Spotting Fake Reviews via Collective Positive-Unlabeled Learning

Huayi Li*, Zhiyuan Chen*, Bing Liu*, Xiaokai Wei* and Jidong Shao†
*Department of Computer Science
University of Illinois at Chicago, IL, USA
†Dianping Inc., Shanghai, China
{lhymvp, czyuanacm}@gmail.com liub@cs.uic.edu weixiaokai@gmail.com jidong.shao@dianping.com

***Abstract*—Online reviews have become an increasingly important resource for decision making and product designing. But reviews systems are often targeted by opinion spamming. Although fake review detection has been studied by researchers for years using supervised learning, ground truth of large scale datasets is still unavailable and most of existing approaches of supervised learning are based on pseudo fake reviews rather than real fake reviews. Working with Dianping[1], the largest Chinese review hosting site, we present the first reported work on fake review detection in Chinese with filtered reviews from Dianping's fake review detection system. Dianping's algorithm has a very high precision, but the recall is hard to know. This means that all fake reviews detected by the system are almost certainly fake but the remaining reviews (unknown set) may not be all genuine. Since the unknown set may contain many fake reviews, it is more appropriate to treat it as an unlabeled set. This calls for the model of learning from positive and unlabeled examples (PU learning). By leveraging the intricate dependencies among reviews, users and IP addresses, we first propose a collective classification algorithm called Multi-typed Heterogeneous Collective Classification (MHCC) and then extend it to Collective Positive and Unlabeled learning (CPU). Our experiments are conducted on real-life reviews of 500 restaurants in Shanghai, China. Results show that our proposed models can markedly improve the F1 scores of strong baselines in both PU and non-PU learning settings. Since our models only use language independent features, they can be easily generalized to other languages.**

***Keywords*-Spam Detection, Collective PU Learning**

I. Introduction

Opinions in reviews are increasingly used by individuals and organizations for making purchase decisions, marketing and product design. Positive opinions often mean profits and fames for businesses and individuals, which, unfortunately, give strong incentives for imposters to post fake reviews to promote or to discredit some target products or services. Such individuals are called opinion spammers and their activities are called opinion spamming [4]. Among popular review hosting sites, Dianping is the largest host of Chinese reviews. It was founded in April 2003, in Shanghai, China and now has more than 100 million monthly active users, over 33 million reviews, and more than 10 million local businesses. To improve the quality of their reviews, Dianping developed a system to detect fake reviews. It has been shown that the precision of the system is very high, which means that when the system spots a fake review it is almost certainly a fake review. However, the true recall of their system is unknown as experienced spammers can still compose deceptive or fake reviews that are very hard to detect. This necessitates the model of learning from positive and unlabeled examples (PU learning). With Dianping's review dataset, we are able to deeply investigate the underlying mechanism of opinion spamming and perform a supervised learning on the binary classification task.

Inspired by the work from researchers who proposed graph-based models [1], [2], [9] and [12], we believe that exploiting the subtlety of the correlations between users, reviews and IP addresses would achieve better prediction results. Thus we first propose a collective classification algorithm *MHCC* (Multi-typed Heterogeneous Collective Classification) to identify fake reviews in our defined heterogeneous network over users, reviews and IP addresses. MHCC still considers unlabeled/unfiltered reviews as negative data and incorporates the text features of reviews and behavioral features of users and IP addresses into a relational classifier. MHCC is an iterative algorithm as data instances in MHCC have very rich relational features that are estimated by itself in the previous classification steps. Its strength is that it can detect suspicious users and IPs and propagate the probability from them back to reviews that are hard to classify by merely text features.

However, applying collective classification on the problem is still not enough as there are possibly many hidden fake reviews in the unlabeled set which can confuse the classifier. In addition, a small size of training data for collective classification will restrict its capability of exploiting relational features. To cope with the positive and unlabeled data, we extend MHCC to a Collective PU learning model (*CPU*). The CPU model treats the unlabeled data as negative only in the initialization stage. It then runs iteratively to mine both positive and negative instances from unlabeled set. Once an initial classifier is learned, it starts to assess the classification results and generate confident positive and negative instances based on which subsequent classifiers are trained. Our experiments show that our PU learning model outperforms supervised baselines significantly for both relational and non-relational classifiers. Even provided with a small percentage of training data, the proposed CPU model performs significantly better than the state-of-the-art models. This demonstrates the power

[1]http://www.dianping.com

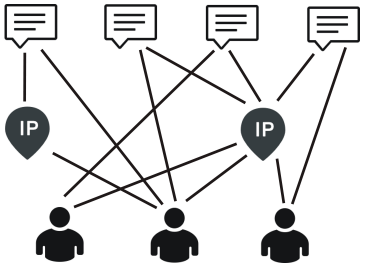

Fig. 1: Sample network of the users, reviews and IP addresses.

of PU learning as its goal is to find hidden positives from the unlabeled set in the absence of negative training data.

II. Related Work

PU learning: PU learning has been shown effective in many applications where only positive data is available. So far the work in [3] is the only one that deals with opinion spam in the PU learning setting. They proposed a PU learning framework called PU-LEA that iteratively removes positive data from unlabeled data. However, they assume a continual and gradual reduction of the negative instances over iterations which unfortunately is not always true. In our experiment, their algorithm identified much fewer positive examples from the unlabeled set.

Classification on heterogeneous networks: Researchers have extended traditional machine learning classification techniques to classifying nodes in networked data. Collective classification is one such technique. Lu et al. [7] used Iterative Classification Algorithm (ICA) in homogeneous networks and Kong et al. [5] extended it to the heterogeneous network by reducing the heterogeneous network to a homogeneous one through the use of meta-path [11]. Inspired by the advantage of collective classification and PU learning, our work combines them into a unified framework which we will discuss later.

III. Problem Definition

In this section, we first introduce some notations and concepts. Then, we formally define the collective positive and unlabeled learning problem in a heterogeneous network.

A heterogeneous network is defined as $\mathcal{G} = (V, T, E)$, where $V = \{v_1, v_2, ..., v_{|V|}\}$ is a set of nodes representing the entities of different types and E is the set of edges incident on V. $e_{i,j} \in E$ if v_i has a link to v_j. In our specific problem, we define our heterogeneous network as an undirect graph such that if $e_{i,j} \in E$ then $e_{j,i} \in E$. $T = \{t_1, t_2, ..., t_{|V|}\}$ is the set of corresponding entity types of nodes. Each t_i is an element of a finite set of types Γ, i.e., $t_i \in \Gamma$. For example, in our defined heterogeneous network, there are three types of nodes, users, reviews and IP addresses (or IPs for short). That is, $\Gamma = \{\text{User, Review, IP}\}$. The edges between nodes represent their dependency relationships. We did not consider restaurants as part of the network because their relations with other type of nodes are not very strong which we will discuss later. Figure 1 schematically shows three types of nodes and some edges between them.

Each node v_i is associated with a class label y_i which is an element of a set of states S_{t_i} that the node belongs to with respect to its entity type t_i. Thus we have $y_i \in S_{t_i}$. In our review dataset, only positive labels for some reviews are available thus we define the states of nodes in Table I. A review has three states, *Fake* (positive class), *Truthful* (negative class), and a special state called *Unlabeled*. A user has two states *Spammer* and *Non-spammer* and an IP also can be either *Suspicious* or *Organic*. It is not necessarily true that all reviews written by spammers or from suspicious IP addresses are deceptive because spammers have a mixed behavior and IPs can be shared by a large number of people. The states for users/IPs stand for the most probable state that those users/IPs are expected to be given their probabilistic estimates.

Node Type t_i	Node States S_{t_i}
Review	*Fake*(+), *Truthful*(-), *Unlabeled*(u)
User	*Spammer*(+), *Non-spammer*(-)
IP	*Suspicious*(+), *Organic*(-)

TABLE I: States of different entities.

Symbol	Definition
$\mathcal{G}$	heterogeneous network
V	set of nodes in the network
T	corresponding types for nodes V
E	edges incident on nodes
Γ	set of entity names
v_i	i-th node in the network
t_i	entity type of node v_i s.t. $t_i \in \Gamma$
S_{t_i}	states that node v_i can be in given its node type t_i
x_i	observed feature vector for node v_i
$\mathcal{X}$	observed feature matrix for all the nodes
y_i	class label for node v_i s.t. $y_i \in S_{t_i}$
$\mathcal{Y}$	class label assignments for all the nodes
A	set of indices for review nodes in the training set
D	set of indices for review nodes in the testing set

TABLE II: Important Notations

We use i as the index for node v_i, which also has a set of attached properties. Apart from its class label y_i, each node or data intance v_i contains the observed features x_i. So we define the feature space for all nodes as $\mathcal{X}$ and the class label space as $\mathcal{Y}$. As we are solving the problem using a classification approach, we split our dataset into two parts: training and testing. Let A be the set of indices of nodes in the training set and D be the set of those in the testing set. Our task is to predict the class labels of reviews $\{y_i \mid i \in D\ ,\ y_i \in \{+, -\}\}$ given the input of the heterogeneous network $\mathcal{G}$ with only some positive labels $\{y_i \mid i \in A\ ,\ y_i \in \{+, u\}\}$. In summary, we list the notations and definitions in Table II.

IV. Collective Classification Models on Heterogeneous Networks

In this section, we first introduce the classic Iterative Classification Algorithm (ICA) [10] and its applications. Then, we will describe our modified algorithm which is more appropriate for the collective classification problem on heterogeneous networks.

A. Collective Classification on reviews

A conventional classifier is a function f that maps the observed feature matrix $\mathcal{X}$ onto the class label space $\mathcal{Y}$. In our setting, unigrams and bigrams are features of reviews. We normalize those text features by converting them into TF-IDF values. Feature vectors are usually regarded as independent of each other. However, ICA breaks the independence assumption. It is commonly used as an approximate inference algorithm whose premise is very simple. Consider a node v_i whose class label $y_i \in \mathcal{Y}$ needs to be determined and suppose we have a neighborhood function $\mathcal{N}_i$ (Eqn.1) that returns the indices of its neighbors. $\mathcal{M}_i$ (Eqn. 2) is the class label vector of neighbors of node i which is derived from $\mathcal{N}_i$ and the estimated class labels matrix $\hat{\mathcal{Y}}$. ICA trains a local classifier f that takes in the observed features x_i of v_i as well as the estimated labels from its neighbors $\mathcal{M}_i$. Since many labels of nodes are unknown, this process has to be executed iteratively and in each iteration the label y_i of each node is assigned with the current best estimates from the local classifier f (Eqn. 3). The classifier can be any conventional classifier such as Logistic Regression, Support Vector Machines (SVM).

$$\mathcal{N}_i = \{\ j\ \mid\ e_{i,j} \in E\} \tag{1}$$

$$\mathcal{M}_i = \{\ \hat{y_j}\ \mid j \in \mathcal{N}_i\ ,\ \hat{y_j} \in \hat{\mathcal{Y}}\} \tag{2}$$

$$y_i = f(x_i, \mathcal{M}_i) \tag{3}$$

However, the ICA algorithm is not directly applicable in our context because our network is in fact a multipartite graph in which nodes of the same type are not directly connected. In our case, reviews are the type of nodes that we aim to classify, but reviews are not neighbors of each other. This calls for the need of a model that can establish connections between reviews to help the model take advantages of ICA. Grounded on the common sense, we found a strong intuition that reviews from the same user or IP address tend to have similar class labels. Because spammers are paid per review, they write as many fake reviews as possible to maximize their profits. Sloppy spammers may use the same user account to write multiple reviews while more experienced spammers own multiple accounts. In both cases, we assume spammers do not change user accounts or IP addresses very often. Then reviews sharing the same user or IP would have similar labels. But reviews of the same restaurant may not have direct impact on each other and that's why we exclude restaurants from the network. In order to encode the correlation between reviews via users and IPs, we reconstruct the neighborhood function as follows where R means that the type of node is review:

$$\mathcal{N}_i = \{\ j \mid \exists k\ \ e_{i,k} \in E\ ,\ e_{j,k} \in E\ ,\ t_i = t_j = R\} \tag{4}$$

If we simply treat the unlabeled set as negative data, we formulate the basic solution of the collective classification on reviews using ICA. As our contribution is not the ICA algorithm, we will not discuss it in this paper. Please see more details on [10]. In general, this idea is in fact similar to the HCC (Heterogeneous Collective Classification) model proposed in [5] when the length of meta-path is set to two. Longer path will not achieve better results as the correlation becomes weak.

Algorithm 1 MHCC Model for Supervised Learning

Input: Heterogenous network $\mathcal{G} = \{V, T, E\}$
Training dataset indices A, Testing dataset indices D
Feature matrix $\mathcal{X}^R,\ \mathcal{X}^U,\ \mathcal{X}^I$
Labels $\mathcal{Y}_A^R = \{y_i \mid i \in A\ , y_i \in \{+, -\}\},\ \mathcal{Y}^U,\ \mathcal{Y}^I$
Output: $\mathcal{Y}_D^R = \{y_i \mid i \in D\ ,\ y_i \in \{+, -\}\}$
superscripts are the entity types (R: review, U: user, I: IP)

1: $\mathcal{N}$, $\hat{\mathcal{Y}}$ = INITIALIZE($\mathcal{G}, A, D, \mathcal{X}^R, \mathcal{X}^U, \mathcal{X}^I, \mathcal{Y}_A^R, \mathcal{Y}^U, \mathcal{Y}^I$)
2: *//iterative steps*
3: **while** $\hat{\mathcal{Y}}$ not stabilized and maximal iterations have not elapsed **do**
4: $\mathcal{M} = \varnothing$ *//adjacent matrix of relational features*
5: $\hat{\mathcal{Y}}$ = PREDICT($\mathcal{N}, \mathcal{M}, A, D, \mathcal{X}^R, \mathcal{X}^U, \mathcal{X}^I, \mathcal{Y}_A^R, \mathcal{Y}^U, \mathcal{Y}^I$)
6: **end while**
7: Output $\mathcal{Y}_D = \{\hat{y_i} \mid i \in D\ ,\ \hat{y_i} \in \hat{\mathcal{Y}}\}$
8:
9: **procedure** INITIALIZE($\mathcal{G}, A, D, \mathcal{X}^R, \mathcal{X}^U, \mathcal{X}^I, \mathcal{Y}_A^R, \mathcal{Y}^U, \mathcal{Y}^I$)
10: Build $\mathcal{N}_i$ from $\mathcal{G}$ for $i \in \{1, 2, ..., |V|\}$ using Eq. 1, 4
11: $\mathcal{N} = \{\mathcal{N}_i \mid i \in \{1, 2, ..., |V|\}\}$
12: *//Construct features matrix $\mathcal{X}_A^R$, $\mathcal{X}_D^R$ from training data*
13: *//A and testing data D respectively*
14: $\mathcal{X}_A^R = \{x_i \mid i \in A\}$
15: $\mathcal{X}_D^R = \{x_i \mid i \in D\}$
16: Train a review classifier f^R from $\mathcal{X}_A^R$ and $\mathcal{Y}_A^R$
17: *//bootstrapping*
18: $\hat{\mathcal{Y}^R} = \mathcal{Y}_A^R$, $\hat{\mathcal{Y}^U} = \mathcal{Y}^U$, $\hat{\mathcal{Y}^I} = \mathcal{Y}^I$
19: **for** $i \in D$ **do**
20: *//estimate the label $\hat{y_i}$ for reviews in testing set*
21: $\hat{y_i} = f^R(x_i)$
22: $\hat{\mathcal{Y}^R} = \hat{\mathcal{Y}^R} \cup \hat{y_i}$
23: **end for**
24: **return** $\mathcal{N}$, $\hat{\mathcal{Y}}$
25: **end procedure**
26:
27: **procedure** PREDICT($\mathcal{N}, \mathcal{M}, A, D, \mathcal{X}^R, \mathcal{X}^U, \mathcal{X}^I, \mathcal{Y}_A^R, \mathcal{Y}^U, \mathcal{Y}^I$)
28: **for** $i \in \{1, 2, ..., |V|\}$ **do**
29: $\mathcal{M}_i = \{\ \hat{y_j}\ \mid j \in \mathcal{N}_i\ ,\ \hat{y_j} \in \hat{\mathcal{Y}^R} \cup \hat{\mathcal{Y}^U} \cup \hat{\mathcal{Y}^I}\}$
30: $\mathcal{M} = \mathcal{M} \cup \mathcal{M}_i$
31: **end for**
32: Train f^R from $\mathcal{X}_A^R$, $\mathcal{Y}_A^R$ and $\mathcal{M}$
33: Train f^U from $\mathcal{X}^U$, $\mathcal{Y}^U$ and $\mathcal{M}$
34: Train f^I from $\mathcal{X}^I$, $\mathcal{Y}^I$ and $\mathcal{M}$
35: *//update $\hat{\mathcal{Y}^R}$, $\hat{\mathcal{Y}^U}$, $\hat{\mathcal{Y}^I}$*
36: $\hat{\mathcal{Y}^R} = \mathcal{Y}_A^R$, $\hat{\mathcal{Y}^U} = \varnothing$, $\hat{\mathcal{Y}^I} = \varnothing$
37: **for** $i \in \{1, 2, ...|V|\}$ **do**
38: $k = t_i$ *// $t_i \in \{R, U, I\}$, t_i is the entity type*
39: $\hat{y_i} = f^k(x_i, \mathcal{M}_i)$
40: $\hat{\mathcal{Y}^k} = \hat{\mathcal{Y}^k} \cup \hat{y_i}$
41: **end for**
42: **return** $\hat{\mathcal{Y}}$
43: **end procedure**

B. Multi-typed Heterogeneous Collective Classification

This aforementioned approach only utilizes features from reviews and labels of reviews in the training set. The valuable information of users and IPs are neglected in the previous model. As Mukherjee et al. [8] pointed out that behavior features are strongly indicative clues of opinion spam, we want to construct behavior features for users and IP addresses to amend the collective learning with reviews. There are two challenges for the collective classification on the nodes of users and IPs. On one hand, for the supervised learning problem, there are no labels for users or IPs. As opposed to reviews

each of which is either fake or truthful, users and IPs appear in mixed behaviors making it extremely difficult and expensive to label and evaluate. As a consequence, the classification task becomes very difficult. On the other hand, behaviors calibrated from reviews of 500 restaurants are not strong enough to reveal the pattern of spamming. To tackle these two problems, we first obtain reviews of the users or IPs towards all restaurants in Shanghai in a two-year period to construct reliable behavioral features. Then we initialize the labels of users and IPs with the majority class label of their related review nodes.

We modify ICA to incorporate the observed features of user and IP nodes as proposed in [8]. For simplicity, we use the superscripts R, U, I as indicators of reviews, users and IPs respectively. For example, $\mathcal{X}^R$ is the feature matrix for reviews, and $\mathcal{X}^I$ means the behavioral features for IPs. Here we still treat unlabeled data as negative data and present the Multi-typed Heterogeneous Collective Classification (MHCC) in Algorithm 1. There are two main steps in the algorithm: initialization and iterative prediction. Both initialization and iterative prediction steps are different from HCC. Data instances in MHCC have richer relational features provided from the estimate of the classifier in the previous step. In the initialization step, we first compute the derived adjacency matrix in terms of Eqn. 1 and 4 (Line 10-11) and then train a classifier for review nodes based on their observed features (Line 14-16). The initial classifier gives a rough estimate of each review being in the positive (fake review) class while the estimated labels of user and IPs in this step are just derived from majority class label of their related reviews (Line 18-23). In the iterative prediction step, we construct a relational feature matrix $\mathcal{M}$ from the estimate labels of the neighboring nodes in Line 28-31 based on which we then train three different relational classifiers for reviews, users and IPs (Line 32-34). In Line 36-41, the newly trained relational classifiers would provide more accurate estimates of three types of nodes. This process happens iteratively (Line 3-6) as we always use the previous estimates to train current classifiers which provide the estimate for the classifiers in the next iteration.

C. Collective Positive-Unlabeled Learning Model

In this subsection, we show how to augment the collective classification model with the Positive-Unlabeled Learning framework to improve its performance. The MHCC model has two main drawbacks:

- There are potentially fake reviews hiding in the unlabel reviews that Dianping's algorithm did not capture. Since MHCC simply treats unlabeled examples as negative, if there are still a large number of fake reviews in the unlabeled set, MHCC is trained using data with wrong labels;
- The ad-hoc labels of users and IPs used in MHCC may not be very accurate as they are computed from labels of neighboring reviews. As the negative labels of neighboring reviews may actually be positive, the labels from users and IPs should be updated as more reviews are correctly classified.

Algorithm 2 CPU Model for PU learning

Input: Heterogenous network $\mathcal{G} = \{V, T, E\}$
Training dataset indices A, Testing dataset indices D
Feature matrix $\mathcal{X}^R$, $\mathcal{X}^U$, $\mathcal{X}^I$
Labels $\mathcal{Y}_A^R = \{y_i \mid i \in A\,, y_i \in \{+, u\}\}$, $\mathcal{Y}^U$, $\mathcal{Y}^I$
Output: $\mathcal{Y}_D^R = \{y_i \mid i \in D\ ,\ y_i \in \{+, -\}\}$

1: $\mathcal{N}$, $\hat{\mathcal{Y}}$ = INITIALIZE($\mathcal{G}, A, D, \mathcal{X}^R, \mathcal{X}^U, \mathcal{X}^I, \mathcal{Y}_A^R, \mathcal{Y}^U, \mathcal{Y}^I$)
2: **while** $\hat{\mathcal{Y}}$ not stabilized and maximal iterations have not elapsed **do**
3: $\mathcal{M} = \varnothing$ *//adjacent matrix of relational features*
4: $\hat{\mathcal{Y}}$ = PREDICT($\mathcal{N}, \mathcal{M}, A, D, \mathcal{X}^R, \mathcal{X}^U, \mathcal{X}^I, \mathcal{Y}_A^R, \mathcal{Y}^U, \mathcal{Y}^I$)
5: **end while**
6: Output $\mathcal{Y}_D = \{\hat{y}_i \mid i \in \mathcal{D}\ ,\ \hat{y}_i \in \hat{\mathcal{Y}}\}$
7:
8: **procedure** PREDICT($\mathcal{N}, \mathcal{M}, A, D, \mathcal{X}^R, \mathcal{X}^U, \mathcal{X}^I, \mathcal{Y}_A^R, \mathcal{Y}^U, \mathcal{Y}^I$)
9: **for** $i \in \{1, 2, ..., |V|\}$ **do**
10: $\mathcal{M}_i = \{\ \hat{y_j}\ \mid j \in \mathcal{N}_i\ ,\ \hat{y_j} \in \hat{\mathcal{Y}^R} \cup \hat{\mathcal{Y}^U} \cup \hat{\mathcal{Y}^I}\}$
11: $\mathcal{M} = \mathcal{M} \cup \mathcal{M}_i$
12: **end for**
13: Train f^R from $\mathcal{X}_A^R$, $\mathcal{Y}_A^R$ and $\mathcal{M}$
14: Train f^U from $\mathcal{X}^U$, $\mathcal{Y}^U$ and $\mathcal{M}$
15: Train f^I from $\mathcal{X}^I$, $\mathcal{Y}^I$ and $\mathcal{M}$
16: *//update* $\hat{\mathcal{Y}^R}$, $\hat{\mathcal{Y}^U}$, $\hat{\mathcal{Y}^I}$
17: $\hat{\mathcal{Y}^R} = \mathcal{Y}_A^R$, $\hat{\mathcal{Y}^U} = \varnothing$, $\hat{\mathcal{Y}^I} = \varnothing$
18: **for** $i \in \{1, 2, ..., |V|\}$ **do**
19: $k = t_i$ *//$t_i \in \{R, U, I\}$ t_i is the entity type*
20: $\hat{y_i} = f^k(x_i, \mathcal{M}_i)$
21: $\hat{\mathcal{Y}^k} = \hat{\mathcal{Y}^k} \cup \hat{y_i}$
22: **end for**
23: Compute the mean μ_k and σ_k of $\hat{\mathcal{Y}^k}$ of all three types
24: **for** $i \in \{1, 2, ..., |V|\}$ **do**
25: $k = t_i$ *// $t_i \in \{R, U, I\}$ t_i is the entity type*
26: **if** $\hat{y_i} >= \mu_k + \sigma_k$ **then** *// reliable positive data*
27: update $\mathcal{Y}^k$ with label y_i set to $+$
28: **if** $i \in D$ **then**
29: *//reliable positive in testing reviews for training*
30: move i from D to A
31: **end if**
32: **else if** $\hat{y_i} < \mu_k - \sigma_k$ **then** *// reliable negative data*
33: **if** $i \in D$ or y_i is not $+$ **then**
34: update $\mathcal{Y}^k$ with label y_i set to $-$
35: **end if**
36: **if** $i \in D$ **then**
37: *//reliable negative in testing reviews for training*
38: move i from D to A
39: **end if**
40: **end if**
41: **end for**
42: **return** $\hat{\mathcal{Y}}$
43: **end procedure**

To address these two issues, we change the iterative steps of MHCC to allow the labels of training to be modified by the model dynamically. New labels are updated with respect to the confident positives and negatives from all entities types. We call the new model Collective Postive-Unlabeled learning (CPU) model which is illustrated in Algorithm 2. The initialization step is the same as Algorithm 1, but there are two major differences in the iterative steps.

The first difference of MHCC and CPU is that CPU allows initial labels to be violated if current probability estimate strongly indicates the opposite prediction (Line 27 and 34). This is especially useful for mining fake reviews that are identified by the collective classifier but passed Dianping's filter. During each iteration, the model produces new estimates of nodes for all three types. For each type of nodes, we assume the probability is a normal distribution such that we can identify a set of reliable positive and reliable negative data instances (Line 23-41). Once a reliable positive is identified, we set its label from unlabel to positive. However, relying on

the fact that Dianping's filter has a very high precision, we only update the label of a reliable negative if its label was previously unlabeled as shown in Line 34.

The second difference is that CPU also uses testing data for training if their estimates are reliable (Line 28-31 and 31-39). Although we have no prior knowledge about labels of testing data, we rely on that the highly confident positive and negative examples are correct. It is impossible for non-relational classifier because testing data is now interacting with training data in the network, and it is also not suitable for conventional relational classifier since labels of testing dataset cannot be used. In many applications, labeled data is much smaller than the unlabeled set. The benefit of CPU model allows itself to learn data instances whose class labels were uncovered in previous classification steps. As more positive and negative data from unlabeled dataset are discovered, more training data can be used for training which is especially useful for the relational classifier CPU because the more training data there is in the relational feature matrix, the more predicative power the model has.

V. Experiment

We now evaluate our models and compare with several baselines using real-life restaurant reviews from Dianping.

A. Datasets and Preprocessing

	Fake reviews	Unlabeled reviews	Total
# of reviews	3523	6242	9765
# of unique users	3310	5894	9067
# of unique IPs	1314	4564	5535
# of reviews per user	1.064	1.059	1.077
# of reviews per IP	2.681	1.368	1.764
Avg # of words	53.17	63.21	59.59

TABLE III: Statistics of the 500 restaurants in Shanghai

Our experiment dataset consists of filtered (fake) reviews and unfiltered (unlabeled) reviews[2] from 500 restaurants in Shanghai, China. We use the most recent reviews of those restaurants between November 1st, 2011 and November 28th, 2013. Table III shows the statistics of our dataset. It is interesting to see that among the fake reviews, the number of reviews per IP is almost twice as many as those of unlabeled ones, which indicates that IPs of spammers are more active than IPs of organic users. So it also explains that by using network efforts why the relational classifier can improve the traditional classifier. Another interesting finding is that the length of fake reviews identified by Dianping's algorithm is on average shorter than unlabeled reviews. It might be due to the fact that non-spammers compose truthful reviews based on their genuine experiences while in contrast, it is harder for spammers to make up long stories without real experiences. Compared with English reviews, reviews in Chinese are much shorter. Reviews in our dataset contain 85.5 words on average while reviews in Yelp's data challenge[3] have an average length of 130.6 words.

[2]http://liu.cs.uic.edu/download/dianping

[3]http://www.yelp.com/dataset_challenge

Training and Testing: Our dataset contains two classes, positive and unlabeled data. For non-PU learning models, we treat unlabeled reviews as negative data in both training and testing. However, as for our proposed Collective Positive-Unlabeled learning model and other PU learning baseline models, we treat unlabeled data as unlabeled in training but in testing we assume unlabeled instances are negative because we have no idea which unlabeled reviews are actually fake. We choose this way as it is fair for all models as they are evaluated in the same way. Our results were obtained through 10-fold cross-validation with data instances randomly shuffled.

Evaluation Metrics The effectiveness of the proposed model was evaluated by means of the standard F1-score(F1), Precision(P), Recall(R) and Accuracy(A).

B. Compared Methods

We now list the baselines from related works and compare them with our proposed models.

Logistic Regression (LR): We use logistic regression because it produces a probability estimate of each review for being in the positive class (in our case the fake review class) efficiently. For the fairness of comparison, we append the user and IP features to the reviews. LR also serves as the base learner for other relational classifiers that will be discussed later including our proposed model.

PU-LEArning (PU-LEA): To our best knowledge, this is so far the only PU Learning algorithm that has been employed for spam detection. Hernández et al. [3] proposed PU-LEA algorithm to iteratively remove positive data instances from unlabeled ones. Their algorithm treats the unlabeled reviews as the negative data. The trained classifier tries to uncover hidden positives in the unlabel set gradually which in turn helps train a new classifier with updated labels. Their work becomes our compared baseline in the PU learning setting without network relations explored.

Spy-EM and Spy-SVM: Liu et al. [6] built a series of classifiers for Learning from Positive and Unlabeled data (LPU). We compared their models with our Collective PU learning model to show that incorporating the relations between users, IPs and reviews can improve the performance.

Heterogeneous Collective Classification (HCC) : Kong et al. [5] proposed a heterogeneous classifier replying on meta-path [11]. By considering different linkage paths in the heterogeneous network, their model captures dependencies of objects of various types.

C. Model Comparison

We now evaluate the effectiveness of our proposed models on the collective classification task. 10-fold cross validation is performed for each model. F1, precision, recall and accuracy are reported in Figure 2. First of all, all three collective classification models (HCC, MHCC, CPU) that explicitly exploit label dependencies from various aspects can achieve much better classification results (F1-scores, precision and accuracy) than non-relational classifiers which classify each instance independently. This finding supports the intuition that

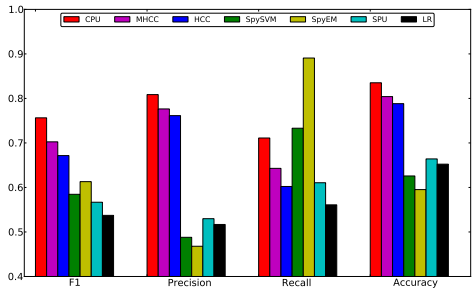

Fig. 2: Evaluation of different models based on 10-fold cross validation. Note: although Spy-EM has the highest recall, it over-predicts the positive class and hence the lowest precision.

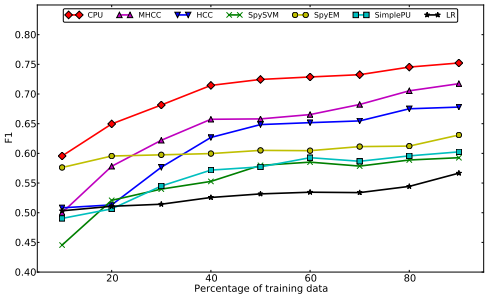

Fig. 3: F1-score of all models with various training size

collective classification classifiers are superior as they take into account the labels of neighboring nodes. If a review is fake, reviews from the same user or IP tend to be suspicious.

Secondly, non-relational PU learning models (PU-LEA, Spy-EM, Spy-SVM) have higher recall than traditional supervised classifier LR despite some loss in precision. This is due to the fact that Dianping's unlabeled reviews contain hidden positives which could provide a purer positive and negative instances for training. Last but not least, our proposed models, MHCC and CPU outperform the strong baseline HCC with higher precision and recall. It is not surprising that MHCC beats HCC as it incorporates more complex correlations from all types of nodes in the heterogeneous network. Again, CPU further improves MHCC in that more positive and negative users, reviews and IPs are correctly identified which in turn can train more accurate classifiers. Fake reviews that are not filtered by Dianping's system could provide additional evidence of their associated users and IPs. Likewise suspicious users and IPs can disclose more hidden fake reviews. Exploiting the complex structures from the network, MHCC and CPU effectively identify many false negatives that LR and HCC cannot find. The use of labels of neighboring objects can largely improve the precision as opposed to other PU learning models and the base learner LR. F1-scores of 10-fold cross-validation show that CPU significantly improves HCC and MHCC based on the on paired t-test ($p < 0.0001$).

Generally speaking, the more training data, the better performance collective classification classifiers can get. In order to compare the relational classifiers (HCC, MHCC and CPU) with the other non-relational classifiers given different sizes of training data. We show the F1-scores of different models varying the size of training data from 10% to 90% in Figure 3. It can be read from the chart that when training data is less than 20% most local classifiers (both LR and PU learning ones) do a better job than most relational ones. But when given more than 40% training data, relational classifiers are significantly improved. It is worth noting that our proposed Collective PU learning still achieve a relatively good results even if only a small amount of training data is provided. This sheds light on the effectiveness of PU learning in the collective classification setting as our CPU model most effectively uncovers hidden positives from unlabeled data and iteratively enhances itself.

VI. Conclusion

This paper studied the problem of fake review detection in the Collective PU learning framework. We first proposed a supervised learning algorithm MHCC for the heterogeneous network of reviews, users and IPs and then extended it to CPU model which is more appropriate for PU learning problem because the labels of reviews have very high precision but unknown recall. With the labeled data provided by the review hosting website Dianping, we conducted several experiments to show that combining collective classification and PU learning, our proposed CPU model has major advantages over existing state-of-the-art baseline algorithms. It not only outperforms them, but also more importantly, detects a large number of potential fake reviews hidden in the unlabeled set, which shows the power of PU learning in solving the problem. Moreover, because our models only use language independent features, they can be generalized to any other languages.

References

[1] L. Akoglu, R. Chandy, and C. Faloutsos, "Opinion fraud detection in online reviews by network effects," in *ICWSM*, 2013.

[2] G. Fei, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh, "Exploiting burstiness in reviews for review spammer detection," in *ICWSM*, 2013.

[3] D. Hernández, R. Guzmán, M. Móntes ỹ Gomez, and P. Rosso, "Using pu-learning to detect deceptive opinion spam," in *Proceedings of the 4th Workshop on Computational Approaches to S ubjectivity, Sentiment and Social Media Analysis*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 38–45.

[4] N. Jindal and B. Liu, "Opinion spam and analysis," in *WSDM*, 2008, pp. 219–230.

[5] X. Kong, P. S. Yu, Y. Ding, and D. J. Wild, "Meta path-based collective classification in heterogeneous information networks," in *CIKM*, 2012, pp. 1567–1571.

[6] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *ICDM*, 2003, pp. 179–188.

[7] Q. Lu and L. Getoor, "Link-based classification." in *ICML*, 2003, pp. 496–503.

[8] A. Mukherjee, V. Venkataraman, B. Liu, and N. S. Glance, "What yelp fake review filter might be doing?" in *ICWSM*, 2013.

[9] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *WWW*, 2007, pp. 201–210.

[10] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[11] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, vol. 4, no. 11, pp. 992–1003, 2011.

[12] G. Wang, S. Xie, B. Liu, and P. S. Yu, "Review graph based online store review spammer detection," in *ICDM*, 2011, pp. 1242–1247.