duration, number of ratings, average rating, number of views, number of users who set the video as favorite, number of comments received and number of video responses received. We also collected statistics about the author of the video responses of each video, and the sequence order in which the video responses were posted.

The crawling process was terminated only after an entire weakly connected component (WCC) of graph (X, Y) was collected. WCC is a maximal subgraph of a directed graph such that for every pair of vertices u, v in the subgraph, there is an undirected path from u to v. Collecting the WCC is a common approach to sample social network graphs, since it tends to include the most active users. Additionally, Mislove

et al. [38] showed that the users not included in the largest WCC tend to be either part of very small, isolated clusters or are not connected to other users at all. The dataset collected is used to build our test collection, as described next.

B. Building a User Test Collection

The main goal of creating a user test collection is to study the patterns and characteristics of each user class. Thus, desired properties for our test collection are: (1) having a significant number of users of all three categories; (2) including, but not restricting to, spammers and promoters which are aggressive in their strategies and generate large amounts of pollution in the system; and (3) including a large number of legitimate users with different behavioral profiles. We argue that these properties may *not* be achieved by simply randomly sampling the collection. The reasons for this are twofold. First, as we previously reported [8], spammers and promoters correspond to a very small fraction of users. Thus, randomly selecting a number of users from the crawled data would lead us to a small number of spammers and promoters, which can compromise the creation of effective training and test datasets for our analysis. Moreover, research has shown that the sample does not need to follow the class distribution in the collection in order to achieve effective classification [45]. Second, randomly selecting legitimate users may lead to a large number of users with similar behavior (i.e., users who post one video response to a discussed topic), not including examples with different profiles.

Aiming at capturing the aforementioned properties, we first defined three strategies for user selection. These strategies are discussed below. Next, we developed a website to help volunteers to analyze each video posted by each selected user, and then manually label her as either legitimate, spammer or promoter. Videos removed by their owners or private videos represented a negligible fraction of the analyzed videos (less than 1%) and were not considered. Note that this manual classification relies on human judgment to decide whether a video is related to another. In order to minimize the impact of human error, three volunteers analyzed all video responses of each selected user in order to independently classify her into one of the three categories. In case of tie, a fourth independent volunteer was heard. Each user was classified based on majority voting. We note that the volunteers agreed in about 97% of the analyzed videos, which reflects a high level of confidence to this human classification We note also that volunteers were instructed to favor legitimate users. For instance, if one was not confident that a video response was unrelated to the responded video, she should consider it to be related. Moreover, video responses containing people chatting or expressing their opinions were classified as *not spam*, as we chose not to evaluate the expressed opinions.

The three user selection strategies we adopted are:

1) In order to select users with different levels of interaction through video responses, we first defined four groups of users based on their in- and out-degrees in the video response user graph (Section III-A). Group 1 consists of users with low (≤ 10) in- and out-degrees, and thus who respond to and are responded by only a few other users. Group 2 consists of users with high (> 10) in-degree and low out-degree, and thus receive video responses from many others but post responses to only a few users. Group 3 consists of users with low in-degree and high out-degree, whereas very interactive users, with high in and out-degrees, fall into group 4. One hundred users were randomly selected from each group[1], and manually classified, yielding a total of 382 legitimate users, 10 spammers, and no promoter. The remaining 8 users were discarded as they had had their accounts suspended due to violation of terms of use.
2) Aiming at populating the test collection with polluters, we searched for them where they are more likely to be found. We first note that, in YouTube, a video v can be posted as response to at most *one* video at a time (unless one creates a copy of v and uploads it with a different ID). Thus, it is more costly for spammers to spread their video spam in YouTube than it is, for instance, to disseminate spam by e-mail. Based on this observation, we conjecture that spammers would post their video responses more often to popular videos so as to make each spam visible to a larger community of users. Moreover, some video promoters might eventually be successful and have their target videos listed among the most popular videos. Thus, we browsed the video responses posted to the top 100 most responded videos of all time, selecting a number of *suspect* users[2]. The classification of these suspect users led to the introduction of 7 new legitimate users, 118 new spammers, and 28 new promoters in our test collection.
3) To minimize a possible bias introduced by strategy (2), we *randomly* selected 300 users who posted video responses to the top 100 most responded videos of all time, finding 252 new legitimate users, 29 new spammers and 3 new promoters (16 users with closed accounts were discarded).

In total, our test collection contains 829 users, including 641 classified as legitimate users, 157 as spammers and 31 as promoters[3]. Those users posted 20,644 video responses to 9,796 unique responded videos. Our user test collection aims at supporting research on detecting spammers and promoters. Since the user classification labeling process relies on human judgement, which implies in watching a very large number of videos, the number of users in our test collection is somewhat limited. As future work, we plan to study collaborative ways to increase the size of this test collection.

IV. Analyzing User Behavior Attributes

Legitimate users, spammers and promoters have different goals in the system, and, thus, we expect them also to differ in terms of how they behave (e.g., who they interact with,

[1]Groups 1, 2, 3 and 4 have 162,546, 2,333, 3,189 and 1,154 users. Thus, homogeneous random selection from each one yields a bias towards group 4.

[2]As an example, the owner of a video with a pornographic picture as thumbnail but posted to a political debate video discussion topic.

[3]Our test collection as well as instructions to use it are available at http://homepages.dcc.ufmg.br/∼fabricio/testcollectionsigir09.html

which videos they post) to achieve their purposes. Thus, our next step is to analyze a large set of attributes that reflect user behavior in the system aiming at investigating their relative discriminatory power to distinguish one user class from the others. We consider three attribute sets, namely, video attributes, user attributes, and social network (SN) attributes.

Video attributes capture specific properties of the videos uploaded by the user, i.e., each user has a set of videos in the system, each one with attributes that may serve as indicators of its "quality", *as perceived by others*. In particular, we characterize each video by its duration, numbers of views and of commentaries received, ratings, number of times the video was selected as favorite, as well as numbers of honors and of external links. Moreover, we consider three separate groups of videos owned by the user. The first group contains aggregate information of *all videos* uploaded by the user, being useful to capture how others see the (video) contributions of this user. The second group considers only *video responses*, some of which may be pollution. The last group considers only the *responded videos* to which this user posted video responses (referred to as *target* videos). For each video group, we consider the average and the sum of the aforementioned attributes, summing up 42 video attributes for each user, all of which can be easily derived from data maintained by YouTube. We explicitly choose not to add any attribute that would require processing the multimedia content itself.

The second set of attributes consists of individual characteristics of user behavior. We expect that legitimate users spend more time doing actions such as selecting friends, adding videos as favorites, and subscribing to content updates from others. Thus, we select the following 10 user attributes: number of friends, number of videos uploaded, number of videos watched, number of videos added as favorite, numbers of video responses posted and received, numbers of subscriptions and subscribers, average time between video uploads, and maximum number of videos uploaded in 24 hours.

The third set of attributes captures the social relationships established via video response interactions, which is one of the several possible social networks on YouTube. The idea is that these attributes might capture specific interaction patterns that can help differentiating legitimate users, promoters, and spammers. We selected the following node attributes extracted from the video response user graph, which capture the level of (social) interaction of the corresponding user: clustering coefficient, betweenness, reciprocity, assortativity, and UserRank.

The clustering coefficient of node i, $cc(i)$, is the ratio of the number of existing edges between i's neighbors to the maximum possible number, and captures the communication density between the user's neighbors. The betweenness is a measure of the node's centrality in the graph, that is, nodes appearing in a larger number of the shortest paths between any two nodes have higher betweenness than others [40]. The reciprocity $R(i)$ of node i measures the probability of the corresponding user u_i receiving a video response from each other user to whom she posted a video response, that is, $R(i) = \frac{|OS(i) \cap IS(i)|}{|OS(i)|}$, where $OS(i)$ is the set of users to whom u_i posted a video response, and $IS(i)$ is the set of users who posted video responses to u_i. Node assortativity is defined, as in [12], as the ratio between the node (in/out) degree and the average (in/out) degree of its neighbors. We compute node assortativity for the four types of degree-degree correlations (i.e., in-in, in-out, out-in, out-out). Finally, we also applied the PageRank [11] algorithm, commonly used to assess the popularity of a Web page [33], to our video response user graph. The computed metric, which we refer to as UserRank, indicates the degree of participation of a user in the system through interactions via video responses. In total, we selected 8 social network attributes.

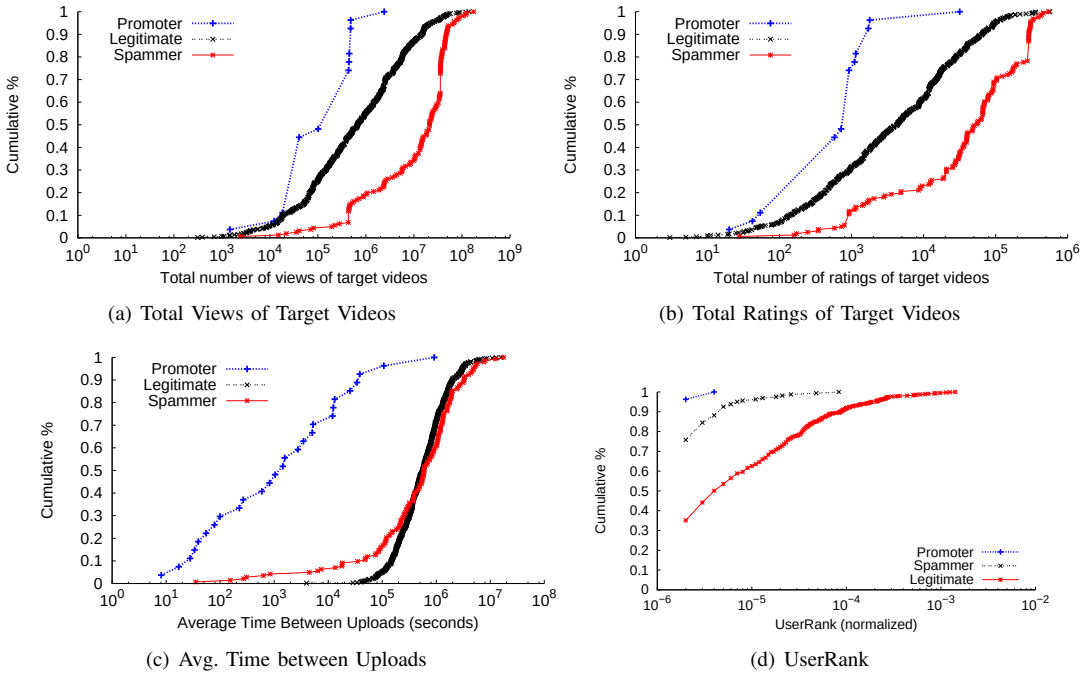
We assessed the relative power of the 60 selected attributes in discriminating one user class from the others by independently applying two well known feature selection methods, namely, information gain and χ^2 (Chi Squared) [49]. We used the implementation of the methods provided on Weka [46], which automatically discretizes all numeric features using its default discretization algorithm [18]. Table I summarizes the results, showing the number of attributes from each set (video, user, and social network) in the top 10, 20, 30, 40, and 50 most discriminative attributes according to the ranking produced by χ^2. Results for information gain are very similar and, thus, are omitted.

Attribute Set	Top 10	Top 20	Top 30	Top 40	Top 50
Video	9	18	25	30	36
User	1	2	4	7	9
SN	0	0	1	3	5

TABLE I
NUMBER OF ATTRIBUTES AT TOP POSITIONS IN χ^2 RANKING

Note that 9 out of the 10 most discriminative attributes are video-related. In fact, the most discriminative attribute (according to *both* methods), is the total number of views (i.e., the popularity) of the *target* videos. Figure 1(a) presents the cumulative distributions of this metric for each user class, showing a clear distinction among them. The curve for spammers is much more skewed towards a larger number of views, since these users tend to target popular videos in order to attract more visibility to their content. In contrast, the curve for promoters is more skewed towards the other end, as these users tend to target videos that are *still* not very popular, aiming at raising their visibility. Legitimate users, being driven mostly by social relationships and interests, exhibit an intermediary behavior, targeting videos with a wide range of popularity. The same distinction can be noticed for the distributions of the total ratings of target videos, shown in Figure 1(b), another metric that captures user feedback with respect to these videos, and is among the top 10 most discriminative attributes.

The most discriminative user and social network attributes are the average time between video uploads and the UserRank, respectively. In fact, Figure 1(c) and (d) show that, in spite of appearing in lower positions in the ranking, particularly for the UserRank attribute (see Table I), these two attributes have potential to be able to separate user classes apart. In particular, the distribution of the average time between video uploads clearly distinguishes promoters, who tend to upload at a much higher frequency since their success depends on them posting



(a) Total Views of Target Videos

(b) Total Ratings of Target Videos

(c) Avg. Time between Uploads

(d) UserRank

Fig. 1. Cumulative Distribution of User Behavior Attributes

as many video responses to the target as possible. Figure 1(c) also shows that, at least with respect to this user attribute, spammers can not be clearly distinguished from legitimate users. Finally, Figure 1(d) shows that legitimate users tend to have much higher UserRank values than spammers, who, in turn, have higher UserRank values than promoters. This indicates that, as expected, legitimate users tend to have a much more participative role (system-wide) in the video response interactions than users from the other two classes, which are much more selective when choosing their targets.

V. Detection Approaches

In this section, we present the machine learning techniques we use to automatically detect spammers and promoters on YouTube. Our approach exploits the attributes discussed in the previous section, representing each user by a vector of attribute values. The core of our approach is a supervised classification algorithm, which learns a classification model from a set of previously labeled (i.e., pre-classified) training data, and applies the acquired model to classify new (unseen) users into three classes: spammers, promoters, and legitimate users. We present two supervised algorithms as basis for our detection method, namely: (1) a Support Vector Machine (SVM) classifier [31], which is considered a state-of-the-art method in supervised classification for a number of tasks (Section V-A), and (2) the Lazy Associative Classifier (LAC) [44], which has produced superior performance than SVM in some supervised tasks such as text classification [43], tag recommendation [37], and ranking [42] (Section V-B). Finally, in Section V-C, we propose an algorithm to select users for labeling as an attempt to reduce the amout of training data.

A. SVM

The goal of SVM is to find the hyperplane that optimally separates with a maximum margin the training data belonging to two different classes into two portions of an N-dimensional space. An SVM performs classification by mapping input vectors into an N-dimensional space, and checking in which side of the defined hyperplane the point lies. SVMs are originally designed for binary classification but can be extended to multiple classes using several strategies (e.g., one against all [29]). When the (training) data is not completely linearly separable, one may parameterize SVM to assign a cost to possible misclassifications. By tuning this cost, one may exploit the tradeoff between allowing training errors and forcing rigid margins, or, in other words, allowing *soft margins*. There is also the possibility of defining more complex boundaries for separations using *kernel functions* (e.g., polynomial or radial basis functions - RBF) which map the data points into a different space in which the data become more separable. The choices of the kernel and cost value, two parameters of the classifier, that maximize classification effectiveness are data-dependent.

B. LAC

LAC exploits the fact that, frequently, there are strong associations between attribute-values and classes. Such associations are usually hidden in the training data, and when uncovered, they may reveal important aspects that can be used for the sake of predicting classes.

LAC produces a classification function composed of rules $\mathcal{X} \rightarrow c_i$, indicating the association between a set of attribute-values $\mathcal{X}$ and a class c_i (i.e., spammer, promoter, or legitimate). In the following, we denote as $\mathcal{R}$ an arbitrary rule set.

Similarly, we denote as $\mathcal{R}_{c_i}$ a subset of $\mathcal{R}$ that is composed of rules of the form $\mathcal{X} \rightarrow c_i$ (i.e., rules predicting class c_i). A rule $\mathcal{X} \rightarrow c_i$ is said to match user x if $\mathcal{X} \subseteq x$ (i.e., user x contains all attribute-values in $\mathcal{X}$) and this rule is included in $\mathcal{R}^x_{c_i}$. That is, $\mathcal{R}^x_{c_i}$ is composed of rules predicting class c_i and matching user x. Obviously, $\mathcal{R}^x_{c_i} \subseteq \mathcal{R}_{c_i} \subseteq \mathcal{R}$.

LAC learns the classification function in two broad steps, discussed next.

1) Demand-Driven Rule Extraction: The search space for rules is huge, and thus, computational cost restrictions must be imposed during rule extraction. Let $\mathcal{D}$ and $\mathcal{T}$ be the sets of labeled training data (i.e., pre-classified users) and unlabeled testing data (i.e., to-be-classified users), respectively. Typically, a minimum support threshold (σ_{min}) is employed in order to select frequent rules (i.e., rules occurring at least σ_{min} times in $\mathcal{D}$) from which the classification function is produced. This strategy, although simple, has some problems. If σ_{min} is set too low, a large number of rules will be extracted from $\mathcal{D}$, and often most of these rules are useless for predicting the class of users in $\mathcal{T}$ (a rule $\{\mathcal{X} \rightarrow r_i\}$ is only useful to predict the class of user $d \in \mathcal{T}$ if the set of features $\mathcal{X} \subseteq d$, otherwise the rule is meaningless to d). On the other hand, if σ_{min} is set too high, some important rules will not be included in $\mathcal{R}$, causing problems if some users in $\mathcal{T}$ contain rare features (i.e., features occurring less than σ_{min} times in $\mathcal{D}$). Usually, there is no optimal value for σ_{min}, that is, there is no single value that ensures that only useful rules are included in $\mathcal{R}$, while at the same time important rules are not missed. The method to be proposed next deals with this problem by extracting rules on a demand-driven basis.

Demand-driven rule extraction is delayed until a set of users in $\mathcal{T}$ is given for classification. Then, each individual user d in $\mathcal{T}$ is used as a filter to remove irrelevant features and examples from $\mathcal{D}$. This process produces a projected training data, $\mathcal{D}_d$, which is obtained after removing all feature-values not present in d. Then, a specific rule-set, $\mathcal{R}_d$ extracted from $\mathcal{D}_d$, is produced for each user d in $\mathcal{T}$.

Lemma 1: All rules extracted from $\mathcal{D}_d$ (i.e., $\mathcal{R}_d$) are useful to estimate r^d.

Proof: Since all examples in $\mathcal{D}_d$ contain only feature-values that are present in d, the existence of a rule $\{\mathcal{X} \rightarrow r_i\} \in \mathcal{R}_d$, such that $\mathcal{X} \nsubseteq d$, is impossible. ■

Theorem 1: The number of rules extracted from $\mathcal{D}_d$ increases polynomially with the number of distinct feature-values in $\mathcal{D}$, no matter the value of σ_{min}.

Proof: Let n be the number of distinct feature-values in $\mathcal{D}$. Obviously, the number of all rules is exponential in n (i.e., $O(2^n)$ rules). However, since an arbitrary user $d \in \mathcal{T}$ contains at most l feature-values (with $l \ll n$), then any rule matching d (i.e., an useful rule) can have at most l feature-values in its antecedent. That is, for any rule $\{\mathcal{X} \rightarrow r_i\}$, such that $\mathcal{X} \subseteq d$, $|\mathcal{X}| \leq l$. Consequently, for $\sigma_{min} \approx 0$, the number of possible rules matching d is $k \times (l + \binom{l}{2} + \ldots + \binom{l}{l}) = O(2^l) \ll O(n^l)$, where k is the number of distinct classes. Thus, the number of useful rules increases polynomialy in n. Since, according to Lemma 1, only useful rules are extracted from $\mathcal{D}_d$, then the number of rules extracted for all users in $\mathcal{T}$ is $O(|\mathcal{T}| \times n^l)$. ■

2) Prediction: : Naturally, there is a total ordering among rules as some rules show stronger associations than others. A widely used statistic, called confidence [4] (denoted as $\theta(\mathcal{X} \rightarrow c_i)$), measures the strength of the association between $\mathcal{X}$ and c_i. Put simple, the confidence of the rule $\mathcal{X} \rightarrow c_i$ is given by the conditional probability of c_i being the class of user x, given that $\mathcal{X} \subseteq x$.

Using a single rule to predict the correct class may be prone to error. Instead, the probability (or likelihood) of c_i being the class of user x is estimated by combining rules in $\mathcal{R}^x_{c_i}$. More specifically, $\mathcal{R}^x_{c_i}$ is interpreted as a poll, in which each rule $\mathcal{X} \rightarrow c_i \in \mathcal{R}^x_{c_i}$ is a vote given by features in $\mathcal{X}$ for class c_i. The weight of a vote $\mathcal{X} \rightarrow c_i$ depends on the strength of the association between $\mathcal{X}$ and c_i, which is given by $\theta(\mathcal{X} \rightarrow c_i)$. The process of estimating the probability of c_i being the class of user x starts by summing weighted votes for c_i and then averaging the obtained value by the total number of votes for c_i, as expressed by the score function $s(c_i, x)$ shown in Equation 1 (where $r_j \subseteq \mathcal{R}^x_{c_i}$, and $|\mathcal{R}^x_{c_i}|$ is the number of rules in $\mathcal{R}^x_{c_i}$). Thus, $s(c_i, x)$ gives the average confidence of the rules in $\mathcal{R}^x_{c_i}$. Obviously, the higher the confidence, the stronger the evidence of class membership.

$$s(c_i, x) = \frac{\sum_{j=1}^{|\mathcal{R}^x_{c_i}|} \theta(r_j)}{|\mathcal{R}^x_{c_i}|} \tag{1}$$

The estimated probability of c_i being the class of user x, denoted as $\hat{p}(c_i|x)$, is simply obtained by normalizing $s(c_i, x)$, as shown in Equation 2. A higher value of $\hat{p}(c_i|x)$ indicates a higher likelihood of c_i being the class for user x. The class associated with the highest likelihood is finally predicted as the class for user x.

$$\hat{p}(c_i|x) = \frac{s(c_i, x)}{\sum_{j=1}^{n} s(c_j, x)} \tag{2}$$

Two key parameters of LAC are the maximum size of the rules (i.e., number of attribute-values in $\mathcal{X}$) and the minimum confidence (θ). The former, in particular, should be carefully chosen as it may greatly impact classification effectiveness and training time.

C. Active LAC

In this section we present a novel algorithm referred to as ALAC (Active Lazy Associative Classifier), which relies on an effective selective sampling strategy in order to deal with the high cost of labeling large amounts of examples. LAC was extended to allow itself to select the subset of examples to be labeled, thus performing active classification. It does this sequentially, using the requested labeled examples to inform its decision of which example to select next. The hope is that by only requesting the labels of most informative examples, ALAC can learn to detect spammers and promoters using significantly fewer labeled examples than would be required if the examples were randomly sampled. Next, we describe the sampling function used by ALAC as well its stop condition.

1) Sampling Function: Consider a initial, potentially large, set of unlabeled users $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$. The problem we investigate in this section is how to select a small subset of users in $\mathcal{U}$, such that the selected users carry almost the same information of all users in $\mathcal{U}$. These highly informative users will compose the training data $\mathcal{D}$, and, ideally, $|\mathcal{D}| \ll |\mathcal{U}|$. This approach exploits the redundancy in feature-space that exists between different users in $\mathcal{U}$. That is, many users in $\mathcal{U}$ may share some of their feature-values, and our approach uses this fact to perform an effective selective sampling strategy.

Intuitively, if a user $u_i \in \mathcal{U}$ is inserted into $\mathcal{D}$, then the number of useful rules for users in $\mathcal{U}$ that share feature-values with u_i will possibly increase. In constrast, the number of useful rules for those users in $\mathcal{U}$ that do not share any feature-value with u_i will clearly remain unchanged. Therefore, the number of rules extracted for each user in $\mathcal{U}$ can be used as an approximation of the amount of redundant information between users already in $\mathcal{D}$ and users in $\mathcal{U}$. The sampling function employed exploits this key idea, by selecting users that contribute primarily with non-redundant information, and these informative users are those likely to demand the fewer number of rules from $\mathcal{D}$. More specifically, the sampling function $\gamma(\mathcal{U})$ returns a user in $\mathcal{U}$ according to Equation 3:

$$\gamma(\mathcal{U}) = \{u_i \text{ such that } \forall u_j : \ |\mathcal{R}_{u_i}| \ < \ |\mathcal{R}_{u_j}|\} \qquad (3)$$

The users returned by the sampling function are inserted into $\mathcal{D}$, but it also remains in $\mathcal{U}$. In the next round, the sampling function is executed again, but the number of rules extracted from $\mathcal{D}$ for each user in $\mathcal{U}$ is likely to change due to the user recently inserted into $\mathcal{D}$. The intuition for choosing the user which demands the fewest rules is that such user should share fewer feature-values with users that were already inserted into $\mathcal{D}$. That is, if only few rules are extracted for a user u_i, then this is an evidence that $\mathcal{D}$ does not contain users that are similar to u_i. Thus, the information provided by user u_i is not redundant, and u_i is a highly informative user. This simple heuristic works in a fine-grained level of feature-values trying to maximize the diversity in the training set.

Notice that initially $\mathcal{D}$ is empty, and thus our algorithm cannot extract any rule from it. The first user to be labeled and inserted into $\mathcal{D}$ is selected from the set of available users $\mathcal{U}$. In order to maximize the initial coverage of $\mathcal{D}$, the selected user is the one that maximizes the size of the projected data in $\mathcal{U}$, that is, it is the user d for which $\mathcal{U}_d$ is the largest. This is the user that shares more feature-values with the other users of the collection and can be considered as the best representative of it. After the first user is selected and labeled, the algorithm proceeds using the fewest rules heuristic, as described above.

2) Natural Stop Condition: After selecting the first user and at each of the posterior rounds, ALAC executes the sampling function and a new example is inserted into $\mathcal{D}$. At iteration i, the selected user is denoted as $\gamma_i(\mathcal{U})$, and it is likely to be as dissimilar as possible from the users already in $\mathcal{D}$=$\{\gamma_{i-1}(\mathcal{U}), \gamma_{i-2}(\mathcal{U}), \ldots, \gamma_1(\mathcal{U})\}$. The algorithm keeps inserting users into the training data, until the stop criterion is achieved.

Lemma 2: If $\gamma_i(\mathcal{U}) \in \mathcal{D}$ then $\gamma_i(\mathcal{U})$=$\gamma_j(\mathcal{U})$ $\forall j$>i.

Proof: If $\gamma_i(\mathcal{U}) \in \mathcal{D}$ then the inclusion of $\gamma_i(\mathcal{U})$ does not change $\mathcal{D}$. As a result, any further execution of the sampling function must return the same user returned by $\gamma_i(\mathcal{U})$, and $\mathcal{D}$ will never change. ■

The algorithm stops when all available users in $\mathcal{U}$ are less informative than any user already inserted into $\mathcal{D}$. This occurs exactly when ALAC selects a user which is already in $\mathcal{D}$. According to Lemma 2, when this condition is reached, ALAC will keep selecting the same user over and over again, and there is no information gain with the inclusion of these users. At this point, the training data $\mathcal{D}$ contains the most informative users. All steps of the sampling process used by ALAC are shown in Algorithm 2.

Algorithm 2 Selective Sampling using Association Rules

Require: Unlabeled data $\mathcal{U}$, and σ_{min} (≈ 0)
Ensure: The training data $\mathcal{D}$
continue
  for all user $u_i \in \mathcal{U}$ **do**
    $\mathcal{D}_{u_i} \Leftarrow \mathcal{D}$ projected according to u_i
    $\mathcal{R}_{u_i} \Leftarrow$ rules extracted from $\mathcal{D}_{u_i} \mid \sigma \geq \sigma_{min}$
  end for
  if $\mathcal{D}$ **is** $\emptyset$ **then**
    $\gamma_i(\mathcal{U}) \Leftarrow u_i$ such that $\forall u_j : |\mathcal{D}_{u_i}| > |\mathcal{D}_{u_j}|\}$
  else
    $\gamma_i(\mathcal{U}) \Leftarrow u_i$ such that $\forall u_j : |\mathcal{R}_{u_i}| < |\mathcal{R}_{u_j}|\}$
  end if
  if $\gamma_i(\mathcal{U}) \in \mathcal{D}$ **then break**
  else append $\gamma_i(\mathcal{U})$ **to** $\mathcal{D}$

VI. Evaluation

In this section, we evaluate the viability of applying machine learning techniques to automatically detect spammers and promoters on YouTube. We start by introducing the metrics used to evaluate our supervised approaches (Section VI-A) as well as our experimental setup (Section VI-B). Then, in the following three sections, we present the most relevant results of the considered supervised classifiers (Section VI-C), discussing the performance tradeoffs of favoring the correct classification of one class over the others (Section VI-D) as well as of reducing the attribute set (Section VI-E). Finally, Section VI-F presents the results of our active learning approach.

A. Evaluation Metrics

To assess the effectiveness of our classification strategies we use the standard information retrieval metrics of recall, precision, Micro-F1, and Macro-F1 [48]. The recall (r) of a class X is the ratio of the number of users correctly classified as X to the number of users in class X. Precision (p) of a class X is the ratio of the number of users classified correctly as X to the total predicted as users of class X.

The F1 metric is the harmonic mean between both precision and recall, and is defined as $F1 = 2pr/(p+r)$. Two variations of F1, namely, micro and macro, are normally reported to evaluate classification effectiveness. Micro-F1 is calculated by first computing global precision and recall values for all classes, and then calculating F1. Micro-F1 considers equally

important the classification of *each user*, independently of its class, and basically measures the capability of the classifier to predict the correct class on a per-user basis. In contrast, Macro-F1 values are computed by first calculating F1 values for each class in isolation, and then averaging over all classes. Macro-F1 considers equally important the effectiveness in *each class*, independently of the relative size of the class. Thus, the two metrics provide complementary assessments of the classification effectiveness. Macro-F1 is especially important when the class distribution is very skewed, as in our case, to verify the capability of the classifier to perform well in the smaller as well as in the larger classes.

B. Experimental Setup

Unless otherwise noted, the classification experiments discussed in this section are performed using a 5-fold cross-validation. In each test, the original sample is partitioned into 5 sub-samples, out of which 4 are used as training data, and the remaining one is used for testing the classifier. The process is then repeated 5 times, with each of the 5 sub-samples used exactly once as test data, thus producing 5 results. The entire 5-fold cross validation is repeated 5 times with different seeds used to shuffle the original dataset, thus producing 25 different results for each test. The results reported are averages of the 25 runs. We also report error intervals with 95% of confidence level [30].

We used a non-linear SVM with RBF kernel. The implementation of SVM used in our experiments is provided with libSVM [17], an open source SVM package that allows searching for the best classifier parameters, namely type of kernel and cost, using the *training data*, a mandatory step in the classifier setup. In particular, we used the *easy* tool from libSVM, which provides a series of optimizations, including normalization of all numerical attributes. In case of LAC, best parameters were also obtained using cross-validation in the training set, being the maximum size of the rules set to five (i.e., at most four attribute-values in the antecedent of the rule) and the minimum confidence set to 0.01.

In the next two sections, we discuss the results obtained with the two classifiers using all 60 attributes, since, as discussed in Section IV, even attributes with low ranks according to the employed feature selection methods (e.g., UserRank) may have some discriminatory power. Moreover, both classifiers are known for dealing well with high dimensional spaces. For instance, SVM is able to properly choose the weights for each attribute, giving low weigths to attributes that are not helpful for classification. LAC, on the other hand, through its projection, is able to focus only on a small subset of features and values that actually occur in the given test user to be classified. In Section VI-E, we discuss the impact of using different subsets of the attributes on classification effectiveness.

C. Effectiveness of Supervised Classification

We start evaluating our approach by comparing the effectiveness of the two supervised classifiers, namely, SVM and LAC. The results are shown in the first two rows of Table II, which provides averages and 95% confidence intervals for Micro- and Macro-F1 measures. The results shown in the other two rows of the table are discussed below and in Section VI-F. We notice that LAC obtained statistically better results than SVM for both metrics. The average Micro-F1 value for LAC is 0.905, meaning that it is predicting the correct class in about 90% of the cases. These results are, on average, 3% better than those obtained by SVM.

Classifier	Micro-F1	Macro-F1
Supervised SVM	0.875 ± 0.012	0.822 ± 0.019
Supervised LAC	0.905 ± 0.008	0.862 ± 0.017
Trivial Baseline	0.773 ± 0.031	0.290 ± 0.024
Active LAC	0.871 ± 0.010	0.847 ± 0.018

TABLE II
SUMMARY OF CLASSIFICATION RESULT S(AVERAGES AND 95% CONFIDENCE INTERVALS)

		Predicted		
		Promoter	**Spammer**	**Legitimate**
True	**Promoter**	**100%**	0%	0%
	Spammer	1.02%	**53.25%**	45.73%
	Legitimate	0%	0.78%	**99.22%**

TABLE III
CLASSIFICATION RESULTS WITH LAC

More importantly, a large value of Macro-F1 for LAC (0.862, on average), with gains of approximately 5% over SVM, indicates that good results are obtained for all three classes. In the case of our test collection, which is skewed towards the legitimate user class, this large Macro-F1 value indicates that we are correctly classifying users from all classes. In comparison with a trivial baseline classifier that chooses to classify every user as legitimate (results shown in the third row of Table II), LAC obtains gains of about 17% in terms of average Micro-F1, and 197% in terms of average Macro-F1.

In order to further analyze LAC's performance, we show, in Table III, the percentage of users from each class that are classified as promoters, spammers and legitimate users, on average. The diagonal (in boldface) indicates the recall for each class: 100% of promoters, 53% of spammers, and 99% of legitimate users were correctly classified, on average. Note that a significant fraction (almost 46%) of spammers are misclassified as legitimate users. In general, these spammers exhibit a dual behavior, sharing a reasonable number of legitimate videos (non-spam) and posting legitimate video responses, thus presenting themselves as legitimate users most of the time, but occasionally posting video spams. This dual behavior masks some important aspects used by the classifier to differentiate spammers from legitimate users. This is further aggravated by the fact that some legitimate users do post their video responses to popular responded videos (see Figure 1(a)), a typical behavior of spammers. Therefore, although our classification results show that we can effectively identify promoters, distinguishing spammers from legitimate users is a

harder task. Nevertheless, being able to identify, on average, half of the spammers with no significant misclassification of legitimate users provides a good starting point for developing automatic tools to help system administrators dealing with content polluters.

Given its good absolute performance and comparable results against SVM, we focus, through the rest of this paper, on LAC as the classifier. We note that LAC is highly scalable, with polynomial time complexity [37], and is able to measure how informative the generated rules are for a given training instance, a capability that will be exploited in Section VI-F to reduce its dependency on large amounts of training data.

D. Impact of Favoring Classes Differently

The classification results discussed in the previous section are obtained assuming that the correct identification of a user is equally important for users from all three classes. However, there might be scenarios in which a system administrator could prefer to correctly identify more users from one class at the possible expense of misclassifying more users from the other classes. For instance, a system administrator, who is interested in sending automatic warning messages to all users classified as spammers, might prefer to act conservatively, avoiding sending messages to legitimate users by mistake, even if this comes at the cost of reducing the number of correctly identified spammers and/or promoters. In contrast, another system administrator, who adopts the policy of manually inspecting each user flagged as polluter before sending a warning, might prefer to favor the correct detection of spammers. In that case, misclassifying a few more legitimate users has no great consequences, and may be acceptable, since these users will be cleared out during manual inspection.

We now explore the classification tradeoffs of favoring the correct identification of users from one class over the others. We do so by using a weight factor $\pi_c > 0$ to represent the *importance* of correctly identifying users from class c. We apply this factor to a given instance x and class c_i by computing the score $\pi_c \times s(c_i, x)$, where $s(c_i, x)$ is defined in Equation 1. Figure 2 shows results as we vary the importance of correctly identifying each class, fixing the importance of the other classes at 1. It shows the average fraction of users from each class classified as the favored class (Figures 2(a-c)) and average Micro- and Macro-F1 values (Figures 2(d-f)). Confidence intervals are omitted for the sake of clarity, although we note that, with 95% of confidence, results do not differ from the reported averages by more than 5%.

Figure 2(b) shows that increasing the relative importance of detecting spammers does lead to a higher fraction of identified spammers, but at the cost of larger fractions of legitimate users and promoters misclassified as spammers. In particular, by setting $\pi_{spammer}$=2, one is able to detect about 85% of the spammers, misclassifying 8.4% of legitimate users and 9% of promoters. Depending on the policy adopted by the system administrator, the misclassification of such a small fraction of legitimate users might not be of much concern. The misclassification of promoters as spammers may be even less consequential. Alternatively, one can choose to set $\pi_{spammer}$=1.4, which, according to Figure 2(e), leads to the best Micro- and Macro-F1 results.

Figures 2(a) and (c) show that practically 100% of legitimate users and promoters, respectively, are correctly detected when the importance of all classes are kept the same (i.e., 1). These results are in agreement with those reported in Table III. Note however that, as shown in Figures 2(a) and Figure 2(d), by reducing the relative importante of legitimate users, one can significantly decrease the fraction of undetected spammers while reducing only slightly the fraction of misclassified legitimate users, leading to a better Macro-F1 result. For instance, for $\pi_{legitimate}$=0.7, the fraction of undetected spammers drops to 26%, whereas the fraction of correctly identified legitimate users remains very large (95%), leading to an average Macro-F1 of 0.892.

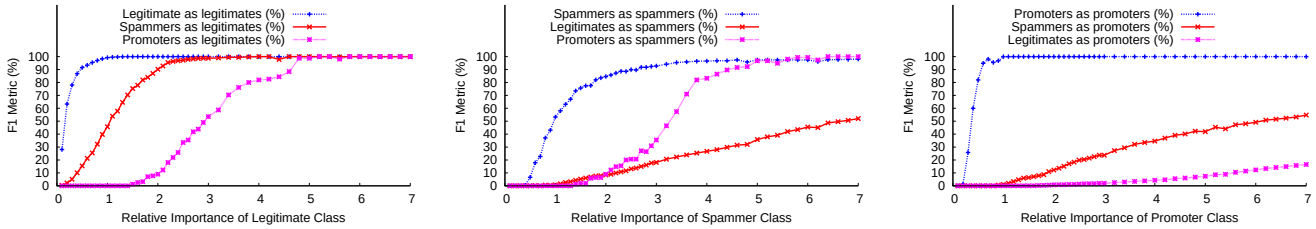
In the rest of this paper, we assign equal importance to the three classes (i.e., $\pi_{legitimate} = \pi_{spammer} = \pi_{promoter} = 1$).

E. Impact of Reducing the Attribute Set

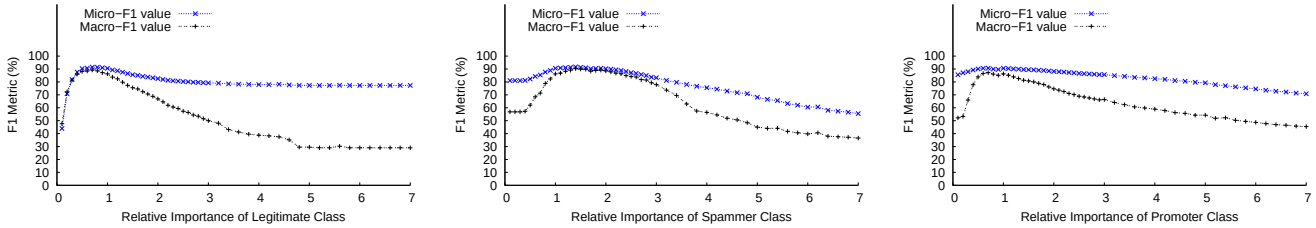
The detection of spammers and promoters is a form of adversarial fight between spammers and anti-spammers mechanisms. In the long term, it is expect that spammers will evolve and adapt to anti-spammers strategies (e.g. using fake accounts to forge some attributes). Consequently, some attributes may become less important whereas others may acquire importance with time. Thus, it is important to understand if different sets of attributes can lead our approach to effective detection. Next, we investigate the impact of using different sets of attributes on classification effectiveness. We leave as future work to investigate the best combination of features that can improve classification results [26].

Our analyses consider two scenarios. Scenario 1 consists of evaluating the impact of gradually removing attributes in a decreasing order of position in the χ^2 ranking. Figure 3(a) shows average Micro-F1 and Macro-F1 values, with corresponding 95% confidence intervals. As we reduce the number of attributes, there are initial losses in terms of both Micro- and Macro-F1, most probably due to the loss of information. With less information (i.e., fewer attributes) to learn the model from, the classifier may loose some of its capabality to predict the correct class of unseen instances, i.e., to generalize. These losses in effectiveness are worse for Macro-F1 (at most 14%) where there is also a higher variability. However, for both metrics, there is some recovery as less noisy and more informative sets of attributes are used to learn the model, so that, in some points, effectiveness is comparable to the one for the initial configuration.

Scenario 2 consists of evaluating our classification when subsets of 10 attributes occupying contiguous positions in the ranking (i.e., the first top 10 attributes, the next 10 attributes, etc) are used. Figure 3(b) shows average Micro- and Macro-F1 values and corresponding 95% confidence intervals for LAC and for a baseline classifier that considers all users as legitimate. LAC provides significant gains over the baseline, in terms of both Micro-F1 and Macro-F1 (but particularly Macro-F1), for all attribute ranges, but the last one (the 10 worst attributes). This confirms the results of our attribute analysis

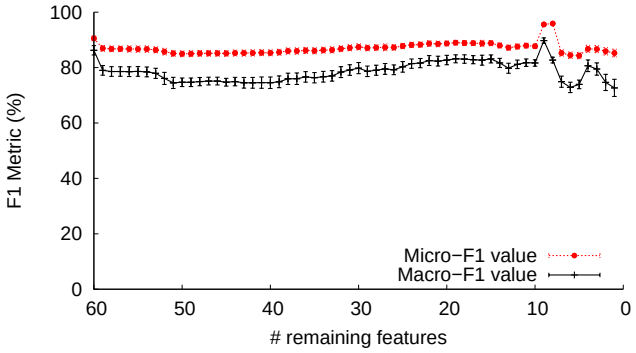


(a) Favoring Legitimate Users (b) Favoring Spammers (c) Favoring Promoters

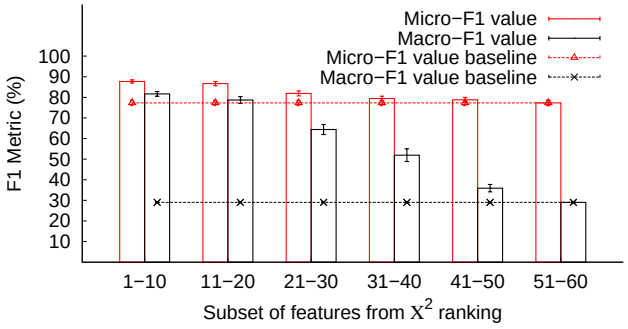


(d) Favoring Legitimate Users (e) Favoring Spammers (f) Favoring Promoters

Fig. 2. Impact of Varying the Relative Importance of Correctly Classifying each Class



(a) Scenario 1



(b) Scenario 2

Fig. 3. Impact of Reducing the Set of Attributes

(Section IV) that shows that even low-ranked attributes have some discriminatory power. In practical terms, significant improvements over the baseline are possible even if not all attributes considered in our experiments can be obtained.

F. Making Polluter Detection Practical

So far our goal was to show the viability of applying machine learning techniques, particularly supervised classification algorithms, to detect spammers and promoters on YouTube. However, the success of these techniques rely on the availability of large amounts of manually labeled examples, which, in turn, requires a highly costly effort from system administrators. Indeed, the cost associated with labeling may render vast amounts of examples impractical in many scenarios. Moreover, in the long run, it is expected that opportunistic users might change how they behave to adapt to anti-pollution strategies. In other words, it is expected that batches of labeled examples will be needed every so often to rebuild the classification model so as to more accurately capture current user behavior patterns[4]. Thus, strategies to reduce the required amount of labeled examples (training data) are of utmost importance to guarantee that our approach can be successfully applied in realistic and practical scenarios. Although the possible reduction in manual effort makes ALAC an attractive approach, the impact of this reduction on classification effectiveness is unknown. Thus, we here evaluate this strategy by investigating the tradeoffs between classification effectiveness and required amount of training data when we move from the supervised LAC to the active LAC.

It is complicated to compare active and supervised classification because, to be fair, both strategies need to be compared under similar conditions in order to avoid favoring one strategy over the other. Even the choice of the set of data from which examples will be selected for labeling may introduce some bias, favoring one technique. We propose to tackle this issue by performing two different and complementary sets of experiments, described next. Our goals with these experiments are twofold: 1) to show that, for a fixed amount of training data, ALAC can make better choices than simply randomly selecting the instances to be labeled; and 2) to show that ALAC can produce performance comparable to supervised LAC with a much smaller amount of training data. Through the rest of

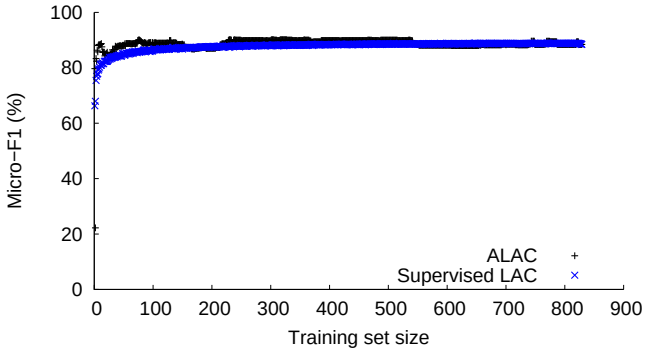
[4]An investigation of the frequency at which the classification model should be retrained is outside the present scope, and is left for future work.

this section we refer to supervised LAC as simply SLAC.

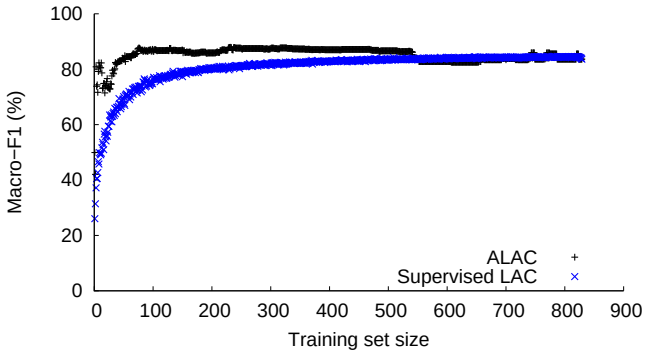
Our first evaluation scenario consists of partitioning the user collection into training and test sets (as in the previous sections) using 5-fold cross validation. In this scenario, SLAC uses the entire training set (4 folds) whereas ALAC uses only a fraction of them, containing the selected examples. The fourth row in Table II shows average Micro-F1 and Macro-F1 results for ALAC along with corresponding 95% confidence intervals. Note that, as expected, the results are slightly worse than those obtained with SLAC (second row in the same table), although the differences are quite small: average Micro-F1 for ALAC is only 3.75% worse than that for SLAC, whereas the relative difference in terms of average Macro-F1 is even smaller (1.7%). Note that this small loss in classification effectiveness comes with a great reduction in the amount of training data: ALAC was executed using, on average, only 18 samples for training, which corresponds to less than 3% the amount of training data used by SLAC. This experiment shows that selectively choosing the amount of training data with ALAC can be effective to detect spammers and promoters and largely reduce the amount of labeled data.

To further evaluate our approach we also considered a practical scenario illustrated in the following experiment. Instead of using 5-fold cross validation, we keep one single collection with all the data available for labeling (829 users). Examples to be manually labeled are chosen automatically by ALAC or, in the case of SLAC, randomly selected. The random selection simulates the generic and most common case of supervised learning in which training samples are chosen to be labeled with no specific criteria. It usually also guarantees that the distribution of classes in the collection is well represented in the sample that will be labeled. For either classifier, the manually labeled data is then used to create a model which is employed to classify the *whole dataset*, including the examples that had be manually labeled. Our goal with this experiment is to show how that ALAC is much more suitable in a practical scenario than SLAC. Note that, in this experiment, the set of unlabeled data from which to choose examples and the test sets are the same for both strategies, making the comparison fair.

For the experiments with SLAC, we proceed as follows: for each possible size k of the training set (starting with one example until the maximum size of 829, corresponding to the whole collection), we produce 100 distinct sets, each one with k examples randomly selected from the whole collection. We then run SLAC using each such set as training data, and report, for each value of k, the average results across 100 executions. In case of ALAC, we simply "turn off" the stop criterium, letting the method choose the most informative k instances, for each value of k. We then compare both strategies for all possible sizes k of training sets, always applying the learned model to the whole dataset. Due to the very large number of experiments, which would take a prohibitive amount of time to be executed, we here report results produced using only the "best" ten attributes according to the χ^2 ranking. We expect similar conclusions if all 60 attributes are used, particularly because, as shown in Figure 3(a), this reduced set of attributes produces a performance comparable to using all 60 attributes in the 5-fold cross-validation experiment, though running much faster.



(a) Micro-F1



(b) Macro-F1

Fig. 4. Classification Effectiveness of Active LAC and Supervised LAC as Function of Training Set Size

Figure 4 shows average Micro-F1 and Macro-F1 results achieved by ALAC and SLAC as a function of the training set size k. We omit confidence intervals of the SLAC results for the sake of clarity, but we note that results do not differ from the reported averages by more than 8%, with 95% confidence. In general, we see that the effectiveness of both methods, for both metrics, improve as k increases, with some instability in the beginning, until the methods converge to a reasonably stable performance[5]. Due to the much larger number of users in the legitimate class, in comparison with the number of spammers and promoters, Micro-F1 converges relatively quicker than Macro-F1 as both strategies start to correctly classify a large number of legitimate users. However, ALAC clearly converges much faster than SLAC. More importantly, with very few examples ALAC already provides excellent Micro- and Macro-F1 numbers, with a performance that SLAC achieves only with many more labeled examples.

For example, in terms of average Micro-F1, SLAC has its maximum effectiveness (0.889) with 771 examples, although a very similar average Micro-F1 (0.870) is also achieved with 130 samples. In comparison, ALAC achieves a similar performance, with an average Micro-F1 of 0.887, with only

[5]Classification results tend to stabilize as new examples inserted into the training set introduce noise instead of bringing new information to the classifier. Indeed, when most of the examples available are chosen for labeling, ALAC has a slight loss in effectiveness, becoming basically identical to SLAC.

12 examples, that is, 9% of the training set required by SLAC. If we look at the results after Micro-F1 stabilizes, ALAC needs only 36 examples (28% of the SLAC training set) to achieve a similar effectiveness, with average Micro-F1 equal to 0.878. Moreover, ALAC is capable to surpass the best results obtained by SLAC with a much smaller amount of training: it achieves an average Micro-F1 of 0.901 with only 76 (carefully selected) examples.

Similar conclusions hold for Macro-F1. SLAC produces maximum average Macro-F1 (0.846) also with 771 training examples. Although convergence is somewhat slower, it is still able to reach similar performance with a smaller training set (e.g., average Macro-F1 is 0.828 with 365 training examples). Further reducing the training set causes SLAC to have an average effectiveness that is far from the maximum, suggesting that the probability of randomly picking a good combination of examples is low. On the other hand, taking the average Macro-F1 achieved by SLAC with 365 examples as baseline, ALAC reaches the same performance with only 42 examples, a reduction in 88% in the required amount of training. Considering the performance after stabilization, ALAC with only 75 examples, 20% of the training set used by SLAC, achieves an average Macro-F1 of 0.874, which is superior to the maximum value obtained with the supervised classification.

It is worth noticing that even when almost all available examples are manually labeled, the generated models do not achieve 100% of accuracy when used to classify the own collection. This is due to noise and inconsistencies inherent to the data, and illustrates the difficulty of the task. It is also interesting to notice that, with about 80% of training, SLAC produces results, in terms of both Micro- and Macro-F1, that are very close to those obtained in the experiments of Section V, although a direct comparison with those results is not fair as the experimental designs are different.

To summarize, the fast convergence of ALAC leads to a great reduction of required labeling effort, in some cases with no significant loss of effectiveness, and, in others, with even superior results, making our algorithm very suitable for practical and realistic scenarios. In particular, if we "turn on" the criterium to stop selecting examples to be labeled, ALAC ends up requiring a training set with only 68 examples, and having a classification effectiveness, in terms of average Micro- and Macro-F1, of 0.881 and 0.845, respectively. In comparison with supervised LAC after stabilization (i.e., 130 samples for Micro-F1 and 365 for Macro-F1), this corresponds to a reduction of 48% and 81% in the required labeling effort for a similar performance in terms of average Micro- and Macro-F1, respectively.

VII. Conclusions and Future Work

Promoters and spammers can pollute video retrieval features of online video social networks, compromising not only user satisfaction with the system, but also the usage of system resources and the effectiveness of content delivery mechanisms such as caching and CDNs. We here proposed an effective solution that can help system administrators to detect spammers and promoters in online video social networks. Relying on a sample of pre-classified users and on a set of user behavior attributes, our supervised classification approaches are able to correctly detect the vast majority of the promoters and many spammers, misclassifying only a very small number of legitimate users. Thus, our proposed approach poses a promising alternative to simply considering all users as legitimate or to randomly selecting users for manual inspection. Moreover, given that the cost of the labeling process may be too high for practical purposes, we also propose an active learning approach, which was able to produce results very close to the completely supervised solutions, but with a greatly reduced amount of labeled data.

We envision some directions towards which our work can evolve. We intend to explore other refinements to the proposed approach such as to use different classification methods, perhaps combining multiple strategies. We believe that better classification effectiveness may require exploring other features which include temporal aspects of user behavior and also features obtained from other social networks established among YouTube users. Additionlly, we intend to explore a better combination of features to improve classification results. Finally, we also plan to extend our general approach to detect malicious and opportunistic users in other online social network sites and contexts.

Acknowledgements

This work is partially supported by the INCT-Web (MCT/CNPq grant 57.3871/2008-6), and by the authors' individual grants and scholarships from CNPq, FAPEMIG, and CAPES.

References

[1] comscore: Americans viewed 12 billion videos online in may 2008. http://www.comscore.com/press/release.asp?press=2324.

[2] New york times. uploading the avantgarde. http://www.nytimes.com/2009/09/06/magazine/06FOB-medium-t.htm. Accessed in July/2010.

[3] Youtube fact sheet. http://www.youtube.com/t/fact_sheet.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Int'l Conference on Management of Data*, pages 207–216, 1993.

[5] F. Benevenuto, C. Costa, M. Vasconcelos, V. Almeida, J. Almeida, and M. Mowbray. Impact of peer incentives on the dissemination of polluted content. In *ACM Symposium on Applied Computing (SAC)*, 2006.

[6] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on twitter. In *Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2010.

[7] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, and M. Gonalves. Detecting spammers and content promoters in online video social networks. In *Int'l ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2009.

[8] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, M. Gonalves, and K. Ross. Video pollution on the web. *First Monday*, 15(4), April 2010.

[9] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, and K. Ross. Video interactions in online video social networks. *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, 5(4):1–25, 2009.

[10] S. Boll. Multitube–where web 2.0 and multimedia could meet. *IEEE MultiMedia*, 14, 2007.

[11] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Int'l World Wide Web Conference (WWW)*, 1998.

[12] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the web topology. In *Int'l ACM SIGIR*, 2007.

[13] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Network*, 17(5):1357–1370, 2009.

[14] D. Chau, Pandit, S. Wang, and C. Faloutsos. Parallel crawling for online social networks. In *Int'l World Wide Web Conference (WWW)*, 2007.

[15] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru. Phi.sh/$ocial: The phishing landscape through short urls. In *Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2011.

[16] C. Costa, V. Soares, J. Almeida, and V. Almeida. Fighting pollution dissemination in peer-to-peer networks. In *ACM symposium on Applied computing (SAC)*, pages 1586–1590, 2007.

[17] R. Fan, P. Chen, and C. Lin. Working set selection using the second order information for training svm. *Journal of Machine Learning Research (JMLR)*, 6, 2005.

[18] U. Fayyad and K. Irani. Multi-interval discretization of continuousvalued attributes for classification learning. In *Int'l Joint Conference on Articial Intelligence*, volume 2, pages 1022–1027, 1993.

[19] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *Int'l Workshop on the Web and Databases (WebDB)*, 2004.

[20] F. Figueiredo, F. Benevenuto, and J. Almeida. The tube over time: Characterizing popularity growth of youtube videos. In *Int'l Conference on Web Search and Web Data Mining (WSDM)*, 2011.

[21] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao. Detecting and characterizing social spam campaigns. In *ACM SIGCOMM Conference on Internet Measurement*, 2010.

[22] S. Garriss, M. Kaminsky, M. Freedman, B. Karp, D. Mazières, and H. Yu. Re: Reliable email. In *USENIX Conference on Networked Systems Design & Implementation (NSDI)*, 2006.

[23] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: A view from the edge. In *Internet Measurement Conference*, 2007.

[24] L. Gomes, J. Almeida, V. Almeida, and W. Meira. Workload models of spam and legitimate e-mails. *Performance Evaluation*, 64, 2007.

[25] C. Grier, K. Thomas, V. Paxson, and M. Zhang. @spam: The underground on 140 characters or less. In *ACM conference on Computer and communications security (CCS)*, pages 27–37, 2010.

[26] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[27] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Int'l. Conference on Very Large Data Bases*, 2004.

[28] 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, 2007.

[29] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. In *IEEE Transactions on Neural Networks*, volume 13, 2002.

[30] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons, INC, 1991.

[31] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*, 1998.

[32] G. Koutrika, F. Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina. Combating spam in tagging systems. In *Int'l Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2007.

[33] A. Langville and C. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[34] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots + machine learning. In *Int'l ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2010.

[35] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in p2p file sharing systems. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1174–1185, 2005.

[36] Y. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. Tseng. Detecting splogs via temporal dynamics using self-similarity analysis. *ACM Transactions on the Web (TWeb)*, 2, 2008.

[37] G. Menezes, J. Almeida, F. Belém, M. Gonçalves, A. Lacerda, E. Moura, G. Pappa, A. Veloso, and N. Ziviani. Demand-driven tag recommendation. In *European Conference on Machine Learning and Knowledge Discovery in Databases (PKDD)*, 2010.

[38] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Internet Measurement Conference*, 2007.

[39] A. Mislove, A. Post, K. Gummadi, and P. Druschel. Ostra: Leverging trust to thwart unwanted communication. In *Symposium on Networked Systems Design and Implementation (NSDI'08)*, 2008.

[40] M. Newman and J. Park. Why social networks are different from other types of networks. *Phys. Rev. E*, 68, 2003.

[41] A. Thomason. Blog spam: A review. In *Conference on Email and Anti-Spam (CEAS)*, 2007.

[42] A. Veloso, H. Almeida, M. Gonçalves, and W. Meira. Learning to rank at query-time using association rules. In *Conference on Research and Development in Information Retrieval*, pages 267–274, 2008.

[43] A. Veloso, W. Meira, M. Cristo, M. Gonçalves, and M. Zaki. Multi-evidence, multi-criteria, lazy associative document classification. In *Conference on Information and Knlwledge Management*, pages 218–227, 2006.

[44] A. Veloso, W. Meira, and M. J. Zaki. Lazy associative classification. In *Int'l Conference on Data Mining (SDM)*, pages 645–654, 2006.

[45] G. Weiss and F. Provost. The effect of class distribution on classifier learning: An empirical study. Technical report, 2001.

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

[47] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. In *ACM SIGCOMM*, 2008.

[48] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrival*, 1, 1999.

[49] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *Int'l Conference on Machine Learning*, 1997.