Positive Unlabeled Learning for Deceptive Reviews Detection

Yafeng Ren **Donghong Ji** **Hongbin Zhang**
Computer School
Wuhan University
Wuhan 430072, China
{renyafeng,dhji,zhanghongbin}@whu.edu.cn

Abstract

Deceptive reviews detection has attracted significant attention from both business and research communities. However, due to the difficulty of human labeling needed for supervised learning, the problem remains to be highly challenging. This paper proposed a novel angle to the problem by modeling PU (positive unlabeled) learning. A semi-supervised model, called mixing population and individual property PU learning (MPIPUL), is proposed. Firstly, some reliable negative examples are identified from the unlabeled dataset. Secondly, some representative positive examples and negative examples are generated based on LDA (Latent Dirichlet Allocation). Thirdly, for the remaining unlabeled examples (we call them *spy* examples), which can not be explicitly identified as positive and negative, two similarity weights are assigned, by which the probability of a *spy* example belonging to the positive class and the negative class are displayed. Finally, *spy* examples and their similarity weights are incorporated into SVM (Support Vector Machine) to build an accurate classifier. Experiments on gold-standard dataset demonstrate the effectiveness of MPIPUL which outperforms the state-of-the-art baselines.

1 Introduction

The Web has dramatically changed the way people express themselves and interact with others, people frequently write reviews on e-commerce sites, forums and blogs to achieve these purposes. For NLP (Natural Language Processing), these user-generated contents are of great value in that they contain abundant information related to people's opinions on certain topics. Currently, online reviews on products and services are used extensively by consumers and businesses to conduct decisive purchase, product design and marketing strategies. Hence, sentiment analysis and opinion mining based on product reviews have become a popular topic of NLP (Pang and Lee, 2008; Liu, 2012). However, since reviews information can guide people's purchase behavior, positive reviews can result in huge economic benefits and fame for organizations or individuals. This leaves room for promoting the generation of review spams. Through observations and studies of the predecessors (Jindal and Liu, 2008; Ott et al., 2011), review spams are divided into the following two classes:

- **Deceptive Reviews:** Those deliberately mislead readers by giving undeserving positive reviews to some target objects in order to promote the objects, or by giving unjust negative reviews to some target objects in order to damage their reputation.
- **Disruptive Reviews:** Those are non-reviews, which mainly include advertisements and other irrelevant reviews containing no opinion.

Disruptive reviews pose little threat to people, because human can easily identify and ignore them. In this paper, we focus on the more challenging ones: deceptive reviews. Generally, deceptive reviews detection is deemed to be a classification problem (Ott et al., 2011; Li et al., 2011; Feng et al., 2012). Based on the positive and negative examples annotated by people, supervised learning is utilized to build a classifier, and then an unlabeled review can be predicted as deceptive review or truthful one. But the work from Ott et al. (2011) shows that human cannot identify deceptive reviews from their prior knowledge, which indicates that human-annotated review datasets must

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 488–498,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

include some mislabeled examples. These examples will disturb the generation ability of the classifiers. So simple supervised learning is regarded as unsuitable for this task.

It is difficult to come by human labeling needed for supervised learning and evaluation, we cannot obtain the datasets containing deceptive reviews. However, we can get some truthful reviews with high confidence by heuristic rules and prior knowledge. Meanwhile, a lot of unlabeled reviews are available. The problem thus is this: based on some truthful reviews and a lot of unlabeled reviews, can we build an accurate classifier to identify deceptive reviews.

PU (positive unlabeled) learning can be utilized to deal with the above situation (Liu et al., 2002; Liu et al., 2003). Different from traditional supervised learning, PU learning can still build an accurate classifier even without the negative training examples. Several PU learning techniques have been applied successfully in document classification with promising results (Zhang, 2005; Elkan and Noto, 2008; Li et al., 2009; Xiao et al., 2011), while they have yet to be applied in detecting deceptive reviews. Here, we will study how to design PU learning to detect deceptive reviews.

An important challenge is how to deal with *spy* examples (easily mislabeled) of unlabeled reviews, which is not easily handled by the previous PU learning techniques. In this paper, we propose a novel approach, mixing population and individual property PU learning (MPIPUL), by assigning similarity weights and incorporating weights into SVM learning phase. This paper makes the following contributions:

- For the first time, PU learning is defined in the environment of identifying deceptive reviews.
- A novel PU learning is proposed based on LDA and SVM.
- Experimental results demonstrate that our proposed method outperforms the current baselines.

2 Related Work

2.1 Deceptive Reviews Detection

Spam has historically been investigated in the contexts of e-mail (Drucker et al., 1999; Gyongyi et al., 2004) and the Web (Ntoulas et al., 2006). In recent years, researchers have started to look at deceptive reviews.

Jindal and Liu (2008) found that opinion spam was widespread and different from e-mail and Web spam in essence (Jindal and Liu, 2008). They trained models using product review data, by defining features to distinguish duplicate opinion and non-duplicate based on the review text, reviewers and product information. Wu et al. (2010) proposed an alternative strategy of popularity rankings (Wu et al., 2010).

Ott et al. (2011) developed the first dataset containing gold-standard deceptive reviews by crowdsourcing (Ott et al., 2011), and presented three supervised learning methods to detect deceptive reviews by integrating knowledge from psycholinguistics and computational linguistics. This gold-standard dataset will be used in the paper. Li et al. (2011) manually built a review dataset from their crawled reviews (Li et al., 2011), and exploited semi-supervised co-training algorithm to identify deceptive reviews.

Feng et al. (2012) verified the connection between the deceptive reviews and the abnormal distributions (Feng et al., 2012a). Later, they (Feng et al., 2012b) demonstrated that features driven from CFG (Context Free Grammar) parsing trees consistently improve the detection performance.

Mukherjee et al. (2012) proposed detecting group spammers (a group of reviewers who work collaboratively to write deceptive reviews) in product reviews (Mukherjee et al., 2012). The proposed method first used frequent itemset mining to find a set of candidate groups. Then GSRank was presented which can consider relationships among groups, individual reviewers and products they reviewed to detect spammer groups. Later, they also proposed exploiting observed reviewing behaviors to detect opinion spammers in an unsupervised Bayesian inference framework (Mukherjee et al., 2013).

Ren et al. (2014) assumed that there must be some difference on language structure and sentiment polarity between deceptive reviews and truthful ones (Ren et al., 2014a), then they defined the features related to the review text and used genetic algorithm for feature selection, finally they combined two unsupervised clustering algorithm to identify deceptive reviews. Later, they (Ren et al., 2014b) present a new approach, from the viewpoint of correcting the mislabeled

examples, to find deceptive reviews. Firstly, they partition a dataset into several subsets.Then they construct a classifier set for each subset and select the best one to evaluate the whole dataset. Meanwhile, error variables are defined to compute the probability that the examples have been mislabeled. Finally, the mislabeled examples are corrected based on two threshold schemes, majority and non-objection.

Unlike previous studies, PU learning is implemented to identify deceptive reviews.

2.2 Positive Unlabeled Learning

According to the use of the unlabeled data, PU learning can be divided into two classes.

One family of methods built the final classifier by using positive examples dataset and some examples of the unlabeled dataset (Liu et al., 2002; Liu et al., 2003). The basic idea is to find a set of reliable negative examples from the unlabeled data firstly, and then to learn a classifier using EM (Expectation Maximization) or SVM. The performance is limited for neglecting the rest examples of unlabeled dataset.

Another family of methods learned the final classifier by using positive examples dataset and all examples of the unlabeled dataset. Li et al. (Li et al., 2009) studied PU learning in the data stream environment, they proposed a PU learning LELC (PU Learning by Extracting Likely positive and negative micro-Clusters) for document classification, they assume that the examples close together shared the same labels. Xiao et al. (Xiao et al., 2011) proposed a method, called SPUL (similarity-based PU learning), the local similarity-based and global similarity-based mechanisms are proposed to generate the similarity weights for the easily mislabeled examples, respectively. Experimental results show global SPUL generally performs better than local SPUL.

In this paper, a novel PU learning (MPIPUL) is proposed to identify deceptive reviews.

3 Preliminary

Before we introduce the proposed method, we briefly review SVM, which has proven to be an effective classification algorithm (Vapnik, 1998).

Let $T = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(|T|)}, y^{(|T|)})\}$ be a training set, where $x^{(i)} \in R^d$ and $y^{(i)} \in \{+1, -1\}$. SVM aims to seek an optimal separating hyperplane $w^T x^{(i)} + b = 0$, the hyperplane can be obtained by solving the following optimization problem:

$$\begin{aligned} min \quad & F(w, b, \epsilon_i) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{|T|}\epsilon_i \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \epsilon_i, i = 1, \ldots, |T| \\ & \epsilon_i \geq 0, i = 1, \ldots, |T| \end{aligned} \tag{1}$$

where w^T represents the transpose of w, C is a parameter to balance the classification errors and ϵ_i are variables to relax the margin constraints. The optimal classifier can be achieved by using the Lagrange function. For a test example x, if $w^T x + b < 0$, it is classified into the negative class; otherwise, it is positive.

In the following, SVM is extended to incorporate the *spy* examples and their weights, such that the *spy* examples can contribute differently to the classifier construction.

4 The Proposed Method

In this section, we will introduce the proposed approach in details. In our PU learning (MPIPUL), truthful reviews are named positive examples, and deceptive reviews are called negative examples. P is defined as a set which contains all positive examples. U is a set for all unlabeled examples. PU learning aims at building a classifier using P and U. MPIPUL adopts the following four steps:

- Step 1: Extract the reliable negative examples;
- Step 2: Compute the representative positive and negative examples;
- Step 3: Generate the similarity weights for the *spy* examples;
- Step 4: Build the final SVM classifier;

4.1 Extracting Reliable Negative Examples

Considering only positive and unlabeled examples are available in PU learning, some negative examples need to be extracted firstly. These examples will influence the performance of the following three steps. So high-quality negative examples must be guaranteed. Previous works solved the problem with the Spy technique (Liu et al., 2002) or the Rocchio technique (Liu et al., 2003), we integrate them in order to get reliable negative examples. Let subsets NS_1 and NS_2 contain the

corresponding reliable negative examples extracted by the two techniques, respectively. Examples are considered to be a reliable negative only if both techniques agree that they are negative. That is, $NS = NS_1 \cap NS_2$, where NS contains the reliable negative examples.

After reliable negative examples are extracted, there are still some unlabeled examples (we call *spy* examples) in set U, let subset $US = U - NS$, which stores all the *spy* examples. It is crucial to determine how to deal with these *spy* examples.

4.2 Computing Representative Positive and Negative Examples

Generally, a classifier can be constructed to predict deceptive reviews based on the positive examples set P and the reliable negative examples set NS. But the classifier is not accurate enough for lacking of making full use of unlabeled dataset U. In order to utilize *spy* examples in subset US, some representative positive and negative examples are calculated firstly. Since the examples have different styles in sentiment polarity and topic distribution, for every class, computing one representative example is not suitable. For the positive class or the negative class, to ensure there is a big difference between the different representative examples. This paper proposes clustering reliable negative examples into several groups based on LDA (Latent Dirichlet Allocation) topic model and K-means, and then multiple representative examples can be obtained.

LDA topic model is known as a parametric Bayesian clustering model (Blei et al., 2003), and assumes that each document can be represented as the distribution of several topics, each document is associated with common topics. LDA can well capture the relationship between internal documents.

In our experiments based on LDA model, we can get the topic distribution for the reliable negative examples, then some reliable negative examples which are similar in topic distribution will be clustered into a group by K-means. Finally, these reliable negative examples can be clustered into n micro-clusters $(NS_1, NS_2, \ldots, NS_n)$. Here,

$$n = 30 * |NS|/(|US| + |NS|) \quad (2)$$

Here, according to the suggestion of previous work (Xiao et al., 2011), we examine the impact of the different parameter (from 10 to 60) on overall performance, and select the best value 30.

Based on the modified Rocchio formula (Buckley et al., 1999), n representative positive examples (p_k) and n negative ones (n_k) can be obtained using the following formula:

$$p_k = \alpha \frac{1}{|P|} \sum_{i=1}^{|P|} \frac{x^{(i)}}{\| x^{(i)} \|} - \beta \frac{1}{|NS_k|} \sum_{i=1}^{|NS_k|} \frac{x^{(i)}}{\| x^{(i)} \|}$$

$$n_k = \alpha \frac{1}{|NS_k|} \sum_{i=1}^{|NS_k|} \frac{x^{(i)}}{\| x^{(i)} \|} - \beta \frac{1}{|P|} \sum_{i=1}^{|P|} \frac{x^{(i)}}{\| x^{(i)} \|}$$

$$k = 1, \ldots, n \quad (3)$$

According to previous works (Buckley et al., 1994), where the value of α and β are set to 16 and 4 respectively. The research from Buckley et al. demonstrate that this combination emphasizes occurrences in the relevant documents as opposed to non-relevant documents.

4.3 Generating Similarity Weights

For a *spy* example x, since we do not know which class it should belong to, enforcing x to the positive class or the negative class will lead to some mislabeled examples, which disturbs the performance of final classifier. We represent a *spy* example x using the following probability model:

$$\{x, (p^+(x), p^-(x))\}, \quad p^+(x) + p^-(x) = 1 \quad (4)$$

Where $p^+(x)$ and $p^-(x)$ are similarity weights which represent the probability of x belonging to the positive class and the negative class, respectively. For example, $\{x, (1, 0)\}$ means that x is positive, while $\{x, (0, 1)\}$ indicates that x is identified to be negative. For $\{x, (p^+(x), p^-(x))\}$, where $0 < p^+(x) < 1$ and $0 < p^-(x) < 1$, it implies that the probability of x belonging to the positive class and the negative class are both considered.

In this section, similarity weights are decided by mixing global information (population property) and local information (individual property). Then all *spy* examples and their similarity weights are incorporated into a SVM-based learning model.

4.3.1 Population Property

Population property means that the examples in each micro-cluster share the similarity in sentiment polarity and topic distribution, and they belong to the same category with a high possibility. In our framework, in order to compare with the representative examples, all *spy* examples are firstly clustered into n micro-clusters

$(US_1, US_2, \ldots, US_n)$ based on LDA and K-means. Then, for every *spy* example x in one micro-cluster US_i, we tags with temporary label by finding its most similar representative example. Finally, we can get the similarity weights for a *spy* example x in micro-cluster US_i, their probability pertaining to the positive class and negative class can be represented by the following formula:

$$\begin{aligned} p\_pop(x) &= \frac{|positive|}{|US_i|} \\ n\_pop(x) &= \frac{|negative|}{|US_i|} \end{aligned} \tag{5}$$

where $|US_i|$ represents the number of all examples in micro-cluster US_i, $|positive|$ means the number of the examples which is called temporary positive in US_i, and $|negative|$ means the number of the examples which is called temporary negative in US_i.

For example, Figure 1 shows the part (C1, C2, C3, C4) of the clustering results for the *spy* examples based on LDA and K-means, the examples x in C4 are assigned with weights $p\_pop(x) = \frac{4}{9}, n\_pop(x) = \frac{5}{9}$, the examples x in C1 are assigned with weights $p\_pop(x) = 1, n\_pop(x) = 0$.

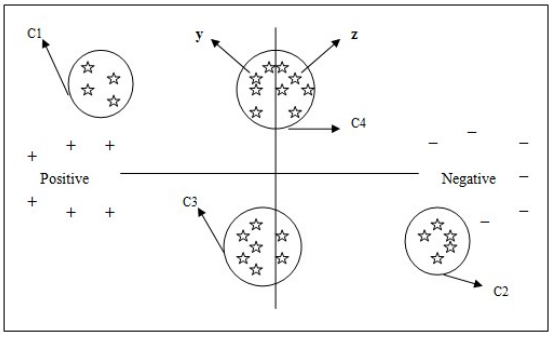

Figure 1: Illustration of population property

The advantage of population property lies in the fact that it considers the similar relationship between the examples, from which the same micro-cluster are assigned the same similarity weight. However, it cannot distinguish the difference of examples in one micro-cluster. In fact, the similarity weights of examples from the same micro-cluster can be different, since they are located physically different. For example, for the *spy* example y and z in micro-cluster C4, it is apparently unreasonable that we assign the same similarity weights to them. So we should join the local information (individual property) when we are computing the similarity weights for a *spy* example.

4.3.2 Individual Property

Individual property is taken into account to measure the relationship between every *spy* example and all representative ones. Specifically, for example x, we firstly compute its similarity to each of the representative examples, and then the probability of the example x belonging to the positive class and negative class can be calculated using the following formula:

$$\begin{aligned} p\_ind(x) &= \frac{\sum_{k=1}^{n} sim(x, p_k)}{\sum_{k=1}^{n}(sim(x, p_k) + sim(x, n_k))} \\ n\_ind(x) &= \frac{\sum_{k=1}^{n} sim(x, n_k)}{\sum_{k=1}^{n}(sim(x, p_k) + sim(x, n_k))} \end{aligned} \tag{6}$$

In the above formula,

$$sim(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

4.3.3 Similarity Weights

A scheme mixing population and individual property is designed to generate the similarity weights of *spy* examples. Specifically, for *spy* example x, their similarity weights can be obtained by the following formula:

$$\begin{aligned} p^+(x) &= \lambda \cdot p\_pop(x) + (1 - \lambda) \cdot p\_ind(x) \\ p^-(x) &= \lambda \cdot n\_pop(x) + (1 - \lambda) \cdot n\_ind(x) \end{aligned} \tag{7}$$

Where λ is a parameter to balance the information from population property and individual property. In the remaining section, we will examine the impact of the parameter λ on overall performance. Meanwhile, it can be easily proved that $p^+(x) + p^-(x) = 1$.

4.4 Constructing SVM Classifier

After performing the third step, each *spy* example x is assigned two similarity weights: $p^+(x)$ and $p^-(x)$. In this section, we will extend the formulation of SVM by incorporating the examples in positive set P, reliable negative set NS, *spy* examples set US and their similarity weights into a SVM-based learning model.

4.4.1 Primal Problem

Since the similarity weights $p^+(x)$ and $p^-(x)$ indicate the probability for a *spy* example x belonging to the positive class and the negative class, respectively. The optimization formula (1) can be

rewritten as the following optimization problem:

$$
\begin{aligned}
min \quad & F(w,b,\epsilon) = \frac{1}{2}||w||^2 + C_1 \sum_{i=1}^{|P|} \epsilon_i + C_2 \cdot \\
& \sum_{j=1}^{|US|} p^+(x^{(j)})\epsilon_j + C_3 \sum_{m=1}^{|US|} p^-(x^{(m)})\epsilon_m \\
& + C_4 \sum_{n=1}^{|NS|} \epsilon_n \\
s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \epsilon_i, \quad x^{(i)} \in P \\
& y^{(j)}(w^T x^{(j)} + b) \geq 1 - \epsilon_j, \quad x^{(j)} \in US \\
& y^{(m)}(w^T x^{(m)} + b) \geq 1 - \epsilon_m, \quad x^{(m)} \in US \\
& y^{(n)}(w^T x^{(n)} + b) \geq 1 - \epsilon_n, \quad x^{(n)} \in NS \\
& \epsilon_i \geq 0, \quad \epsilon_j \geq 0, \quad \epsilon_m \geq 0, \quad \epsilon_n \geq 0
\end{aligned} \tag{8}
$$

Where C_1, C_2, C_3 and C_4 are penalty factors controlling the tradeoff between the hyperplane margin and the errors, $\epsilon_i, \epsilon_j, \epsilon_m$ and ϵ_n are the error terms. $p^+(x^{(j)})\epsilon_j$ and $p^-(x^{(m)})\epsilon_m$ can be considered as errors with different weights. Note that, a bigger value of $p^+(x^{(j)})$ can increase the effect of parameter ϵ_j, so that the corresponding example $x^{(j)}$ becomes more significant towards the positive class. In the following, we will find the dual form to address the above optimization problem.

4.4.2 Dual Problem

Assume α_i and α_j are Lagrange multipliers. To simplify the presentation, we redefine some notations as follows:

$$
C_i^+ = \begin{cases} C_1, & x^{(i)} \in P \\ C_2 p^+(x^{(j)}), & x^{(j)} \in US \end{cases}
$$

$$
C_j^- = \begin{cases} C_3 p^-(x^{(m)}), & x^{(m)} \in US \\ C_4, & x^{(n)} \in NS \end{cases}
$$

Based on the above definitions, we let $T^+ = P \cup US$, $T^- = US \cup NS$ and $T^* = T^+ \cup T^-$. The Wolfe dual of primal formulation can be obtained as follows (Appendix A for the calculation process):

$$
\begin{aligned}
max \quad & W(\alpha) = \sum_{i=1}^{|T^*|} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{|T^*|} \alpha_i \alpha_j y^{(i)} \cdot \\
& y^{(j)} < x^{(i)}, x^{(j)} > \\
s.t. \quad & C_i^+ \geq \alpha_i \geq 0, \quad x^{(i)} \in T^+ \\
& C_j^- \geq \alpha_j \geq 0, \quad x^{(j)} \in T^- \\
& \sum_{i=1}^{|T^+|} \alpha_i - \sum_{j=1}^{|T^-|} \alpha_j = 0
\end{aligned} \tag{9}
$$

where $< x^{(i)}, x^{(j)} >$ is the inner product of $x^{(i)}$ and $x^{(j)}$. In order to get the better performance, we can replace them by using kernel function $\phi(x^{(i)})$ and $\phi(x^{(j)})$, respectively. The kernel track can convert the input space into a high-dimension feature space. It can solve the uneven distribution of dataset and complex problem from heterogeneous data sources, which allows data to get a better expression in the new space (Lanckriet et al., 2004; Lee et al., 2007).

After solving the above problem, w can be obtained, then b can also be obtained by using KKT (Karush-Kuhn-Tucker) conditions. For a test example x, if $w^T x + b > 0$, it belongs to the positive class. Otherwise, it is negative.

5 Experiments

We aim to evaluate whether our proposed PU learning can identify deceptive reviews properly. We firstly describe the gold-standard dataset, and then introduce the way to generate the positive examples P and unlabeled examples U. Finally we present human performance in gold-standard dataset.

5.1 Datasets

There is very little progress in detection of deceptive reviews, one reason is the lack of standard dataset for algorithm evaluation. The gold-standard dataset is created based on crowdsourcing platform (Ott et al., 2011), which is also adopted as the experimental dataset in this paper.

5.1.1 Deceptive Reviews

Crowdsourcing services can carry out massive data collection and annotation; it defines the task in the network platform, and paid for online anonymous workers to complete the task.

Humans cannot be precisely distinguish deceptive ones from existing reviews, but they can create deceptive reviews as one part of the dataset. Ott et al. (2011) accomplish this work by AMT (Amazon Mechanical Turk). They set 400 tasks for 20 hotels, in which each hotel gets 20 tasks. Specific task is: If you are a hotel market department employee, for each positive review you wrote for the benefit for hotel development, you may get one dollar. They collect 400 deceptive reviews.

5.1.2 Truthful Reviews

For the collection of truthful reviews, they get 6977 reviews from TripAdvisor[1] based on the same 20 Chicago hotels, and remove some reviews on the basis of the following constraints:

- Delete all non-five star reviews;
- Delete all non-English reviews;
- Delete all reviews which are less than 75 characters;
- Delete all reviews written by first-time authors;

2124 reviews are gathered after filtering. 400 of them are chosen as truthful ones for balancing the number of deceptive reviews, as well as maintaining consistent with the distribution of the length of deceptive reviews. 800 reviews constitute whole gold-standard dataset at last.

5.2 Experiment Setup

We conduct 10-fold cross-validation: the dataset is randomly split into ten folds, where nine folds are selected for training and the tenth fold for test. In training dataset, it contains 360 truthful reviews and 360 deceptive ones. This paper is intended to apply PU learning to identify deceptive reviews. We specially make the following setting: take 20% of the truthful reviews in training set as positive examples dataset P, all remaining truthful and deceptive reviews in training set as the unlabeled dataset U. Therefore, during one round of the algorithm, the training set contains 720 examples including 72 positive examples (set P) and 648 unlabeled examples (set U), and the test set contains 80 examples including 40 positive and 40 negative ones. In order to verify the stability of the proposed method, we also experiment another two different settings, which account for 30% and 40% of the truthful reviews in training set as positive examples dataset P respectively.

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

5.3 Human Performance

Human performance reflects the degree of difficulty to address this task. The rationality of PU learning is closely related to human performance.

We solicit the help of three volunteer students, who were asked to make judgments on test subset (corresponding to the tenth fold of our cross-validation experiments, contains 40 deceptive reviews and 40 truthful reviews). Additionally, to test the extent to which the individual human judges are biased, we evaluate the performance of two virtual meta-judges: one is the MAJORITY meta-judge when at last two out of three human judge believe the review to be deceptive, and the other is the SKEPTIC when any human judge believes the review to be deceptive. It is apparent from the results that human judges are not particularly effective at this task (Table 1). Inter-annotator agreement among the three judges, computed using Fleiss' kappa, is 0.09. Landis and Koch (Landis and Koch, 1977) suggest that scores in the range (0.00, 0.20) correspond to "slight agreemen" between annotators. The largest pairwise Cohen's kappa is 0.11 between JUDGE-1 and JUDGE-3, far below generally accepted pairwise agreement levels. We can infer that the dataset which are annotated by people will include a lot of mislabeled examples. Identifying deceptive reviews by simply using supervised learning methods is not appropriate. So we propose addressing this issue by using PU learning.

Table 1: Human performance

Methods		Accuracy (%)
Human	JUDGE-1	57.9
	JUDGE-2	55.4
	JUDGE-3	61.7
META	MAJORITY	58.3
	SKEPTIC	62.4

6 Results and Analysis

In order to verify the effectiveness of our proposed method, we perform two PU learning (LELC and SPUL) in the gold-standard dataset.

6.1 Experimental Results

Table 2 shows that the experimental results compared with different PU learning techniques. In Table 2, $P(20\%)$ means that we randomly select 20 percentages of truthful reviews to form the positive examples subset P. In our MPIPUL framework, we set $\lambda = 0.3$. We can see that our proposed method can obtain 83.91%, 85.43% and 86.69% in accuracy from different experimental settings, respectively. Compared to the current best method (SPUL-global), the accuracy can be improved 2.06% on average. MPUPUL can improve 3.21% on average than LELC. The above discussion shows our proposed methods consistently outperform the other PU baselines.

Table 2: Accuracy on the different PU learning

Baselines	P(20%)	P(30%)	P(40%)
LELC	81.12	82.08	83.21
SPUL-local	81.43	82.71	84.09
SPUL-global	81.89	83.24	84.73
MPIPUL (0.3)	83.91	85.43	86.69

PU learning framework in this paper can obtain the better performance. Two factors contribute to the improved performance. Firstly, LDA can capture the deeper information of the reviews in topic distribution. Secondly, strategies of mixing population and individual property can generate the similarity weights for *spy* examples, and these examples and their similarity weights are extended into SVM, which can build a more accurate classifier.

6.2 Parameter Sensitivity

For the *spy* examples, the similarity weights are generated by population property and individual property. Should we select the more population information or individual information? In MPIPUL, parameter λ is utilized to adjust this process. So we experiment with the different value of the parameter λ on MPUPUL performance (Figure 2).

As showed in Figure 2, for $P(20\%)$, if $\lambda < 0.3$, the performance increases linearly, if $\lambda > 0.3$, the performance will decrease linearly. Meanwhile, we can get the same trends for $P(30\%)$ and $P(40\%)$. Based on the above discussion, MPIPUL can get the best performance when $\lambda \approx 0.3$.

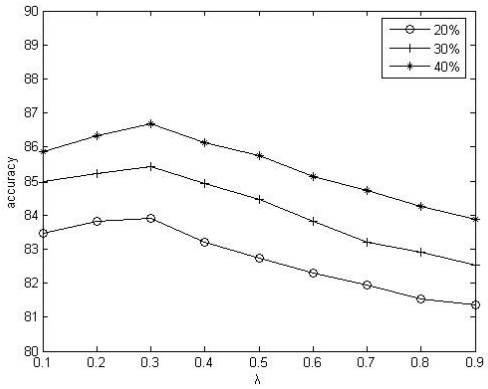

Figure 2: Algorithm performance on different parameter

7 Conclusions and Future Work

This paper proposes a novel PU learning (MPIPUL) technique to identify deceptive reviews based on LDA and SVM. Firstly, the *spy* examples are assigned similarity weights by integrating the information from the population property and individual property. Then the *spy* examples and their similarity weights are incorporated into SVM learning phase to build an accurate classifier. Experimental results on gold-standard dataset show the effectiveness of our method.

In future work, we will discuss the application of our proposed method in the massive dataset.

Acknowledgments

We are grateful to the anonymous reviewers for their thoughtful comments. This work is supported by the State Key Program of National Natural Science Foundation of China (Grant No.61133012), the National Natural Science Foundation of China (Grant No.61173062, 61373108) and the National Philosophy Social Science Major Bidding Project of China (Grant No. 11&ZD189).

References

Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. Detecting spam web pages through content analysis. In *Proceedings of the 15th International Conference on World Wide Web*, page 83-92, Edinburgh, Scotland.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Spotting opinion spammers using behavioral footprints. In *Proceeding of the 19th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Ming, page 632-640, Lyon, France.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceeding of the 21st International Conference on World Wide Web*, page 191-200, New York, USA.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Morgan & Claypool Publishers*. San Rafael, USA.

Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *Proceedings of the 19th International Conference on Machine Learning*, page 387-394, San Francisco, USA.

Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2003. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Ming*, page 179-182, Washington, USA.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1-135.

Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Ming*, page 213-220, Las Vegas, USA.

Chirs Buckley, Bgrard Salton, and James Allan. 1994. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the 17th Annual International SIGIR Conference on Research and Development Retrieval*, page 292-300, Dublin, Ireland.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993-1022.

Dell Zhang. 2005. A simple probabilistic approach to learning from positive and unlabeled examples. In *Proceedings of the 5th Annual UK Workshop on Computational Intelligence*, page 83-87.

Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *Proceeding of the 22nd International Joint Conference on Artificial Intelligence*, page 2488-2493, Barcelona, Spain.

Fang Wu and Bernardo A. Huberman. 2010. Opinion formation under costly express. *ACM Transactions on Intelligence System Technology*, 1(5):1-13.

Gert R. G. Lanckeriet, Nello Cristianini, Peter Bartlett, Laurent EI Ghaoui, and Michael I.Jordan. 2004. Learning the kernel matrix with seim-difinit programming. *Journal of Machine Learning Research*, 5:27-72.

Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048-1054.

Kumar Ankita and Sminchisescu Cristian. 2006. Support kernel machines for object recognition. In *Proceedings of the IEEE 11th International Conference on Computer Vision*, page 1-8, Rio de Janeiro, Brzail.

Myle Ott, Yelin Choi, Claire Caridie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceeding of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technoloies*, page 309-319, Portland, USA.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceeding of the 1st ACM International Conference on Web Search and Data Mining*, page 137-142, California, USA.

Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159-174.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012. Distributional footprints of deceptive product reviews. In *Proceeding of the 6th International AAAI Conference on WebBlogs and Social Media*, page 98-105, Dublin, Ireland.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceeding of the 50th Annual Meeting of the Association for Computational Linguistics*, page 171-175, Jeju Island, Korea.

Vladimir N. Vapnik. 1998. Statistical learning theory. *Springer*. New York, USA.

Wanjui Lee, Sergey Verzakov, and Robert P. Duin. 2007. Kernel combination versus classifier combination. In *Proceedings of the 7th International Workshop on Multiple Classifier Systems*, page 22-31, Rrague, Czech Republic.

Xiaoli Li, Philip S. Yu, Bing Liu, and See Kiong Ng. 2009. Positive unlabeled learning for data stream classification. In *Proceedings of the SIAM International Conference on Data Ming*, page 257-268, Nevada, USA.

Yafeng Ren, Donghong Ji, Lan Yin, and Hongbin Zhang. 2014. Finding deceptive opinion spam by correcting the mislabled instances. *Chinese Journal of Electronics*, 23(4):702-707.

Yafeng Ren, Lan Yin, and Donghong Ji. 2014. Deceptive reviews detection based on language structure and sentiment polarity. *Journal of Frontiers of Computer Science and Technology*, 8(3):313-320.

Yanshan Xiao, Bing Liu, Jie Yin, Longbing Cao, Chengqi Zhang, and Zhifeng Hao. 2011. Similarity-based approach for positive and unlabeled learning. In *Proceeding of the 22nd International Joint Conference on Artifical Intelligence*, page 1577-1582, Barcelona, Spain.

Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedesen. 2004. Combating web spam web with trustrank. In *Proceedings of the 30th International Conference on Very Large Data Bases*, page 576-587, Toronto, Canada.

Appendix A

The optimization problem is as follows:

$$\begin{aligned} min \qquad & F(w,b,\epsilon) = \frac{1}{2}||w||^2 + C_1\sum_{i=1}^{|P|}\epsilon_i + C_2\cdot \\ & \sum_{j=1}^{|US|}p^+(x^{(j)})\epsilon_j + C_3\sum_{m=1}^{|US|}p^-(x^{(m)})\epsilon_m + \\ & C_4\sum_{n=1}^{|NS|}\epsilon_n \\ s.t. \quad & y^{(i)}(w^Tx^{(i)}+b) \geq 1-\epsilon_i, \quad x^{(i)} \in P \\ & y^{(j)}(w^Tx^{(j)}+b) \geq 1-\epsilon_j, \quad x^{(j)} \in US \\ & y^{(m)}(w^Tx^{(m)}+b) \geq 1-\epsilon_m, \quad x^{(m)} \in US \\ & y^{(n)}(w^Tx^{(n)}+b) \geq 1-\epsilon_n, \quad x^{(n)} \in NS \\ & \epsilon_i \geq 0, \quad \epsilon_j \geq 0, \quad \epsilon_m \geq 0, \quad \epsilon_n \geq 0 \end{aligned} \tag{10}$$

We construct the Lagrangian function for the above optimization problem, we have:

$$\begin{aligned} L(w,b,\epsilon,\alpha,\gamma) = & F(w,b,\epsilon) + \sum_{i=1}^{|P|}\alpha_i[-y^{(i)}\cdot \\ & (w^Tx^{(i)}+b)+1-\epsilon_i] + \sum_{j=1}^{|US|}\alpha_j[-y^{(j)}(w^Tx^{(j)}+ \\ & b)+1-\epsilon_j] + \sum_{m=1}^{|US|}\alpha_m[-y^{(m)}(w^Tx^{(m)}+b)+1 \\ & -\epsilon_m] + \sum_{n=1}^{|NS|}\alpha_n[-y^{(n)}(w^Tx^{(n)}+b)+1-\epsilon_n] - \\ & \sum_{i=1}^{|P|}\gamma_i\epsilon_i - \sum_{j=1}^{|US|}\gamma_j\epsilon_j - \sum_{m=1}^{|US|}\gamma_m\epsilon_m - \sum_{n=1}^{|NS|}\gamma_n\epsilon_n \end{aligned} \tag{11}$$

Here, the α and γ are Lagrange multipliers. To find the dual form of the problem, we need to first minimize $L(w,b,\epsilon,\alpha,\gamma)$ with respect to w and b, we will do by setting the derivatives of L with respect to w and b to zero, we have:

$$\begin{aligned} \frac{\partial L(w,b,\epsilon,\alpha,\gamma)}{\partial w} = & w - \sum_{i=1}^{|P|}\alpha_i y^{(i)}x^{(i)} - \sum_{j=1}^{|US|} \\ & \alpha_j y^{(j)}x^{(j)} - \sum_{m=1}^{|US|}\alpha_m y^{(m)}x^{(m)} - \sum_{n=1}^{|NS|}\alpha_n y^{(n)}\cdot \\ & x^{(n)} = 0 \end{aligned} \tag{12}$$

This implies that

$$\begin{aligned} w = & \sum_{i=1}^{|P|}\alpha_i y^{(i)}x^{(i)} + \sum_{j=1}^{|US|}\alpha_j y^{(j)}x^{(j)} + \sum_{m=1}^{|US|}\alpha_m\cdot \\ & y^{(m)}x^{(m)} + \sum_{n=1}^{|NS|}\alpha_n y^{(n)}x^{(n)} \end{aligned} \tag{13}$$

Here, to simplify the presentation, we redefine some notations in the following:

$$T^+ = P \cup US, T^- = US \cup NS, T^* = T^+ \cup T^-$$

$$C_i^+ = \begin{cases} C_1, & x^{(i)} \in P \\ C_2 p^+ x^{(j)}, & x^{(j)} \in US \end{cases}$$

$$C_j^- = \begin{cases} C_3 p^- x^{(m)}, & x^{(m)} \in US \\ C_4, & x^{(n)} \in NS \end{cases}$$

so we obtain

$$w = \sum_{i=1}^{|T^*|}\alpha_i y^{(i)}x^{(i)} \tag{14}$$

As for the derivative with respect to b, we obtain

$$\begin{aligned} \frac{\partial L(w,b,\epsilon,\alpha,\gamma)}{\partial b} = & -\sum_{i=1}^{|P|}\alpha_i y^{(i)} - \sum_{j=1}^{|US|}\alpha_j y^{(j)} \\ & - \sum_{m=1}^{|US|}\alpha_m y^{(m)} - \sum_{n=1}^{|NS|}\alpha_n y^{(n)} = 0 \end{aligned} \tag{15}$$

We get:

$$\sum_{i=1}^{|T^*|}\alpha_i y^{(i)} = 0 \tag{16}$$

If we take Equation (14) and (16) back into the Lagrangian function (Equation 11), and simplify, we get

$$L(w,b,\epsilon,\alpha,\gamma) = \sum_{i=1}^{|T^*|}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{|T^*|} y^{(i)}y^{(j)}\alpha_i \cdot \alpha_j < x^{(i)}, x^{(j)} > \tag{17}$$

To the primal optimization formula (10), we can obtain the following dual optimization problem:

$$\begin{aligned} max \qquad W(\alpha) &= \sum_{i=1}^{|T^*|}\alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{|T^*|} \alpha_i\alpha_j y^{(i)} \cdot y^{(j)} < x^{(i)}, x^{(j)} > \\ s.t. \quad & C_i^+ \geq \alpha_i \geq 0, \quad x^{(i)} \in T^+ \\ & C_j^- \geq \alpha_j \geq 0, \quad x^{(j)} \in T^- \\ & \sum_{i=1}^{|T^+|}\alpha_i - \sum_{j=1}^{|T^-|}\alpha_j = 0 \end{aligned} \tag{18}$$

where $< x^{(i)}, x^{(j)} >$ is the inner product of $x^{(i)}$ and $x^{(j)}$, we can replace them by using kernel function $\phi(x^{(i)})$ and $\phi(x^{(j)})$, respectively.