A Multi-view Approach for Detecting Non-Cooperative Users in Online Video Sharing Systems

Hendrickson R. Langbehn, Saulo Ricci, Marcos A. Gonçalves,
Jussara M. Almeida, Gisele L. Pappa, Fabrício Benevenuto

Universidade Federal de Minas Gerais, Brazil
{reiter, saulomrr, mgoncalv, jussara, glpappa, fabricio}@dcc.ufmg.br

Abstract. Most online video sharing systems (OVSSs), such as YouTube and Yahoo! Video, have several mechanisms for supporting interactions among users. One such mechanism is the video response feature in YouTube, which allows a user to post a video in response to another video. While increasingly popular, the video response feature opens the opportunity for non-cooperative users to introduce "content pollution" into the system, thus causing loss of service effectiveness and credibility as well as waste of system resources. For instance, non-cooperative users, to whom we refer as *spammers*, may post unrelated videos in response to another video (the responded video), typically a very popular one, aiming at gaining visibility towards their own videos. In addition, users referred to as *content promoters* post several unrelated videos in response to a single responded one with the intent of increasing the visibility of the latter.

Previous work on detecting spammers and content promoters on YouTube has relied mostly on supervised classication methods. The drawback of applying supervised solutions to this specific problem is that, besides extremely costly (in some cases thousands of videos have to be watched and labeled), the learning process has to be continuously performed to cope with changes in the strategies adopted by non-cooperative users. In this work, we explore the use of multi-view semi-supervised strategies, which allows us to reduce significantly the amount of training, to detect non-cooperative users on YouTube. Our proposed method explores the fact that, in this problem, there is a natural partition of the feature space in sub-groups or "views", each being able to classify a given user when enough training data is available. Moreover, we propose to deal with the problem of view combination as a rank aggregation problem, where rankings based on confidence in the classification are combined to decide whether an unlabeled example should be included in the training set. Our results demonstrate that we are able to reduce the amount of training in about 80% without significant losses in classification effectiveness.

Categories and Subject Descriptors: H.3.5 [**Online Information Services**]: Web-based services

Keywords: multi-view classification, social networks, video pollution

1. INTRODUCTION

With the popularization of the Web in the last few years, the number of people that use the Internet is increasingly growing. A significant portion of these users watch online videos. According to [comScore 2010a], 84.48% of the Internet audience in the United States watched online videos in March 2010, being responsible for displaying more than 31 billion videos in the period. Because of this demand for online videos, online video sharing systems (OVSSs), such as YouTube and Yahoo! Video[1], are experiencing a vertiginous growth of popularity. Among these sites, YouTube is the one that most stands out, being responsible for providing 41.8% of the total amount of videos watched in the period mentioned above. The fact that users may create their own videos and post them on the Web,

[1]http://www.youtube.com, http://www.video.yahoo.com

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

passing from the role of viewers of the content to the role of content creators, greatly contribute to the popularity of OVSSs. According to [YouTube 2010], every minute, 24 hours of video is uploaded to YouTube. Moreover, according to [comScore 2010b], YouTube is the second site in number of queries received in March 2010. Typically, OVSSs have several mechanisms to facilitate the retrieval of videos. One such mechanism is a search engine, where the user types keywords in order to find videos related to an information need. Another mechanism consists of various ordered lists of top-videos, each one sorted according to a certain criterium, such as number of times a video was viewed or number of comments that a video received. Yet another mechanism consists of relationships established among users and/or videos. For example, each user can have a list of friends or a list of favorite videos. Thus, a user can retrieve the favorite videos of her friends very easily. Another example is the ability of a user to respond to a video by posting a related video in response to it, as in the *video response* feature provided by YouTube. A user is able to watch all the responses posted to some desired video, which supposedly would contain related content also of interest, without having to make a new search[2].

The three aforementioned factors (1) popularization of OVSSs, (2) the possibility for users to post their own videos, and (3) the mechanisms of video retrieval, make room for non-cooperative actions by the users themselves. Previous work has found evidence of non-cooperative users exploiting the video response feature in YouTube [Benevenuto et al. 2009]. Such users basically post completely unrelated videos in response to previously uploaded videos. One example of such actions is a video containing advertisement of adult content posted in response to a video of a very popular soccer game. According to [Benevenuto et al. 2009], those non-cooperative users can be classified into two types: spammers and content promoters. Spammers fit exactly in the previous example, since they are users who post unrelated videos in response to popular videos in order to increase the visibility of their own videos. A promoter is a user who tries to gain visibility towards her own video by posting a large number of videos, mostly unrelated, in response to it, aiming at boosting the number of video responses of the target video and making it enter more quickly in the top-list of most responded videos. Similarly, the same types of non-cooperative actions may occur in other features, such as the comments posted by users.

Content pollution brings several disadvantages to OVSSs, including: (1) loss of service effectiveness and credibility, as users, when navigating through the system, may be faced with an unacceptable amount of polluted content, (2) waste of space as the system has to store all the polluted content, (3) waste of bandwidth as users may watch at least a portion of a video to determine that it is pollution, and (4) loss of effectiveness of caches and content distribution networks that the OVSSs employ to replicate popular content (i.e., videos in the top-lists) so as to improve the service provided to users.

The first and, to the best of our knowledge, unique effort to address the problem of content pollution in OVSSs was done in [Benevenuto et al. 2009]. The authors propose classification-based mechanisms to identify users who are spammers and promoters, differentiating them from legitimate users of the system. Applying a supervised classification algorithm to a collection of 829 pre-classified users, the authors were able to detect the vast majority of the promoters as well as large fraction of the spammers.

Supervised methods need to “learn" a classification function through a set of training data. The drawback of applying such methods to detect non-cooperative users in YouTube is that the manual generation of the training base is very costly, as thousands of videos must be watched. For instance, in [Benevenuto et al. 2009], the authors mention that more than 20,000 videos were manually classified in order to build the collection of 829 users. Moreover, the learning process has to be continuously performed, usually with different training sets, to cope with changes in the strategies adopted by non-cooperative users. Alternatively, unsupervised methods require no training data at all, although the lower cost comes at the expense of a lower classification effectiveness. A better compromise between cost and classification effectiveness may be achieved with semi-supervised methods, which combine a smaller amount of labeled data with a large amount of unlabeled data to improve classification.

[2]YouTube also provides a related video list, associated with each video, which is created according to a proprietary algorithm.

In this work, we explore multi-view semi-supervised classification strategies, which allow us to reduce significantly the amount of training needed to detect spammers and content promoters in OVSSs. Our proposed methods explore the fact that, in this problem, there is a natural partition of the feature space in sub-groups or "views", each being able to classify a given user when enough training data is available. Thus, it is possible to combine the views to allow unlabeled data to be used to augment a much smaller set of labeled samples. We explore two strategies for combining the results from multiple views and selecting which unlabeled samples should be included in the training set. One strategy is based on the agreement of views regarding the label of an unlabeled sample, whereas the other is based on a rank aggregation strategy in which rankings based on the confidence in the classification are combined. We applied our methods to the same user collection used in [Benevenuto et al. 2009]. Our results demonstrate that we are able to reduce the amount of training by a factor of 5 without significant losses in classification effectiveness.

The rest of the paper is organized as follows. Next section discusses related work. Section 3 presents an overview of our multi-view method, describing the two view combination strategies and the classifier adopted. Section 4 describes our evaluation methodology, whereas the most representative results are discussed in Section 5. Finally, Section 6 offers conclusions and directions for future work.

2. RELATED WORK

Content pollution has been found in various applications and domains. Web spam is a type of pollution that is manifested through the creation of (typically fake) web pages. These spam web pages, typically useless for human visitors, are built so as alter or inflate the results of link analysis algorithms (e.g. PageRank [Brin and Page 1998]) used by search engines. The ultimate goal is to mislead search engines into erroneously lead users to certain sites. In [Fetterly et al. 2004], the authors propose to use statistical analysis of various web page properties, such as linkage structure and page content, to locate spam web pages. In [Castillo et al. 2007], the authors propose to identify hosts of spam pages through the use of the Web graph topology. They characterize the web pages according to attributes taken from the Web graph itself and to text attributes extracted from pages of each host. Strategies to semi-automatically separate good web pages from spam are proposed in [Gyöngyi et al. 2004]. The basic idea is to start with a number of manually classified good seeds, and then exploit the link structure of the web to discover other pages that are likely to be good as well.

In the e-mail domain, a characterization of traffic with the goal of identifying properties that distinguish legitimate e-mails from e-mail spams is presented in [Gomes et al. 2007]. The authors identify a number of characteristics, such as e-mail arrival process, e-mail size distribution and temporal locality of e-mail recipients, which can be used to separate legitimate traffic from spam, and conjecture that such differences are due to inherent differences in the way legitimate users and spammers behave: whereas the former are typically driven by bilateral relationships, spamming is typically a unilateral action, driven by the goal of reaching as many users as possible. In [Xie et al. 2008], the authors use features extracted from the content of e-mails, such as the time when the e-mail was sent, the sender IP address and URLs contained in the e-mail body, to identify URLs that lead to spam web pages as well as IP addresses of botnet hosts. Botnets are programs that are distributed across multiple computers and used to send a large number of spams in a short period of time.

In [Thomason 2007], the author addresses the presence of spam in blogs, claiming they are due to the combination of three factors, namely, the existence of several means to create a spam on a blog (e.g., blog post, comments, etc), the potential of reaching a large number of people with a single spam, and the limitation of anti-spam technology available at the time for blogs. The author also evaluates the effectiveness of two e-mail anti-spam tools in classifying blog comment spams. Another approach is taken by Lin *et al.* [Lin et al. 2008], who use the temporal dynamics of attributes extracted from the content of blog posts to identify spam blogs. They propose an SVM-based spam blog detector using the proposed features which reaches a 90% accuracy on real world data sets.

Many previously proposed strategies for identifying and combating content pollution on the Web are based on evidence extracted from textual descriptions of the content, treating the text as a set of objects with associated attributes, and then using some classification method to identify polluted content [Heymann et al. 2007]. A framework to detect spam in tagging systems is proposed in [Koutrika et al. 2007]. Some other strategies are based on image processing algorithms to detect spam in images. An example of this strategy is presented in [Wu et al. 2005] where image attributes are used in conjunction with attributes taken from the text of e-mails to improve e-mail spam detection.

The first evidence of the occurrence of non-cooperative behavior on the use of the video response feature on YouTube was raised in [Benevenuto et al. 2009]. In that article, the authors present a comprehensive characterization of the properties of the YouTube *video response network*, that is, the network that emerges from video-based user interactions. In [Benevenuto et al. 2009], the same authors further characterize the behavior of three classes of users, namely, legitimate users, spammers and content promoters. They exploit several attributes based on the users' profiles, the users' social behavior in the system (i.e., the relationships established among them) and the videos posted by the users as well as their target (responded) videos to classify users into one of the three classes. Adopting a supervised classification algorithm, they were able to detect the vast majority of promoters (over 95% of accuracy). While a significant amount of spammers were detected, the proposed method also missed a large fraction of them, which were incorrectly considered legitimate users. The false negatives may be due to spammers who exhibit a dual behavior, acting similarly to legitimate users some of the time.

The results presented in [Benevenuto et al. 2009] leave two venues for further exploration, namely, improving the detection of spammers and reducing the cost of building the training set. The former seems to require the use of content-based techniques to extract semantics and compare pairs of videos. This is outside the scope of this paper. Here, we are concerned with the second problem, that is, reducing the cost of building the training data without degrading classification effectiveness.

As previously mentioned, supervised classification algorithms require a training phase in which all examples must be previously manually labeled. This classification is very costly and requires the involvement of a large number of people to be held on time. In [Blum and Mitchell 1998], the authors present a multi-view approach for classification of web pages, where the labeling cost is reduced. However, the proposed approach assumes that there is total agreement in the classification performed from each view, which may not always be the case. A less constrained multi-view classification approach is proposed in [Christoudias et al. 2008]. The authors use a conditional entropy criterion to detect differences in the predictions of the classifiers. When divergence is identified in any sample, the sample is filtered, i.e., removed from the unlabeled data set and placed in a data set that will not be used. After this process, none of the samples in the unlabeled data set has divergences and, therefore, the multi-view classification approach can be used normally.

We here extend the work presented in [Benevenuto et al. 2009] by proposing multi-view classification approaches to reduce the labeling cost. As in [Christoudias et al. 2008], our approaches considers that there may be divergence in the classifications of each view. We here deal with divergence adopting one of two strategies, namely: (1) considering only elements for which all views agreed in the classification, i.e., disregarding elements whose classification diverged and (2) exploring the confidence of each view's prediction and taking, among elements whose classifications diverged, only those with the largest aggregated confidence for a given class using certain ranking aggregation strategies. These strategies are much simpler than the one adopted in [Christoudias et al. 2008] and, as will be shown in Section 5, lead to very good results.

3. MULTI-VIEW SEMI-SUPERVISED APPROACH TO DETECT NON-COOPERATIVE USERS

In this section, we present an overview of our multi-view semi-supervised approach to detect spammers and content promoters on OVSSs (Section 3.1). Then, we present our proposed methods to combine the results from different views, a step required by the semi-supervised algorithm (Section 3.2). Finally, we briefly present the classifier we used in each view (Section 3.3).

3.1 Overview of our Approach

The main idea behind our proposed multi-view semi-supervised classification is start with a small set of labeled data as the training set while still being able to expand it with examples extracted from a set of unlabeled data. The expanded training set is then used to classify the desired objects. During the insertion of new elements into the training set, the training set and the unlabeled set are partitioned into views. Each view consists of a number of attributes of the elements in the training set. In other words, each element in the training set is present in all views, represented by a different set of attributes in each of them. Take, for instance, the classification of YouTube users performed in [Benevenuto et al. 2009]. We can say that the authors explore three different views of each user, namely: the user profile, her social network, and the objects owned by or target by her. When no more elements can be inserted into the training set, all the views of the final training set are integrated into a single one, which is used as the final training data used in the classification. The idea is that, to work, the predictions of the classifiers trained with data from each view (hereafter also called the "views") should be compatible, i.e., all samples are labeled identically by all or most of the views, and the elements' representations in the views should be uncorrelated, i.e., each element is described by a different and disjoint set of attributes in each view.

This introduction of examples into the training set occurs through an iterative process, according to Algorithm 1. In each iteration of the algorithm, the classifier of each view v is trained with its corresponding training set L_v, and is used to predict the class of each element u_v^m in the unlabeled dataset U_v. Along with the predictions, the evaluation step also gives the confidence $\theta_v^m(k)$ of each element u_v^m being of class k. The confidence will be better explained in Section 3.3, where the classifier is presented. Based on the predictions of each classifier, a method is used to determine whether there are unlabeled elements that can be inserted into the training set and to give the final predicted class for each of these elements, which is done by the GetNewInsertions function. As an input, the GetNewInsertions function also uses the fraction of elements belonging to each class in the original training set, computed in lines 3-5 of the algorithm. This can guide some policies about how many elements from each class should be incorporated as training data, as we shall see in Section 3.2. Then, selected items have their class updated to the final predicted class and are incorporated into the training set and removed from the unlabeled data set. In other words, each selected example u_v^m has its class updated and is removed from the unlabeled data set of each view U_v and inserted into the training set of each view L_v. This process is repeated until either no more elements can be inserted into the training set or the unlabeled data set is empty. The methods used for the selection of elements to be included in the training set define the view combination strategy adopted. We discuss the strategies considered in this work in the next section.

For the specific problem of detecting non-cooperative users on OVSSs, our proposed multi-view semi-supervised approach can be applied to: (1) decrease the amount of training data needed by the classifier as well as (2) increase the quality of the classification process. Semi-supervised approaches can start with much smaller training set than the ones used in supervised approaches, since they can add more examples to the training set by applying the iterative process described in Algorithm 1. Indeed, semi-supervised approaches require only enough examples from each class to allow the classifiers to execute (reasonably) well in the first iteration of the algorithm, and select new examples to be introduced in the training set. How many examples is enough for the classifier depends on a variety of factors, like how discriminative the attributes of the examples given are.

Our approach can also help improving the classification effectiveness by letting us introduce a 4^{th} view - a video content view - composed of attributes extracted from the video itself. As discussed in Section 2, this might improve the detection of spammers. However, given that we do not have access to a test collection with video content based attributes, we leave this task for future work. In this paper, our focus is on applying the proposed approach to reduce as much as possible the amount of training data required by the classifier without loss in the quality of the classification.

Algorithm 1 Multi-view Semi-supervised Training Expansion

```
Input: The number of views $V$, a set $C$ of classifiers $C_v$ for each view $v$ ($v = 1...V$), the number of classes $K$, the number of
    elements $M$ in the unlabeled data set, a set $L$ of labeled samples $l_v^n$ with the attributes from each view $v$ ($v = 1...V$) for
    each element $l^n$ ($n = 1...N$), and a set $U$ of unlabeled samples $u_v^m$ with the attributes from each view $v$ ($v = 1...V$) for
    each element $u^m$ ($m = 1...M$). {$N$ is the number of elements in the labeled data set}
Output: Set $L$ expanded to include all initial elements as well as new elements added by the algorithm.
 1: repeat
 2:   $insertion \leftarrow FALSE$
 3:   for $k \leftarrow 1$ to $K$ do
 4:     $B_k \leftarrow$ the fraction of elements from class $k$ in $L$
 5:   end for
 6:   for $v \leftarrow 1$ to $V$ do
 7:     Train $C_v$ on $L_v$
 8:     Evaluate $C_v$ on $U_v$ giving predictions in $P_v$ and confidences in $\theta_v$ {$\theta$ is a set of the confidences $\theta_v^m(k)$ from each view
        $v$ of the pertinence of element $u^m$ to class $k$}
 9:   end for
10:   {$I$ receives elements that can be added to the training data along with the predicted class for each element}
11:   $I \leftarrow$ GetNewInsertions($V, K, B, M, UP, \theta$)
12:   if $|I| > 0$ then
13:     $insertion \leftarrow TRUE$
14:   end if
15:   for $i \leftarrow 1$ to $|I|$ do
16:     for $v \leftarrow 1$ to $V$ do
17:       $L_v \leftarrow L_v \cup u_v^i \in I$
18:       $U_v \leftarrow U_v \setminus u_v^i \in I$
19:     end for
20:   end for
21:   $M \leftarrow |U|$ {Update the number of elements in the unlabeled data set}
22: until ($|U| = 0$ or $insertion = FALSE$)
```

3.2 Combining Results from Multiple Views

An important step of the multi-view semi-supervised approach is how to select the elements to be inserted into the training set on each iteration, i.e., function *GetNewInsertions* in Algorithm 1. This function is responsible for combining the classification results of different views, that is, assigning a label to each element, and then selecting those, according to certain criteria, that should be added to the training set. Elements that are not selected remain as part of the unlabeled data set. Several strategies can be adopted to perform this task. We here explore two such strategies: one is based on view agreement and the other is based on a method to aggregate ranked lists called Borda Count [Black 1963].

Before introducing each strategy, we first address one of the criteria used to select elements to be inserted into the training set, which is applied by both strategies. During our initial experiments we found that it is very important, for the sake of classification effectiveness, to keep the distribution of the number of labeled elements per class roughly stable as new elements are inserted into the training set. In the specific problem of classifying non-cooperative users in OVSSs, user collections (and the one used here in particular) tend to be very skewed [Benevenuto et al. 2009] as most users tend to be legitimate. Thus, there is a natural bias towards the larger class. If we do not keep the class distribution roughly stable in the training set, this bias will tend to increase even more, compromising the classification effectiveness for the smaller classes, which is, in the specific case, what we care most. Therefore, we compute the initial distribution of elements across classes in the training set, specified as the fractions B of elements in each class, and use it to constrain the insertion of new elements in each iteration.

We now describe each considered strategy to combine the results from all views. The View Agreement strategy works as shown in Algorithm 2. For each element u^m in the unlabeled data set, u^m is selected provided that all views agree on the class predicted for it (say, class k). If selected, u^m is inserted into the set of candidate elements for the predicted class, S_k. Note that we can not insert u^m directly into the I set, or else we might change the distribution of elements across classes in the training set. After the candidate selection process finishes, we use the confidence of the predictions θ and the target class distribution B to determine which elements from the sets of candidate S will be inserted into I, what is done by the GetElementsByConfidence function. We do so by choosing the elements from each S_k with largest general confidence, constraining the number of selected elements

Algorithm 2 GetNewInsertions_ViewAgreement

```
Input: The number of views $V$, the number of classes $K$, a set $B$ with the initial fraction $B_k$ of elements from each class $k$,
    $k = 1...K$, the number of elements $M$ in the unlabeled data set, a set $U$ of unlabeled samples $u_v^m$ with the attributes from
    each view $v$, $v = 1...V$, for each element $u^m$, $m = 1...M$, a set $P$ of the class predictions $P_v^m$ from each view $v$, $v = 1...V$,
    for each element $u^m$, $m = 1...M$, and a set $\theta$ of the confidences $\theta_v^m(k)$ from each view $v$ of the pertinence of element $u^m$
    to class $k$.
Output: Set $I$ with the elements selected to be inserted along with the predicted class of each element.
 1: for $k \leftarrow 1$ to $K$ do
 2:     $S_k \leftarrow \{\}$ { $S_k$ will contain candidate elements of class $k$ to be considered for insertion}
 3: end for
 4: for $m \leftarrow 1$ to $M$ do
 5:     if $P_r^m = P_s^m, \forall r, s \in V, r \neq s$ then
 6:         {All views agree on class predicted for element $u^m$}
 7:         $k \leftarrow P_r^m$ {$k$ gets the class predicted by all views for element $u^m$}
 8:         $S_k \leftarrow S_k \cup u^m$
 9:     end if
10: end for
11: $I \leftarrow$ GetElementsByConfidence($B, \theta, S$) {Select elements from each $S_k$ with largest general confidence, constrained by class
    distribution $B$}
```

from each set so as to keep the distribution of elements across classes as close as possible to B. If any class does not have any agreement, or if the number of agreements is too small to keep the classes distribution unchanged, what is measured by the size of each S_k, the GetElementsByConfidence function returns an empty set to I, which usually implies in stop the whole process. We can calculate the general confidence in the prediction of each element u^m in several ways. We here use the sum of the confidences of each view for the predicted class. For example, if there are $V = 2$ views, and the class predicted (by both views) for element u^m is k, then the general confidence θ^m in the prediction of element u^m will be the confidence of the view 1 for element u^m being from class k plus the confidence of view 2 for element u^m being from class k, that is $\theta^m = \theta_1^m(k) + \theta_2^m(k)$. Alternatively, we could weight the confidence from each view by a factor reflecting the trust we have in it.

Algorithm 3 GetNewInsertions_BordaCount

```
Input: The number of views $V$, the number of classes $K$, a set $B$ with the initial fraction $B_k$ of elements from each class $k$,
    $k = 1...K$, the number of elements $M$ in the unlabeled data set, a set $U$ of unlabeled samples $u_v^m$ with the attributes from
    each view $v$, $v = 1...V$, for each element $u^m$, $m = 1...M$, a set $P$ of the class predictions $P_v^m$ from each view $v$, $v = 1...V$,
    for each element $u^m$, $m = 1...M$, and a set $\theta$ of the confidences $\theta_v^m(k)$ from each view $v$ of the pertinence of element $u^m$
    to class $k$.
Output: Set $I$ with the elements selected to be inserted along with the predicted class of each element in $I$.
 1: $I \leftarrow \{\}$
 2: for $k \leftarrow 1$ to $K$ do
 3:     $Q_k \leftarrow 0$ {$Q_k$ will contain the total aggregated ranking values $Q_k^m$ for each element $u^m$, produced by all views for class
        $k$. $Q_k$ values are initially set to zero.}
 4:     for $v \leftarrow 1$ to $V$ do
 5:         $sortedU_v(k) \leftarrow$ Sort $U$ by $\theta_v(k)$, in ascending order
 6:         for $m \leftarrow 1$ to $M$ do
 7:             $Q_k^m \leftarrow Q_k^m$+GetRankingValue($u_k^m$,$sortedU_v(k)$) {Aggregate ranking values for $u_k^m$ obtained with all views for
                class $k$}
 8:         end for
 9:     end for
10:     Sort $U$ by $Q_k$ values, in descending order {Sort $U$ according to final aggregated ranking values}
11:     $S_k \leftarrow$GetElementsByRank($B_k, U, P$) {Get the first elements of U sorted by the aggregated ranking values in $Q_k$ keeping
        the initial proportion of classes}
12:     $I \leftarrow I \cup S_k$
13:     $U \leftarrow U \setminus S_k$
14:     $M \leftarrow |U|$ {Update the number of elements in the unlabeled data set}
15: end for
```

The Borda Count strategy is based on a method previously proposed to combine list of candidates in elections and later used to solve computational problems such as the aggregation of rankings produced by multiple search engines [Dwork et al. 2001]. It was thus originally proposed in a very different context, and, to the best of our knowledge, this is the first work that explores its use in the context of multi-view classification. The advantage of this strategy is that is more resilient to issues related to the magnitude of the absolute values of the confidence in the prediction of elements in classes, which for some classes may be very low/high or may not help distinguish much between several classes (e.g., close confidence for all classes). These issues may affect any method which rely on the absolute values of the confidence (e.g., summation, average, max, min, etc). The strategy is presented in Algorithm 3. It calculates K ranking aggregation values for each element u^m in the unlabeled data set. Each such value corresponds to the aggregation of the rankings generated by each view v with respect to each class k ($k = 1...K$). The algorithm considers one class at a time and, for each element, it gets the confidence of the prediction of each view with respect to the class under consideration. Then, for each view, the algorithm sorts the elements, in increasing order, according to the confidence of the view for that class (line 5), so that the last element is the one with the highest confidence in that view for that class. The sorted elements get a ranking value according to the position occupied by them in the rank (function GetRankingValue), with elements in last positions getting higher values. These values are summed up for all views (line 7) to give a final value of the aggregation of the different ranks for a class. This final value is used to sort the elements again, in descending order (line 10). The top elements of this final rank, i.e., the ones with the higher rank aggregation values, which had its class predicted as k for at least one view, are selected to be inserted into the training data set in a way that the distribution of elements across classes B is not changed, what is done by the GetElementsByRank function (line 11). Like with the View Agreement strategy, a weighted sum of the ranking values could also be applied.

Notice that, for the sake of simplicity, both Algorithms 2 and 3 present the computation of set I for *all classes*. However, the set of candidate elements of each class k selected to be added to the training data, S_k, is determined independently for each class. Thus, different strategies can be applied to different subsets of the classes, depending on the characteristics of the problem being solved. In fact, we do explore such a hybrid approach in our experiments described in Section 5, as we shall see. In the following section, we present a brief description of the classifier used in our proposed multi-view method.

3.3 The Classifier

We use Lazy Associative Classification (LAC) [Veloso et al. 2006] as our classifier. 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 for elements.

LAC produces a classification function composed of rules $X \rightarrow k$, indicating the association between a set of attribute values X and a class k. In the following, we denote as R an arbitraty rule set. Similarly, we denote as R_k a subset of R that is composed of rules of the form $X \rightarrow k$, i.e., rules predicting class k. A rule $X \rightarrow k$ is said to match element u^m if $X \subseteq u^m$, (i.e., element u^m contains all attribute-values in X) and this rule is included in R_k^m. That is, R_k^m is composed of rules predicting class k and matching element u^m. As we can note, $R_k^m \subseteq R_k \subseteq R$.

LAC learns the classification function in two broad steps:

—**Demand-Driven Rule Extraction**: In order to avoid rule explosion, LAC extracts rules from the training data on a demand driven fashion [Veloso et al. 2006; Veloso et al. 2008], at learning time. It projects the search space for rules according to information about elements in the test set, allowing rule extraction with efficiency. In other words, LAC projects/filters the training data according to the attribute-values of element u^m in the test set, and extracts rules from this projected training data, which is denoted as D^m. This ensures that only rules that carry information about element u^m are extracted from the training data, drastically bounding the number of possible rules.

—**Prediction**: There is a total ordering among rules, in the sense that some rules show stronger associations than others. A widely used statistic, called confidence [Agrawal et al. 1993] (denoted as $\theta(X \rightarrow k)$), measures the strength of the association between X and k. The confidence of the rule $X \rightarrow k$ is given by the conditional probability of k being the class of element u^m, given that $X \subseteq u^m$. Using a single rule to predict the correct class may be prone to error. Instead, the probability of k being the class of element u^m is estimated by combining rules in R_k^m. More specifically, R_k^m is interpreted as a poll, in which each rule $X \rightarrow k \in R_k^m$ is a vote given by features in X for class k.

The weight of a vote $X \rightarrow k$ depends on the strength of the association between X and k, which is given by $\theta(X \rightarrow k)$. The process of estimating the probability of k being the class of element u^m starts by summing weighted votes for k and then averaging the obtained value by the total number of votes for k, as expressed by the score function $s(k, u^m)$, shown in Equation 1 (where $r_x \subseteq R_k^m$ and $|R_k^m|$ is the number of rules in R_k^m). Thus, $s(k, u^m)$ gives the average confidence of the rules in R_k^m. Obviously, the higher the confidence, the stronger the evidence of class membership.

$$s(k, u^m) = \frac{\sum_{x=1}^{|R_k^m|} \theta(r_x)}{|R_k^m|} \tag{1}$$

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

$$\hat{p}(k|u^m) = \frac{s(k, u^m)}{\sum_{l=1}^{K} s(l, u^m)} \tag{2}$$

We note that in Algorithms 1, 2 and 3, we denoted by $\theta_v^m(k)$ the confidence of view v on predicting class k for element u^m. Thus, $\theta_v^m(k)$ is indeed the value of $\hat{p}(k|u^m)$ for view v's classifier.

4. EVALUATION METHODOLOGY

In this section, we describe how the multi-view semi-supervised classification method, presented in the previous section, is applied to detect non-cooperative users, namely spammers and content promoters, in OVSSs. We start by describing, in Section 4.1, our user test collection, introducing the views considered by our approaches and their associated attributes. Next, the metrics used in the evaluation of our solutions are introduced in Section 4.2, whereas the experimental setup is presented in Section 4.3.

4.1 User Test Collection

In order to evaluate our proposed approaches to identify spammers and content promoters in OVSSs, we need a test collection composed of users of the target system, which in our case is YouTube. In this collection, all users must be pre-classified into legitimate users, spammers or promoters. The process of building such a collection is very expensive, as it requires human effort in watching a potentially very large number of videos. Thus, we here use the same user collection presented in [Benevenuto et al. 2009], which was built primarily through a crawling of YouTube followed by a selection of a subset of the crawled users to be manually classified.

The crawling phase consisted of collecting a sample of users involved in interactions through the use of YouTube video response, i.e., users who had posted or received video responses. This crawling, performed in January 2008, gathered a total of 264,460 users, 381,616 responded videos and 701,950 video responses. Users were gathered by starting with a number of seeds and following their interactions via video responses (i.e., snowball strategy), thus building, at the end, a video response user network.

In the manual classification phase, a selected user was labelled as a “spammer" if she posted at least one video response that was considered unrelated to the responded video. She was labeled as “promoter" if she posted several videos in response to a responded video with the aim of promoting this responded video. A user who is neither promoter nor spammer was labeled as “legitimate". The user test collection built from the manual classification has a total of 829 users, consisting of 641 legitimate users, 157 spammers and 31 promoters. These users have posted 20,644 video responses to 9,796 responded videos. This collection has the following characteristics: (1) it has a significant number of users from all three classes, (2) it includes, but is not limited to, non-cooperative users with aggressive strategies, as these are the users who generate the most pollution in the system and (3) includes legitimate users with different behaviors. In the following ,we present the user attributes gathered in our collection as well as the multiple views extracted from these attributes and adopted by our classification approaches.

4.1.1 *User Attributes.* Legitimate users, spammers and promoters have different goals in the system and are, thus, expected to act differently while using the system. Such differences may be captured by exploring a number of attributes that reflect how each user uses the system. In particular, our user test collection contains a total of 60 attributes per user, which can be divided into three groups: video attributes, user attributes and attributes of the social network established among the users through the use of the video response feature.

The video attributes associated with a user relate to features of the videos posted by her as well as the videos responded by her (i.e., the videos that were target of her video responses). The video features considered are: the duration, the number of views, the number of comments received, ratings, number of times that the video was selected as favorite, number of honors, and number of external links from the video. Note that these attributes serve as indicators of the quality of a video, as perceived by the user community. Three groups of these attributes were created. The first group contains aggregated information from all the videos posted by the user, which may indicate how others see the contributions of this user. The second group contains information only of the video responses posted by the user, which are precisely those videos that can be pollution. The latter group considers only responded videos to which the user posted video responses. For each of these groups were considered the sum and average of each attribute, totaling 42 video attributes.

The user attributes consist of individual features of user behavior, extracted from the user's profile on the system. Ten attributes are used: number of friends, number of videos uploaded, number of videos watched, number of videos added as favorites, number of video responses posted, number of video responses received, number of subscriptions, number of subscribers, average time between uploads and maximum number of videos uploaded within 24 hours.

The attributes of the user's social network capture the social relations established among users through video responses, which is one of the several social networks that emerge among users on YouTube. This network is modelled as a directed graph, where each node represents a user, and a edge (i, j) indicates that the corresponding user u_i posted at least one video in response to some video of user u_j. The 5 attributes of social network included in our user collection are: 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, i.e., nodes appearing in a larger number of shortest paths between any two nodes have higher betweenness than others[Newman and Park 2003]. 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, i.e., $R(i) = \frac{|OS(i) \cap IS(i)|}{|OS(i)|}$, 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 [Castillo et al. 2007], as the ratio between the node (in/out) degree and the average (in/out) degree of its neighbors. Node assortativity was computed to the four types of (in/out)degree-(in/out)degree correlations. The PageRank [Brin and Page 1998] algorithm, commonly used to assess the popularity of a Web page [Langville and Meyer 2006], was applied to the video response user graph built from the collection. The computed metric, called UserRank, indicates the degree of participation of a user in the system through interactions via video responses. In total, 8 social network attributes were used.

4.1.2 *Multi-View Collection.* Unlike in [Benevenuto et al. 2009], where authors applied a single-view supervised classification method, our goal here is to explore multi-view semi-supervised approaches. Thus, we need to extract different views from the user collection. As explained in the previous section, the collection already has three separate groups of attributes, namely user attributes, video attributes and social network attributes. Thus, we take this inherent categorization of the user attributes in the collection to generate a video view, a user view and a social network view. Each view includes only the attributes of the corresponding group. In other words, the video view consists of 42 attributes, the user view has 10 attributes, and the social network view has 8 attributes.

4.2 Evaluation Metrics

We evaluate the classification approaches by comparing mainly the confusion matrices [Kohavi and Provost 1998] produced by each of them. Each element in position (i, j) of this matrix represents the percentage of users from class i (i.e., row i) that were predicted, by the classification, as being of class j (i.e., column j). We choose to focus our evaluation on these matrices because they better expose the tradeoffs between correctly classifying users of one class at the expense of misclassifying users of the others. These tradeoffs are particularly interesting for the specific task of classifying YouTube users as either legitimate or non-cooperative. We envision our approaches being used to help system administrators by “flagging" suspicious users for further (possibly manual) investigation. In that case, we believe that it is preferable to improve the detection of non-cooperative users even if it comes at the expense of misclassifying some legitimate users as non-cooperative. These wrongly classified legitimate users will have the chance to be cleared out later. In contrast, non-cooperative users who are misclassified as legitimate may escape undetected as manual investigation of the large number of (predicted) legitimate users is highly unlikely. Thus, when comparing the classification approaches in Section 5, we focus mainly on the confusion matrices, thus allowing us to better assess the tradeoffs between correctly classifying legitimate and non-cooperative users.

In addition to the confusion matrices, we also consider the F1 metric [Yang 1999], commonly used to evaluate information retrieval tasks. F1 is defined as a function of precision and recall. The precision (p) of a class k is the ratio of the number of users correctly classified to the total number of users predicted to be of class k. The recall (r) of a class k is the ratio of the number of users correctly classified to the number of users in class k.The F1 metric is the harmonic mean between both precision and recall, defined as $F1 = 2pr/(p + r)$. There are two variations of F1, namely Micro-F1 and Macro-F1. Macro-F1 values are computed by first calculating F1 values for each class in isolation and then averaging over all classes. Therefore, Macro-F1 considers equally important the classification effectiveness in each class, independently of the relative size of the classes, being thus more adequate when the class distribution is very skewed. Since our user test collection is inherently very skewed towards legitimate users[3], we consider only Macro-F1 in our evaluation.

4.3 Experimental Setup

We ran a series of experiments with five classification approaches. Three approaches are based on the proposed multi-view semi-supervised method, and explore the two view combination strategies introduced in Section 3.2. To evaluate the effectiveness of these approaches, we also consider two baselines. The first one is a single-view supervised method using all the training data available. The comparison against this baseline allows us to evaluate the tradeoff between amount of labeled data and classification effectiveness. As a second baseline, we consider the same single-view supervised method which takes the same amount of labeled data as the proposed strategies. The comparison against this second baseline allows us to evaluate the impact on the classification of incorporating new examples into the training set.

All five classification approaches use the LAC classifier (Section 3.3) and our user collection (Section 4.1) consisting of 60 attributes. For the multi-view semi-supervised approaches, we separated these attributes into 3 views, as discussed in Section 4.1.2. Note that, unlike in [Benevenuto et al. 2009], where the authors used a Support Vector Machine (SVM) classifier, we here choose to use LAC. This choice is mainly because the estimated probabilities of a user being in a class, essential to the

[3]We do expect any other user collection of the same type, being representative of the entire user population, to contain a much larger number of legitimate users than of spammers and promoters.

multi-view approaches, were found to be much more reliable, according to some initial experiments comparing SVM and LAC. Moreover, we also found that LAC achieved somewhat better results, when 100% of training data is used.

The classification experiments were performed using a 5-fold cross validation. The original sample is partitioned into 5 sub-samples. In each test, four of the sub-samples are used as training data and the remaining one is used as test data. For the semi-supervised approaches, the training data is further partitioned into labeled and unlabeled data sets, and used as explained in Section 3.1. This process is repeated 5 times, with each of the 5 sub-samples used exactly once as the test data.The entire 5-fold cross validation process is repeated 5 times, using different seeds to shuffle the original data set. Thus, the classification results reported in the next section for each considered approach are averages of 25 runs.

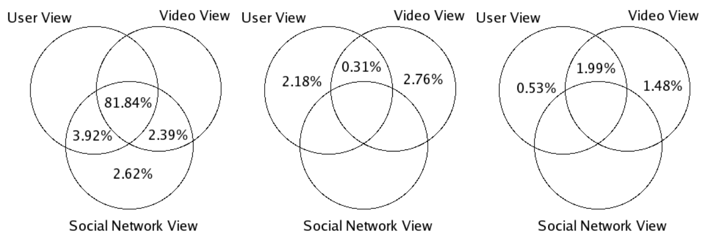
5. EXPERIMENTAL RESULTS

This section presents the most relevant results of our comparison of the different classification approaches considered. All reported results are averages of 25 runs, as explained in the previous section. In all experiments, the test sets (one fold per run) are kept the same for all evaluated approaches. Reported results have standard deviations under 5% of the means.

As explained in Section 4.3, for experimental purposes, when evaluating the multi-view semi-supervised approaches, we need to partition the original training data in each experiment (i.e., 4 folds) into labeled and unlabeled data in order to simulate the situation in which we have a small amount of labeled data for a large amount of unlabeled samples. The goal is to achieve the best tradeoff between the amount of training data, which should be minimum, and the effectiveness of the approach, which should be as close as possible to that of the supervised method using all the available training data. In order to evaluate this tradeoff, we ran a set of initial experiments with our proposed multi-view strategies, increasingly reducing the percentage of the original training data provided as labeled samples for the strategies, while leaving the remaining amount as unlabeled. We tested various percentages, and found that using only 20% of the original training data (and leaving 80% of it as unlabeled) led to the best tradeoff. Next, we focus our comparison on the results when 20% of the original training set is used as labeled data by the multi-view semi-supervised approaches and by the second baseline. Detailed results with other percentages are ommitted due to space constraints, although we briefly discuss some of them at the end of this section.

Before presenting our classification results, we note that, for both multi-view approaches explored in this work, only two views were considered when introducing new examples into the training data: the user view and the video view. The social network view was disregarded because initial classification experiments applying each view in isolation indicated that it is insufficient to distinguish between different user classes, presenting very poor classification effectiveness. These results are illustrated in Figure 1, which shows the intersections, percentage-wise, of the predictions of each view in the first iteration of the multi-view semi-supervised method on the unlabeled set. Figure 1a), which refers to the performance on predicting legitimate users, shows that all predictions are inside the social network view, regardless of the predictions of the other views. This means that the social network view predicts all users as legitimate users. The large percentage found in the intersections of the three views in Figure 1a) reflects the fact that most users are in fact legitimate, which makes it easier to predict for this class. More importantly, in Figures 1b) and 1c), all predictions are outside the scope of the social network view, implying that it is not able to predict any user as either promoter or spammer. Obviously, there is no agreement with the other views regarding these classes. We note that, while the social network view was disregarded while adding new elements into the training data, all 60 attributes are used in the final classification of the test sets. The same holds for the two (supervised) baselines.

We start our analyses by considering the performance of our baselines, i.e., classifiers trained with all training data available (Table I) and with the same 20% used by the multi-view approaches (Table II). For the first baseline, promoters and legitimate users are classified correctly in almost 100% of the



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

Fig. 1: Prediction of classes by each view

cases, but only 53% of spammers are correctly classified, on average[4]. A further investigation revealed that several of these spammers are indeed very hard to identify based only on their behaviors as they act very similarly to legitimate users[5], despite the fact that most of the videos they upload to the system were considered spam. Regarding the second baseline, Table II shows that it produces results that are in fact worse than the previous ones, mainly with regards to the classification of spammers. In particular, the fraction of correctly identified spammers (in the diagonal) dropped by more than 20%. Moreover, the fractions of promoters and of spammers that were misclassified as legitimate users increases by at least the same factor. As discussed in Section 4.2, such loss in classification performance is particularly worrisome as those users would probably escape undetected.

Table I: Classification with Baseline 1 (Supervised Method with 100% of Training Data)

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

Table II: Classification with Baseline 2 (Supervised Method with 20% of Training Data)

			Predicted	
		Promoter	Spammer	Legitimate
	Promoter	96.13%	1.29%	2.58%
True	Spammer	3.46%	41.92%	54.62%
	Legitimate	0%	2.57%	97.43%

Table III shows the confusion matrix obtained with the multi-view classification based on the View Agreement strategy. The problem with this approach is that the views do not agree with regard to the majority of spammers and promoters, as can be (partially) seen in Figures 1b) and 1c). Recall that the process of adding new samples to the training set stops once no agreement is obtained with respect to any class. Thus, this approach ends up adding only a small number of new promoters and spammers to the training set, limiting its effectiveness. When it comes to promoters, in spite of the small agreement rate, this approach is still able to add enough of them during all iterations of the algorithm so as to keep the initial proportions (see discussion in Section 3), because the number of promoters in the initial training set is very small. The real problem is with spammers. Figures 1b) and 1c) show that the agreement between the two views is even lower for this class. As consequence, the process of introducing new samples into the training set stops after only a few iterations of the algorithm (maximum of two, in our experiments), and very few new spammers (and users in general) are included in the training set. Thus, similarly to the baseline with 20% of the training set, this approach is very ineffective in the prediction of spammers. Moreover, semi-supervised approaches may add to the training set a few number of samples with incorrect classes. When the process continues for several iterations, this problem may become less prominent, if the majority of the insertions are correct. However, since

[4]We should notice that these results are slightly different from those reported in [Benevenuto et al. 2009] because the classifiers used are different: we here use LAC, whereas the previous work used SVM.
[5]We found no clear distinction between the values of several attributes of such spammers and the typical values of the same attributes in legitimate users.

here the number of iterations is very small, any sample inserted with the wrong class may significantly impact the classifier in its future decisions, ultimately leading it to wrong predictions. This is specially important in the case of promoters, as the number of elements of that class is very small.

Table III: Classification with the View Agreement Approach

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	89.68%	7.10%	3.23%
	Spammer	3.21%	43.59%	53.21%
	Legitimate	0%	3.13%	96.87%

Clearly, spammers is the most difficult class to predict, by both baselines and by the View Agreement approach. We should note that the much lower agreement between the two views with regard to that specific class may ultimately impact the multi-view classification of all three classes, as it causes the earlier interruption of the process of adding new samples (of all classes) to the training set. As a matter of fact, we did observe in our experiments that the great disagreement between views for spammers is the main factor limiting the continuity of the training expansion process. In spite of that, we note that the results of that approach for promoters and legitimate users are reasonably good (Table III), as they are based on enough agreement with higher confidence. Thus, we next explore a hybrid approach that applies the Borda Count algorithm, described in Section 3.2, only for spammers, keeping the View Agreement strategy for the other 2 classes. In other words, we create the candidate sets of legitimate users and promoters according to the agreement between the two views (as in Algorithm 1) and the candidate set of spammers according to the final aggregated ranking produced by Algorithm 2. We then select users from each set according to the corresponding criteria (confidence for legitimate users and promoters, and rank for spammers), keeping the relative proportions similar as in the initial training set.

Table IV shows the confusion matrix with the results of this hybrid approach. In comparison with the baseline with 100% of training, this approach is only slightly worse in predicting promoters and legitimate users. However it achieves comparable (and, in some folds, slightly better) performance when it comes to correctly identifying spammers, the hardest class. These results are indeed quite promising considering the great reduction (by a factor of 5) on the required amount of labeled data. Moreover, given the envisioned application of our technique as a tool to help system administrators by filtering suspicious users for further (manual) investigation, we believe the results for promoters and legitimate users are also quite positive. The small fraction of misclassified promoters were considered as spammers, that is, they were predicted at least as non-cooperative users. Moreover, the fraction of misclassified legitimate users is reasonably small. The misclassification of such users could be reversed during manual investigation. Specifically for the collection used in this work, the reduction in the required amount of labeled data means that 530 less users need to be manually evaluated, while the increase of misclassified legitimate users means that only 10 additional legitimate users (7.37% of them) need to be manually evaluated after the classification process. In very large user collections, this tradeoff needs to be better studied. However, it is always worth to remind the various costs associated with the action of spammers which justify a more aggressive approach towards identifying them. There are direct costs associated with the use of bandwidth, network, and cache, and indirect costs due to a possible loss of credibility and reputation of the service from the dissatisfaction of the users, which are perhaps even worse than the direct costs. Because of this, a small increase in the number of users that have to be manually inspected later is acceptable, provided there is an increase in the amount of spammers being identified correctly. In comparison with the baseline with 20% of training, our approach is able to improve spammer detection by 35%, at the cost of only a slight degradation (6%) in the correct classification of legitimate users. We should also note that, in comparison with the basic View Agreement approach, the hybrid strategy improved the correct classification of both classes of non-cooperative users, indicating that, indeed, the very low agreement of both views with regard to spammers impacted the classification of both classes.

For sake of completeness, we also experimented with the Borda Count approach applied to all three classes. The use of Borda Count approach for the three classes has proved to be worse than the hybrid

Table IV: Classification with the Hybrid Borda Count / View Agreement Approach

		Predicted		
		Promoter	Spammer	Legitimate
True	Promoter	96.77%	3.23%	0%
	Spammer	3.21%	56.67%	40.13%
	Legitimate	0.09%	8.15%	91.76%

approach used before, erroneously classifying 27% of promoters and 57% of spammers. The Borda count uses each view confidence individually in the sense that it first produces a ranking of users for each view before the rank aggregation takes place. Thus, misclassifications with high confidence (e.g., spammers classified as legitimates with high confidence) may cause wrong insertions into the training data that can be avoided when the insertions are based on the high levels of agreement that occur specifically for the legitimate and promoters classes, as observed in our datasets. Thus, the additional spammers and promoters incorrectly added to the training set when compared to the view agreement approach, caused an overall drop in the classification effectiveness.

Finally, we also analyze the performance of the methods under the macro-F1 metric. The results for the supervised method with 100% and 20% of training data are 0.86 and 0.78, respectively. In comparison, the macro-F1 results for the View Agreement and the Hybrid approaches are, respectively, 0.77 and 0.8. Thus, in terms of this metric, the multi-view approach combining View Agreement and Borda Count strategies is slightly better than the other approaches using the same amount of training (20%), and it is only around 7% worse than the classifier with 5 times more training data. Nevertheless, if the correct classification of non-cooperative users is favored at the expense of misclassifying legitimate users, the results in Tables I-IV show much more clearly the superiority of the proposed Hybrid View Agreement / Borda Count approach.

As a final note, we recall that 20% of the original training set is the smallest amount of labeled data required to produce competitive results in comparison with the supervised approaches. Just to illustrate, if we further reduce it to 15% of the original training set, the fraction of correctly classified promoters drops sharply to only 52%. The classifier requires a minimal amount of examples of each class to learn the needed patterns to identify specific types of behavior. If a particular class does not have enough examples, the classifier will not have enough information about the elements of this class, thus incurring in misclassifications. When we used only 15% of the original training set as labeled data, the number of promoters, the smallest class, used for learning, was very small (just three examples), compared with the amount of promoters when we used 20% of the training set (five examples). Thus, in the first iteration of the semi-supervised algorithm, there were incorrect insertions of promoters into the training set, i.e., promoters are inserted as erroneous examples of other classes. This error caused the number of incorrect insertions of promoters to increase further over the additional iterations, which explains the verified loss in the effectiveness of the classifier, mostly due to the misclassified promoters.

6. CONCLUSIONS AND FUTURE WORK

Our proposed method explores a multi-view semi-supervised classification algorithm, which requires a much smaller amount of training data than previously proposed supervised methods. We evaluate two approaches, built from two different strategies for combining results from multiple views, using a sample of pre-classified users and a set of user behavior attributes. In comparison with supervised methods, our best approach, which combines the View Agreement and the Borda Count view combination strategies, achieves a much more favorable tradeoff between detecting non-cooperative users and reducing amount of training data, at the possible expense of a slight increase in the fraction of misclassified legitimate users. In particular, it achieves comparable performance, particularly on detecting spammers (the hardest class), with 5 times fewer training data than the supervised method.

As future work, we intend to build a new, much larger, user test collection as well as explore the use of fourth view - the video content view - as means to further improve spammer detection, by more clearly distinguishing them from legitimate users.

REFERENCES

Agrawal, R., Imieliński, T., and Swami, A. Mining association rules between sets of items in large databases. In *SIGMOD*. pp. 207–216, 1993.

Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Gonçalves, M. Detecting spammers and content promoters in online video social networks. In *SIGIR*, 2009.

Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Ross, K. Video interactions in online video social networks. *ACM Trans. Multimedia Comput. Commun. Appl.* 5 (4): 1–25, 2009.

Black, D. *The theory of committees and elections*. Cambridge University Press, London :, 1958, 1963.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *COLT*. pp. 92–100, 1998.

Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* 30 (1-7): 107–117, 1998.

Castillo, C., Donato, D., Murdock, V., and Silvestri, F. Know your neighbors: Web spam detection using the web topology. In *SIGIR*, 2007.

Christoudias, C., Urtasun, R., and Darrell, T. Multi-view learning in the presence of view disagreement. In *UAI*, 2008.

comScore. comscore releases march 2010 u.s. online video rankings. http://www.comscore.com/Press_Events/Press_Releases/2010/4/comScore_Releases_March_2010_U.S._Online_Video_Rankings, 2010a.

comScore. comscore releases march 2010 u.s. search engine rankings. http://www.comscore.com/Press_Events/Press_Releases/2010/4/comScore_Releases_March_2010_U.S._Search_Engine_Rankings, 2010b.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank aggregation methods for the web. In *WWW*. pp. 613–622, 2001.

Fetterly, D., Manasse, M., and Najork, M. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *WebDB*. pp. 1–6, 2004.

Gomes, L. H., Cazita, C., Almeida, J. M., Almeida, V., and Meira, Jr., W. Workload models of spam and legitimate e-mails. *Perform. Eval.* 64 (7-8): 690–714, 2007.

Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. Combating web spam with trustrank. In *VLDB*. pp. 576–587, 2004.

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

Kohavi, R. and Provost, F. Glossary of terms. *Special issue on applications of machine learning and the knowledge discovery process* 30 (2-3): 271–274, 1998.

Koutrika, G., Effendi, F. A., Gyöngyi, Z., Heymann, P., and Garcia-Molina, H. Combating spam in tagging systems. In *AIRWeb*. pp. 57–64, 2007.

Langville, A. N. and Meyer, C. D. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, Princeton, NJ, USA, 2006.

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

Newman, M. and Park, J. Why social networks are different from other types of networks. *Physical Review E* 68 (3): 36122, 2003.

Thomason, A. Blog spam: A review. In *CEAS*, 2007.

Veloso, A., Meira, Jr., W., Cristo, M., Gonçalves, M., and Zaki, M. Multi-evidence, multi-criteria, lazy associative document classification. In *CIKM*. pp. 218–227, 2006.

Veloso, A., Meira Jr., W., and Zaki, M. J. Lazy associative classification. In *ICDM*. pp. 645–654, 2006.

Veloso, A. A., Almeida, H. M., Gonçalves, M. A., and Meira Jr., W. Learning to rank at query-time using association rules. In *SIGIR*. pp. 267–274, 2008.

Wu, C.-T., Cheng, K.-T., Zhu, Q., and Wu, Y.-L. Using visual features for anti-spam filtering. In *ICIP*. pp. 509–512, 2005.

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I. Spamming botnets: signatures and characteristics. In *SIGCOMM*, V. Bahl, D. Wetherall, S. Savage, and I. Stoica (Eds.). pp. 171–182, 2008.

Yang, Y. An evaluation of statistical approaches to text categorization. *Information Retrieval* 1 (1-2): 69–90, 1999.

YouTube. Youtube fact sheet. http://www.youtube.com/t/fact_sheet, 2010.