Spammers Detection from Product Reviews: A Hybrid Model

Zhiang Wu[1], Youquan Wang[1], Yaqiong Wang[2], Junjie Wu[3,*], Jie Cao[1,*], and Lu Zhang[1]
[1]School of Information Engineering, Nanjing University of Finance & Economics, Nanjing, China
[2]Carlson School of Management, University of Minnesota, Twin Cities, USA
[3]School of Economics & Management, Beihang University, Beijing, China
[*]corresponding author: *wujj@buaa.edu.cn, jie.cao@njue.edu.cn*

Abstract—**Driven by profits, spam reviews for product promotion or suppression become increasingly rampant in online shopping platforms. This paper focuses on detecting hidden spam users based on product reviews. In the literature, there have been tremendous studies suggesting diversified methods for spammer detection, but whether these methods can be combined effectively for higher performance remains unclear. Along this line, a hybrid PU-learning-based Spammer Detection (hPSD) model is proposed in this paper. On one hand, hPSD can detect multi-type spammers by injecting or recognizing only a small portion of positive samples, which meets particularly real-world application scenarios. More importantly, hPSD can leverage both user features and user relations to build a spammer classifier via a semi-supervised hybrid learning framework. Experimental results on movie data sets with shilling injection show that hPSD outperforms several state-of-the-art baseline methods. In particular, hPSD shows great potential in detecting hidden spammers as well as their underlying employers from a real-life Amazon data set. These demonstrate the effectiveness and practical value of hPSD for real-life applications.**

I. Introduction

Online product ratings and reviews strongly influence purchase decisions of vast customers [1]. A high proportion of positive reviews with a high average rating could bring significant financial gains, whereas the opposite might cause great loss. As a result, some product providers and/or merchants have strong incentives to create biased online reviews and extreme ratings. Product review spammer detection [2], [3] has therefore gained particular interests in recent years.

A plethora of methods and systems have been devised for detecting spam reviews, spam users and even spam groups [4], [5], [6], [2], [3]. Most of them propose to use features of review contents and reviewer behaviors for constructing various classifiers. The success of these methods depend in large on the assumption that spammers' abnormal behaviors can be captured by effective features. In reality, however, the behaviors of some cunning spammers are almost the same as normal users. Take the user *chenyanyan*[1] in Amazon.cn as an example. According to her homepage, she wrote many reviews containing rich content, received many "useful" votes, and purchased many books — seemingly an absolute normal user. However, if we look into her rated products, nearly all rated books were released by a same issuer named "Xindao Culture". This implies that she is actually a book promoter. Motivated by this, we believe the linkages between users and products can be exploited to enhance spammer detection. If a user often rates spammed products, she is more likely to be a spammer; similarly, if a product is often rated by spam users, then it is highly suspicious. How to incorporate linkage information into feature-based classifiers, however, is still an open problem.

There are some other intractable problems in spammer detection. First, spam might exist in multiple types, i.e., [7] gives three types of review spam, which calls for more elaborate modeling. Second, most previous studies either build supervised classifiers [8], [2] or develop unsupervised ranking algorithms [9], [5], which do not agree with the fact that we are often given a few labeled but a vast majority of unlabeled user samples. Third, it is not clear how many spammers existing in real-life e-commerce sites, which makes the validation of detection very difficult.

In this paper, we establish a hybrid PU-learning-based Spammer Detection model called hPSD. hPSD generally has three distinct properties. First, it employs PU-learning to detect multi-type spammers by injecting or labeling only a small portion of positive samples. Also, a novel reliable negative set extraction algorithm PU-learning is designed for PU learning. Second, a hybrid learning model based on Bayesian inference is presented, which allows integrating user-product relation into feature-based learning process. Third, hPSD turns PU learning into semi-supervised learning, which help make full use of both labeled and unlabeled data.

Extensive experiments are conducted on both movie and Amazon data sets. By injecting some artificial shilling attackers into MovieLens and Netflix datasets, hPSD first provides comparative results with eight state-of-the-art shilling attack detectors. hPSD is then utilized to identify duplicate spammers and promoters hidden inside the Amazon.cn dataset. Interestingly, we indeed discover a number of hidden spammers and collusive book publishers under high suspicion of malfeasant marketing.

II. Related Work

In the literature, existing techniques are summarized for detection of three targets [1]: review spam, spam users and spammer groups. Among these, review spam detection has

[1]http://www.amazon.cn/gp/cdp/member-reviews/A1C1M070G5SF67?ie=UTF8&display=public&page=10&sort_by=MostRecentReview

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.73

received dominant research attention [4], [10], [5], [11], [6]. It usually represents a review using a set of review-, reviewer- and product-level features, and thus tries to construct the classification model. Although our work belongs to user-centric detection, the study [5], [6] inspires us a lot on defining features for Amazon data.

Spam users come in a variety of types including product/store review spammers [2], [3], shilling attackers [12], [22], marionette microblog users [13], [14], video promoters [15], and so on. The primary detection methods still utilize various machine learning techniques to construct a classifier based on a set of effective features. Thus, spammers in clever disguise are probably to evade the detection. To further improve the detection accuracy, this paper proposes a principled hybrid learning model exploiting both user's features and user-product relation. Nevertheless, the mutual relation between user and item has ever been utilized to reveal search engine spam [16] and evaluate the distortion led by attackers [17]. To the best of our knowledge, however, no theoretical models to connect two forms of data have been proposed.

III. hPSD: The Model

In this section, we introduce the hPSD model and highlight its essential components. We first give some basic notation. Suppose we are given an unlabeled set U of n users and a set I of m products. Let $R_{i,j} \in \{0, 1\}$ be a binary relation variable, indicating user i has reviewed product j. Let P denote a positive set of some labeled spammers, by artificial synthesis or by manual labeling. Thus, the spammer detection problem can be described as: Given U, P and R, to identify spam users from U.

The procedue of hPSD can be described shortly as follows. We first specify multiple types of spammers based on domain knowledge, each of which is given a P set. hPSD then iteratively detects each type of spammers, by firstly discretizing user feature values, and then extracting reliable nagative set against P, and finally incorporating R into semi-supervised learning for hybrid modeling of spammers. During this procedure, feature discretization, reliable negative set extraction, and hybrid learning scheme are three essential components of hPSD, which are detailed below.

A. Feature Discretization

In spam detection [5], [6], [2], [8], almost all features are numerical. However, modeling the continuous feature directly is not suitable for the spammer detection problem. The major reason is that we do not have any prior knowledge about the feature distribution. Even if the distribution is known or assumed, the distribution on labeled and unlabeled set is probably to be non-identical,

Given any numerical feature f, its possible values of both P and U form a sorted list S. If we want to discretize f into ν categories, we need to find $\nu - 1$ cut points in S. Among a lot of criterion for determining a cut point, we employ the widely-used minimal weighted average variance (WAV) [18].

TABLE I
Two-way Table

		P	U
f	1	a	b
	0	c	d
Σ_{col}		$\lvert P\rvert$	$\lvert U\rvert$

Assume a value v is used to split S into two parts: S_1^v and S_2^v. The WAV of v on S is defined as:

$$\text{WAV}_{vS} = \frac{|S_1^v|}{|S|}\text{Var}(S_1^v) + \frac{|S_2^v|}{|S|}\text{Var}(S_2^v), \tag{1}$$

where $|S|$ is the number of points in this list, and $\text{Var}(S_1^v)$ is the variance of all points in this list (analogical to $|S_1^v|$, $|S_2^v|$ and $\text{Var}(S_2^v)$). Thus, the "best" cut point has the maximum value of $\Delta_v = \text{Var}(S) - \text{WAV}_{vS}$.

To obtain multi-interval discretization, we present a Bisecting V-Clustering algorithm to divide S into ν sub-lists in a binary-recursive way. It first divides all value-points of f (i.e., S) into two clusters based on the best cut point; and then repeatedly selects one cluster with the largest range of value, and divides that cluster into two clusters based on the best cut point again. The procedure will continue unless ν clusters are found. Thus, if we let $F = \nu V$, $u_i \in \mathbb{R}^F$ $1 \le i \le n$ refers to a categorical feature vector of a user, where $u_{il} \in \{0, 1\}$ $1 \le l \le F$ denotes whether ith user's feature contains lth value.

B. Reliable Negative Set Extraction

This step aims to single out a small set of instances from U that are significantly different with instances in P. We do not require RN include ample instances but emphasize that the selected instances should be "reliable". Since it is commonly $|P| \ll |U|$, we target at extracting a RN set with $|RN| \approx |P|$, to avoid the class imbalance in the learning step.

In text classification [19], a reliable negative document is regarded as a document in U, such that it does not contain any word in the core vocabulary of P. In words, if given a set of core features, the feature strength (a.k.a., the discriminative power) between P and RN is expected to be maximized. Then the objective function may be written as follows:

$$O_1 : \max_{f \in F^c} D_f(P \cup RN), \tag{2}$$

where F^c is the set of core features and D_f denotes a feature strength function. Obviously, to maximize Eq. (2) is a NP-hard problem. As an alternative, we proceed to design a greedy RN set extraction heuristic.

For better illustration, we represent a binary feature u_{il} as a two-way table, as shown in Table I. Theoretically speaking, the existing metrics such as information gain, χ^2 and odd ratio can be used to define D_f. However, in Table I, since both $a+c$ and $b+d$ are constants, we proceed to define a simplified feature strength exploiting only a and b as

$$D_f = n_P(f) \log \frac{|P| + |U|}{n_P(f) + n_U(f)} = a \log \frac{n}{a+b}, \tag{3}$$

where $n_P(f) = a$ and $n_U(f) = b$ are the number of instances containing f in P and U, respectively. The basic premise of D_f function is that if a feature is discriminative for the class P, this feature should frequently appear in class P but infrequently appear in the remaining instances.

Algorithm 1 Reliable Negative Set Extraction

```
Input: Positive set $P$; Unlabeled set $U$;
Output: A set of reliable negative instances $RN$, and $RN \leftarrow U$ initially;
 1: procedure RN_EXTRACTION($P, U$)
 2:     for each feature $f_l \in P$ do            ▷ only consider features appearing in $P$
 3:         Compute $D_{f_l}$ using Eq. (3);
 4:     end for
 5:     for examine each feature $f_l$ in $D$-decreasing order do
 6:         Remove instances containing $f_l$ from $RN$;
 7:         if the size of $RN$ is close to that of $P$ then
 8:             return $RN$;
 9:         end if
10:     end for
11: end procedure
```

Given a feature, since $b \swarrow \Rightarrow D_f \nearrow$, to remove instances with $f = 1$ from U (i.e., let $b = 0$) will maximize the objection function. Therefore, the RN extraction problem can be described as: given $RN = U$ initially and a sorted list of features, to remove instances containing this feature in RN, until the scale of RN is reduced approximately to that of P. Algorithm 1 summarizes this procedure. We stress that due to the uncontrollable b, $|RN|$ is unlikely equal to $|P|$. In line 7, we establish the exit criteria as $|RN|$ being closest to $|P|$.

C. Model and Inference

Now, the entire data can be represented $\mathcal{D} = L \cup U,\ L = P \cup RN$ containing a few labeled but majority unlabeled users. We denote class labels as $y_k,\ k = \{0, 1\}$ and assume that given a class label y_k, each feature obeys a multinomial distribution with parameters $\theta_k \in \mathbb{R}^F$. θ_{kl} is the probability that the lth feature-value pair occurs in class k, which satisfies $\sum_{l=1}^{F} \theta_{kl} = 1$. Each instance is assumed to be drawn independently from a mixture distribution of k classes and thus the probability that u_i belongs to class k is $p(y_k|u_i;\theta_k) = \frac{z_k p(u_i|y_k;\theta_k)}{\sum_k z_k p(u_i|y_k;\theta_k)}$, where $z_k = p(y_k)$ is the prior probability of class k which satisfies $\sum_k z_k = 1$. The class-conditional probability of an instance is

$$p(u_i|y_k;\theta_k) = \prod_{l=1}^{F} p(u_{il}|y_k;\theta_k) = \prod_{l=1}^{F} \theta_{kl}^{u_{il}}. \tag{4}$$

The objective of learning targets at maximizing conditional likelihood on $\mathcal{D}$. Meanwhile, the proposed model needs to incorporate user-product relation into the traditional learning on the feature space. To this end, we have:

$$O_2:\ \max_{\theta} \log \prod_{u_i \in \mathcal{D}} \underbrace{\{\underbrace{p(y_k|u_i;\theta_k)^{\Lambda_i}}_{(a)} \prod_{I_j \in R_{i\cdot}} p(y_k|I_j;\theta_k)^{\frac{d}{|R_{i\cdot}|}}}_{(b)}\}, \tag{5}$$

satisfying $\sum_{l=1}^{F} \theta_{kl} = 1$ and $\sum_k z_k = 1$, with

$$p(y_k|I_j;\theta_k) = \prod_{u_i \in R_{\cdot j}} p(y_k|u_i;\theta_k)^{\Lambda_i}, \text{ and } \Lambda_i = \begin{cases} \lambda, & \text{if } u_i \in U, \\ 1, & \text{if } u_i \in L. \end{cases}$$

In Eq. (5), $R_{i\cdot}$ is a set of products rated by u_i, $R_{\cdot j}$ is a set of users rating I_j, $d \in [0, 1]$ is the coefficient that balances between the feature space and user-product relation, and $\lambda \in [0, 1]$ is the weight that reduces the impact of unlabeled set. Meanwhile, the part (a) of Eq. (5) is the probability learned from the feature space, while the part (b) is learned from the user-product relation. With $p(u_i|y_k;\theta_k)$ and the Bayesian theorem, we derive the conditional likelihood

$$p(y_k|u_i;\theta_k) = \frac{z_k p(u_i|y_k;\theta_k)}{\sum_k z_k p(u_i|y_k;\theta_k)} = \frac{z_k \prod_{l=1}^{F} \theta_{kl}^{u_{il}}}{\sum_k z_k \prod_{l=1}^{F} \theta_{kl}^{u_{il}}}. \tag{6}$$

The Lagrange function of Eq. (5) is

$$l:\ \max_{\theta} \log \prod_{u_i \in \mathcal{D}} \{p(y_k|u_i;\theta_k)^{\Lambda_i} \prod_{I_j \in R_{i\cdot}} p(y_k|I_j;\theta_k)^{\frac{d}{|R_{i\cdot}|}}\}$$
$$+ \sum_{k=0}^{1} \xi_k (\sum_{l=1}^{F} \theta_{kl} - 1) + \omega(\sum_{k=0}^{1} z_k - 1). \tag{7}$$

Two sets of parameters, namely z_k and θ_k, need to be estimated here. They could be resolved by an EM-like algorithm. In the M-step, to update z_k, we take the partial derivative of Eq. (7) with respect to z_k. Note that the Stochastic Gradient Training (SGT) [20] is used here. GST treates the derivative on a random sample as an approximation to the derivative on the training data, and updates parameters to increase the conditional log likelihood through one random example at a time. Precisely, the partial derivative of the ith user is

$$\frac{\partial l}{\partial z_k} = \frac{\Lambda_i}{z_k} - \frac{\Lambda_i V_k}{\sum_k z_k V_k} + \sum_{I_j \in R_{i\cdot}} M_i \sum_{u_i \in R_{\cdot j}} (\frac{\Lambda_i}{z_k} - \frac{\Lambda_i V_k}{\sum_k z_k V_k}) + \omega \tag{8}$$

where $V_k = \prod_{l=1}^{F} \hat{\theta}_{kl}^{u_{il}}$ and $M_i = \frac{d}{|R_{i\cdot}|}$. Thus, let the derivative in Eq. (8) to be zero. We get

$$\frac{p-1}{p - pz_0 + z_0} + \frac{C}{z_0} = \frac{C}{1 - z_0} - \sum_{I_j \in R_{i\cdot}} M_i \sum_{u_i \in R_{\cdot j}} \frac{q-1}{z_0 - qz_0 + q} = 0,$$

where $z_0 + z_1 = 1$, $C = 1 + d|R_{i\cdot}|$, $p = \prod_{l=1}^{F} (\frac{\theta_{1l}}{\theta_{0l}})^{u_{il}}$, and $q = \prod_{l=1}^{F} (\frac{\theta_{1l}}{\theta_{0l}})^{u_{il}}$. Since z_0 is difficult to resolve directly, we make some approximations here. Specifically, we take βz_0 as an estimation of 1. This implies that z_0 is initially estimated as the ratio of unlabeled users, and then is estimated as z_0 of last iteration. β would also change with z_0 in each iteration. With this approximation, we finally obtain equations for updating both z_0 and z_1.

$$\hat{z}_0 = z_0 + \zeta \frac{X + C + \sum_{I_j \in R_{i\cdot}} M_i \sum_{u_i \in R_{\cdot j}} Y}{X + 2C + \sum_{I_j \in R_{i\cdot}} M_i \sum_{u_i \in R_{\cdot j}} Y}, \tag{9}$$

$$\hat{z}_1 = z_1 + \zeta \frac{C}{X + 2C + \sum_{I_j \in R_{i\cdot}} M_i \sum_{u_i \in R_{\cdot j}} Y}, \tag{10}$$

where $X = \frac{p-1}{\beta p - p + 1}$, $Y = \frac{q-1}{\beta q - q + 1}$, and ζ is the learning rate to control the magnitude of the changes to parameters. We commonly set $\zeta = 0.01$. Denote a smoothing parameter as α, the number of times the lth feature occurs in kth class as n_{kl}, and the total number of instances in kth class as n_k. A smoothed estimator of the multinomial distribution is $\hat{\theta}_{kl} = \frac{n_{kl} + \alpha}{n_k + \nu\alpha}$. We follow common practice by setting $\alpha = 1$. To update θ_{kl} is equivalent to update n_{kl} and n_k as follows.

$$\hat{n}_{kl} = \sum_{u_i \in L_k} u_{il} + \lambda \sum_{u_i \in U} p(y_k|u_i;\theta_k) \prod_{I_j \in R_{i\cdot}} p(y_k|I_j;\theta_k)^{\frac{d}{|R_{\cdot j}|}} u_{il},$$

$$\hat{n}_k = |L_k| + \lambda \sum_{u_i \in U} p(y_k|u_i;\theta_k) \prod_{I_j \in R_{i\cdot}} p(y_k|I_j;\theta_k)^{\frac{d}{|R_{\cdot j}|}}. \tag{11}$$

Algorithm 2 Hybrid Learning Algorithm

```
Input: Labeled set $L = P \cup RN$; Unlabeled set $U$; Weights: $\lambda$, $d$;
Output: The probability $p(y_k|u_i;\theta_k)$ for each user in $U$;
1: procedure HYB_LEARNING($L$, $U$, $\lambda$, $d$)
2:     Compute $n_{kl}$, $n_k$, and thus $\theta_k$, $z_k$ on $L$, initially;
3:     while $\max_{k,l} |\theta_{jl} - \hat{\theta}_{jl}| \geq \epsilon$ do
4:         Compute $p(u_i|y_k;\theta_k)$ and thus $p(y_k|u_i;\theta_k)$ by Eqs. (4) and (6);
5:         Update $z_k,\ k \in \{0,1\}$ by Eqs. (9) and (10);
6:         Update $n_{kl}$, $n_k$ and thus $\theta_k$ by Eq. (11);
7:     end while
8: end procedure
```

To sum up, we follow an EM-like procedure in order to maximize the conditional likelihood. Basically we iteratively update parameters z_k and θ_k by taking the derivative of the Lagrange function of our objective function. Based on the above inference, we can summarize the hybrid learning procedure, as shown in Algorithm 2.

IV. Experimental Results

Here we evaluate the effectiveness of hPSD by comparing it with various baseline methods on movie and Amazon datasets.

A. Detecting Shilling Attackers in Movie Data

Two benchmark movie datasets are used, as shown in Table II. `u2.base` is one of random splits of the MovieLens 100K dataset. To obtain `s1`, we randomly sample 3000 users from Netflix dataset and delete movies rated less than 20 times.

1) Anomaly Injection: We take *hybrid* shilling attackers with the *push* intent [12], [21], [8], [22] as malicious users to be detected. According to [8], [21], [22], we categorize six kinds of shilling attackers, as shown in Table III. Thus, 120 and 300 attackers equally composed of six types are injected into `u2.base` and `s1`, respectively. Note that original users in both datasets are assumed to be normal.

2) Feature Construction: We here construct ten features in total. Among them, seven features are collected from the literature [8], [22], including Entropy, DegSim, LengthVar, RDMA, FMTD, GFMV and TMF. In addition, we define three new features: popularity rank (PopRank), average distance with other users (DistAvg), and category entropy (CatEnt). PopRank measures the popularity over items rated by a user, and is defined as $\text{PopRank}_i = \sum_{I_j \in R_{i\cdot}} |R_{\cdot j}|/|R_{i\cdot}|$.

One effective way to maximize attackers' predicted values is that to construct a profile which is moderately correlated to a large number of users in order to affect them [9]. In allusion to this rule, we define $\text{DistAvg}_i = \sum_{j=1}^{n} 1 - PCC_{ij}/n$, where PCC_{ij} represents the Pearson Correlation Coefficient.

A basic observation is that a normal user is probably to have fixed interests on movies or products within limited categories, while an attacker select filler items at random or over popular, which may results in items scattered on lots of categories. Based on the category classification provided by movie datasets, we define $\text{CatEnt}_i = -\sum_{g=1}^{G} \frac{S_{ig}}{S} \log_2 \frac{S_{ig}}{S}$, where G is the number of categories, S_{ig} is the number of u_i's rated movies falling in the gth category, and $S = \sum_{g=1}^{G} S_{ig}$. We cannot rule out the possibility that an enthusiast also have a wide range of interests, which implies no single feature can precisely differentiate spammers from normal users.

TABLE II
Characteristics of Experimental Datasets

Dataset	#User	#Item	#Rating	Density
`MovieLens_u2.base`	943	1682	80000	5.04%
`Netflix_s1`	3000	6237	655908	3.51%
`Amazon`	9424	19185	469393	0.26%

TABLE III
Categorization of Shilling Attackers

Rating / Selection	RFM	AFM
Random	Random Attack (Ran)	Average Attack (Avg)
Popular	Random-over-Popular (RoP)	Average-over-Popular (AoP)
Combination	Bandwagon (BanRan)	Bandwagon (BanAvg)

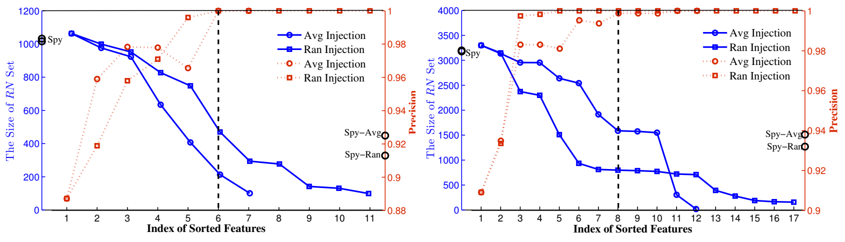
Note: (1) "Selection→ Random" and "Selection→ Popular" denote that attakers randomly select items or select popular items to make them look normal; (2) "Selection→ Combination" denotes that attakers select a part of popular items and randomly select remaining items; (3) Random-Filler Model (RFM) and Average-Filler Model (AFM) mean to rate filler items as ratings or average ratings.

3) Baselines and Evaluation Metrics: We use eight shilling attack detectors as baseline methods in comparison to our hPSD. Firstly, three supervised classification models in WEKA are selected, i.e., C4.5, SVM, and Naïve Bayes (NB). Particularly, we employ two kinds of settings to generate six baseline supervised detectors, i.e., using continuous features and discrete features. To construct the training set, we inject 50 and 150 attackers, equally composed of random and average types, into `u2.base` and `s1` respectively. We further implement two famous unsupervised detectors in MATLAB, i.e., PCA [23] and MDS [21]. They directly run on the user-item rating matrix rather than feature space. To set tunable parameters, i.e., the number of identified attackers of PCA and the number of clusters of MDS, the results with the best setting are reported.

For hPSD, we inject average and random attackers successively as set P, where $|P| = 50$ and 150 for MovieLens and Netflix respectively. In the following experiments, unless stated otherwise, we simply set $\lambda = 0.5$ and $d = 0.2$. Since the ground-truth is known, we adopt the standard metrics such as recall (R), precision (P), and F-measure (F). Note that all these metrics are computed on the spammer class.

4) Overall Comparison: Table IV displays comparative results as the increase of filler size (FS), the ratio of filler items to all items. Note that the lowest FS values of both datasets are close to the average length of user rating. Two observations are noteworthy. First, our hPSD has the overwhelming performance advantages over other detectors. Also encouragingly, the R values of hPSD consistently are the highest, which implies it can effectively identify nearly all injected attackers. But the supervised detectors usually detect few attackers, which results in higher P values yet lower R values. Second, it is obvious that C4.5*, SVM* and NB* are superior to C4.5, SVM and NB. This well validates the vital role of the feature discretization.

5) Quality of RN: As mentioned in Section III-B, hPSD iteratively examines features in D_f descending order and removes instances containing this feature from the current RN. In each round, we track the size of RN set and the number of truely negative instances in RN (denoted as $|TRN|$). Thus, the precision is $|TRN|/|RN|$. Initially, the precision is in fact the ratio of normal users to all users, i.e., about 90%. We also



(a) `MovieLens`, $FS = 10\%$ (b) `Netflix`, $FS = 5\%$

Fig. 1. Precision of the RN extraction.

TABLE IV
COMPARISON AMONG VARIOUS DETECTORS

Metric	Detector	MovieLens				Netflix			
		5%	10%	15%	20%	2%	5%	10%	15%
	hPSD	0.958	0.933	**1**	**1**	0.963	**0.990**	**1**	**1**
	NB⋆	0.750	0.733	0.825	**1**	0.677	0.820	0.833	0.863
	SVM⋆	0.833	0.833	0.975	**1**	0.457	0.663	0.833	0.833
R	C4.5⋆	0.883	**0.958**	0.917	**1**	0.627	0.667	0.667	0.667
	NB	0.642	0.333	0.667	0.667	0.443	0.347	0.387	0.667
	SVM	0.517	0.333	0.650	0.658	0.357	0.577	0.623	0.660
	C4.5	0.450	0.333	0.625	0.685	0.347	0.583	0.337	0.657
	PCA	0.775	0.717	0.667	0.667	0.633	0.597	0.653	0.667
	MDS	0.245	0.508	0.500	0.567	0.263	0.317	0.497	0.333
	hPSD	0.885	0.918	0.938	0.952	0.845	0.892	0.962	0.956
	NB⋆	0.900	0.889	0.934	0.952	0.927	0.942	0.996	0.996
	SVM⋆	0.943	0.926	0.936	0.952	0.890	0.884	0.984	0.984
P	C4.5⋆	0.726	0.723	0.821	0.857	0.908	0.794	0.939	0.957
	NB	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	SVM	0.954	**1**	**1**	**1**	0.877	0.951	0.969	0.971
	C4.5	0.915	0.976	0.987	0.891	0.867	0.931	0.971	0.956
	PCA	0.775	0.717	0.667	0.667	0.633	0.597	0.653	0.667
	MDS	0.365	0.772	0.800	0.883	0.587	0.856	0.877	0.926
	hPSD	**0.920**	**0.926**	0.968	**0.976**	**0.900**	**0.938**	**0.980**	**0.977**
	NB⋆	0.818	0.804	0.876	0.976	0.782	0.877	0.907	0.925
	SVM⋆	0.885	0.877	0.955	0.976	0.604	0.758	0.903	0.903
F	C4.5⋆	0.797	0.824	0.866	0.923	0.742	0.725	0.780	0.786
	NB	0.782	0.500	0.800	0.800	0.614	0.515	0.558	0.800
	SVM	0.670	0.500	0.788	0.794	0.507	0.718	0.759	0.786
	C4.5	0.603	0.497	0.765	0.774	0.495	0.717	0.500	0.779
	PCA	0.775	0.717	0.667	0.667	0.633	0.597	0.653	0.667
	MDS	0.293	0.613	0.615	0.690	0.364	0.462	0.634	0.490

Note: Marking ⋆ denotes features are discretized and thus modeled by multinominal distribution.

employ the `Spy` algorithm [24] for comparison, where 50% instances in P are randomly sampled as spy instances.

Fig. 1 presents the results of two cases: MovieLens with $FS = 10\%$, and Netflix with $FS = 5\%$. As we can see, `Spy` extracted a large RN set including lots of positive instances. The reason is that `Spy` heavily relies on NB* that tends to classify a majority of users as the normal users. However, our method can adjust the scale of RN, and quickly remove all positive instances (i.e., P soars to 1). In particular, when $|RN|$ on both datasets reduces to about 400 and 1000, RN contains none of positive instances. Recall $|RN|$ is set to be close to $|P| = 25, 50$ for two movie datasets. Therefore, the extracted RN set yields 100% accuracy in our experiments.

B. Detecting Attackers Hidden in Amazon Data

Here, we try to use hPSD on a larger proprietary dataset for detecting hidden attackers. The dataset is collected from Amazon China (http://www.amazon.cn) from Sept. 2000 to Dec. 2011. We extract products that have been rated over 15 times, and then pick out users who rated over 20 times. As a result, we obtain an experimental dataset as shown in Table II.

1) Feature Construction: Each record of Amazon dataset contains multiple attributes including User_ID, Product_ID (ASIN), review title, star rating, and posting date. By virtue of the rich information, we construct eight features as attack behavior indicators. Specifically, five features are reformed based on opinion spam behavior indicators [5], [6]: (1) the

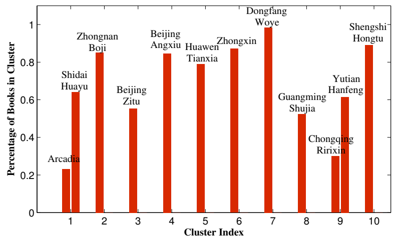


Fig. 2. Dominant publishing company in ten sampled clusters.

difference of first and last rating dates; (2) the number of days when posting ratings; (3) the maximal number of ratings posted in one day; (4) the average ratings per day; (5) the entropy (i.e., variance) on the number of ratings per day. Besides PopRank and CatEnt defined above, we define a novel metric as $\text{SimASIN}_i = \sum_{I_p, I_q \in R_{i\cdot}} \frac{LCS_{p,q}}{|ASIN|}$, to measure the similarity among products' ASIN rated by a user. Here, $LCS_{p,q}$ is the Longest Common Subsequence of two ASIN strings and $|ASIN|$ is a constant 10 in Amazon.

2) Attacker Types: Two types of attackers, i.e., duplicate spammers and promoters, are ubiquitous in real e-commerce platforms [1], [25]. Due to the available review titles, identifying duplicate spammers is relatively simple. A total of 1059 (11.2%) duplicate spammers, with at least 3 identical review titles of which the length exceeds 6 characters, are filtered out. Nevertheless, promoters are more sophisticated and always have the potential marketing goals. Some tricky promoters act almost like normal users and even utilize the fictitious trading to circumvent the detection. In what follows, we showcase the effectiveness of our hPSD model for detecting promoters.

3) Spotting Promoters: A strict principle is used for manually labeling: over 80% of 5-star products rated by the specific user have strong semantic relationships with each other. The semantic relationship among products is experientially judged, such as books/audio/video with the same issuer, cosmetics/cellphones with the same brand, etc. As a result, we manually label 50 promoters and create 6 copies to construct P with 300 instances. hPSD then extracts a RN set with 283 instances and classifies 979 users to the promoter class. So totally 1029 (10.9%) promoters are identified.

We construct a bipartite network containing 1029 user nodes and 4053 product nodes, and each edge represents a user has rated a product. Graclus[2] is used to divide the bipartite network into 30 clusters. By counting the primay product category of

[2]http://www.cs.utexas.edu/users/dml/Software/graclus.html

TABLE V
RESULTS WITH AND WITHOUT HYBRID LEARNING

		Without Hybrid Learning		Σ_{row}
		Normal	Promoter	
With Hybrid Learning	Normal	8411	34	8445
	Promoter	**568**	411	*979*
Σ_{col}		8979	*445*	$\kappa = 0.548$

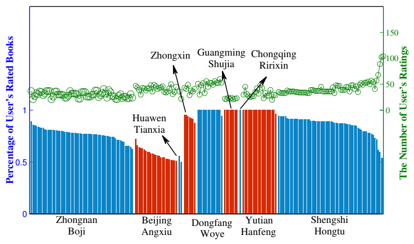


Fig. 3. Abnormal behavior of promoters identified by joint learning.

every cluster, we find that 21 clusters mainly include books and only 9 clusters are about electronics or cosmetics. Behind the appearance that Amazon mainly sells books, we tentatively interpret this as driving sales for books has stronger economic incentive compared with other products. That is, a publishing company commonly released vast books on Amazon and thus it tends to hire promoters to market its own books. To verify this, we sample 10 out of 21 book clusters, and search the primay publishing company of these clusters. Fig. 2 displays the results. It is striking to see that so many users in a cluster assigned almost all ratings to books published by the same company. We therefore believe the identified attackers do contain a high proportion of promoters.

Next, we corroborate the effectiveness of hybrid learning. Table V summarizes the difference of results returned by hPSD with and without hybrid learning. The kappa statistic of 54.8% states two methods achieve a certain consensus. In detail, hPSD with hybrid learning detects extra 568 promoters yet only misses 34 promoters compared with hPSD without hybrid learning. Therefore, we are interested to see whether these 568 newly identified users are abnormal. Recall that we have manually searched 750 books published by 12 deceptive companies, as marked in Fig. 2. Thus, we can compute the percentage of the number of books rated by each newly identified promoter and the number of books released by each deceptive issuer. By filtering the user with its percentage less than 50%, we obtain 204 users concentrating on 9 issuers, as shown in Fig. 3. As we can see, users commonly rate about 50 books, of which a high proportion is released by the same deceptive issuer. This clearly shows these newly identified promoters are indeed abnormal, and thus suffices to verify the effectiveness of hybrid learning.

V. CONCLUSION

This paper proposes a principled hybrid learning model called hPSD to combine both user features and user-product relations for spammer detection. Three essential components of hPSD, including feature discretization, reliable negative set extraction and hybrid learning scheme, are elaborated respectively. Extensive experiments are conducted on both movie data with shilling injection and Amazon data with true yet hidden promoters, to validate the effectiveness and practical value of the proposed model.

ACKNOWLEDGMENT

This research was partially supported by National Natural Science Foundation of China (NSFC) (71571093, 71372188, 61502222), National Center for International Joint Research on E-Business Information Processing (2013B01035), and National Key Technologies R&D Program of China (2013BAH16F03). Junjie Wu was supported in part by National High Technology Research and Development Program of China (SS2014AA012303), NSFC (71322104, 71171007, 71531001, 71471009), Foundation for the Author of National Excellent Doctoral Dissertation of PR China (201189), and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] A. Heydari, M. ali Tavakoli, N. Salim, and Z. Heydari. Detection of review spam: A survey. In ESWA, 2015.
[2] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In CIKM, 2010.
[3] G. Wang, S. Xie, B. Liu, and P. S. Yu. Review graph based online store review spammer detection. In ICDM, 2011.
[4] N. Jindal and B. Liu. Review spam detection. In WWW, 2007.
[5] A. Mukherjee, B. Liu, and N. Glance. Spotting fake reviewer groups in consumer reviews. In WWW, 2012.
[6] A. Mukherjee, A. Kumar, B. Liu, et al. Spotting opinion spammers using behavioral footprints. In KDD, 2013.
[7] N. Jindal and B. Liu. Opinion spam and analysis. In WSDM, 2008.
[8] C. Williams. Profile injection attack detection for securing collaborative recommender systems. In Tech. Rep., 2006.
[9] B. Mehta and W. Nejdl. Unsupervised strategies for shilling detection and robust collaborative filtering. In UMUAI, 2009.
[10] N. Jindal and B. Liu. Analyzing and detecting review spam. In ICDM, 2007.
[11] S. Xie, G. Wang, S. Lin, and P. S. Yu. Review spam detection via temporal pattern discovery. In KDD, 2012.
[12] S. Lam and J. Riedl. Shilling recommender systems for fun and profit. In WWW, 2004.
[13] X. Wu, Z. Feng, W. Fan, J. Gao, and Y. Yu. Detecting marionette microblog users for improved information credibility. In PKDD, 2013.
[14] H. Liu, Y. Zhang, H. Lin, J. Wu, Z. Wu, and X. Zhang. How many zombies around you? In ICDM, 2013.
[15] F. Benevenuto, T. Rodrigues, V. Almeida, et al. Detecting spammers and content promoters in online video social networks. In SIGIR, 2009.
[16] L. Becchetti, C. Castillo, D. Donato, et al. Using rank propagation and probabilistic counting for link-based spam detection. In WebKDD, 2006.
[17] G. Wu, D. Greene, B. Smyth, et al. Distortion as a validation criterion in the identification of suspicious reviews. In SMA, 2010.
[18] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving directions based on taxi trajectories. In GIS, 2010.
[19] G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu. Text classification without negative examples revisit. In TKDE, 2006.
[20] S. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In ICML, 2006.
[21] J. Lee and D. Zhu. Shilling attack detection—a new approach for a trustworthy recommender system. In JoC, 2010.
[22] Z. Wu, J. Wu, J. Cao, et al. HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation. In KDD, 2012.
[23] B. Mehta, T. Hofmann, and P. Fankhauser. Lies and propaganda: Detecting spam users in collaborative filtering. In IUI, 2007.
[24] B. Liu. Web data mining: Exploring hyperlinks, contents, and usage data. Springer, 2007.
[25] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: Social honeypots + machine learning. In SIGIR, 2010.