Negative Training Data can be Harmful to Text Classification

Xiao-Li Li
Institute for Infocomm Research
1 Fusionopolis Way #21-01,
Connexis Singapore 138632
xlli@i2r.a-star.edu.sg

Bing Liu
University of Illinois at Chicago
851 South Morgan Street,
Chicago, IL 60607-7053, USA
liub@cs.uic.edu

See-Kiong Ng
Institute for Infocomm Research
1 Fusionopolis Way #21-01,
Connexis Singapore 138632
skng@i2r.a-star.edu.sg

Abstract

This paper studies the effects of training data on binary text classification and postulates that negative training data is not needed and may even be harmful for the task. Traditional binary classification involves building a classifier using labeled positive and negative training examples. The classifier is then applied to classify test instances into positive and negative classes. A fundamental assumption is that the training and test data are identically distributed. However, this assumption may not hold in practice. In this paper, we study a particular problem where the positive data is identically distributed but the negative data may or may not be so. Many practical text classification and retrieval applications fit this model. We argue that in this setting negative training data should not be used, and that PU learning can be employed to solve the problem. Empirical evaluation has been conducted to support our claim. This result is important as it may fundamentally change the current binary classification paradigm.

1 Introduction

Text classification is a well-studied problem in machine learning, natural language processing, and information retrieval. To build a text classifier, a set of training documents is first labeled with predefined classes. Then, a supervised machine learning algorithm (e.g., Support Vector Machines (SVM), naïve Bayesian classifier (NB)) is applied to the training examples to build a classifier that is subsequently employed to assign class labels to the instances in the test set. In this paper, we focus on binary text classification with two classes (i.e. *positive* and *negative* classes).

Most learning methods assume that the training and test data have identical distributions. However, this assumption may not hold in practice, i.e., the training and the test distributions can be different. The problem is called *covariate shift* or *sample selection bias* (Heckman 1979; Shimodaira 2000; Zadrozny 2004; Huang et al. 2007; Sugiyama et al. 2008; Bickel et al. 2009). In general, this problem is not solvable because the two distributions can be arbitrarily far apart from each other. Various assumptions were made to solve special cases of the problem. One main assumption was that the conditional distribution of the class given an instance is the same over the training and test sets (Shimodaira 2000; Huang et al. 2007; Bickel et al. 2009).

In this paper, we study another special case of the problem in which the positive training and test samples have identical distributions, but the negative training and test samples may have different distributions. We believe this scenario is more applicable for binary text classification. As the focus in many applications is on identifying positive instances correctly, it is important that the positive training and the positive test data have the same distribution. The distributions of the negative training and negative test data can be different. We believe that this special case of the sample selection bias problem is also more applicable for machine learning. We will show that a partially supervised learning model, called *PU learning* (learning from *P*ositive and *U*nlabeled examples) fits this special case quite well (Liu et al. 2002).

Following the notations in (Bickel et al. 2009), our special case of the sample selection bias problem can be formulated as follows: We are given a training sample matrix $\mathbf{X}_L$ with row vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$. The positive and negative training instances are governed by different unknown distributions $p(\mathbf{x}|\lambda)$

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 218–228,
MIT, Massachusetts, USA, 9-11 October 2010.

and $p(\mathbf{x}|\delta)$ respectively. The element y_i of vector $\mathbf{y} = (y_1, y_2, \ldots, y_k)$ is the class label for training instance x_i ($y_i \in \{+1, -1\}$, where +1 and -1 denote positive and negative classes respectively) and is drawn based on an unknown target concept $p(y|\mathbf{x})$. In addition, we are also given an unlabeled test set in matrix $\mathbf{X}_T$ with rows $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+m}$. The (hidden) positive test instances in $\mathbf{X}_T$ are also governed by the unknown distribution $p(\mathbf{x}|\lambda)$, but the (hidden) negative test instances in $\mathbf{X}_T$ are governed by an unknown distribution, $p(\mathbf{x}|\theta)$, where θ may or may not be the same as δ. $p(\mathbf{x}|\theta)$ and $p(\mathbf{x}|\delta)$ can differ arbitrarily, but there is only one unknown target conditional class distribution $p(y|\mathbf{x})$.

This problem setting is common in many applications, especially in those applications where the user is interested in identifying a particular type of documents (i.e. binary text classification). For example, we want to find sentiment analysis papers in the literature. For training a text classifier, we may label the papers in some EMNLP proceedings as sentiment analysis (positive) and non-sentiment analysis (negative) papers. A classifier can then be built to find sentiment analysis papers from ACL and other EMNLP proceedings. However, this labeled training set will not be appropriate for identifying sentiment analysis papers from the WWW, KDD and SIGIR conference proceedings. This is because although the sentiment analysis papers in these proceedings are similar to those in the training data, the non-sentiment analysis papers in these conferences can be quite different. Another example is email spam detection. A spam classification system built using the training data of spam and non-spam emails from a university may not perform well in a company. The reason is that although the spam emails (e.g., unsolicited commercial ads) are similar in both environments, the non-spam emails in them can be quite different.

One can consider labeling the negative data in each environment individually so that only the negative instances relevant to the testing environment are used to train the classifier. However, it is often impractical (if not impossible) to do so. For example, given a large blog hosting site, we want to classify its blogs into those that discuss stock markets (positive), and those that do not (negative). In this case, the negative data covers an arbitrary range of topics. It is clearly impractical to label all the negative data.

Most existing methods for addressing the sample selection bias problem work as follows. First, they estimate the bias of the training data based on the given test data using statistical methods. Then, a classifier is trained on a weighted version of the original training set based on the estimated bias. In this paper, we show that our special case of the sample selection bias problem can be solved in a much simpler and somewhat radical manner—by simply discarding the negative training data altogether. We can use the positive training data and the unlabeled test data to build the classifier using the PU learning model (Liu et al. 2002).

PU learning was originally proposed to solve the learning problem where no labeled negative training data exist. Several algorithms have been developed in the past few years that can learn from a set of labeled positive examples augmented with a set of unlabeled examples. That is, given a set P of positive examples of a particular class (called the *positive* class) and a set U of unlabeled examples (which contains both hidden positive and hidden negative examples), a classifier is built using P and U to classify the data in U as well as future test data into two classes, i.e., those belonging to P (positive) and those not belonging to P (negative). In this paper, we also propose a new PU learning method which gives more consistently accurate results than the current methods.

Our experimental evaluation shows that when the distributions of the negative training and test samples are different, PU learning is much more accurate than traditional supervised learning from the positive and negative training samples. This means that the negative training data actually harms classification in this case. In addition, when the distributions of the negative training and test samples are identical, PU learning is shown to perform equally well as supervised learning, which means that the negative training data is not needed.

This paper thus makes three contributions. First, it formulates a new special case of the sample selection bias problem, and proposes to solve the problem using PU learning by discarding the negative training data. Second, it proposes a new PU learning method which is more accurate than the existing methods. Third, it experimentally demonstrates the effectiveness of the proposed method and shows that negative training data is not needed and can even be harmful. This result is important as it may fundamentally change the way that many practical classification problems should be solved.

2 Related Work

A key assumption made by most machine learning algorithms is that the training and test samples must be drawn from the same distribution. As mentioned, this assumption can be violated in practice. Some researchers have addressed this problem under *covariate shift* or *sample selection bias*. Sample selection bias was first introduced in the econometrics by Heckman (1979). It came into the field of machine learning through the work of Zadrozny (2004). The main approach in machine learning is to first estimate the distribution bias of the training data based on the test data, and then learn using weighted training examples to compensate for the bias (Bickel et al. 2009).

Shimodaira (2000) and Sugiyama and Muller (2005) proposed to estimate the training and test data distributions using kernel density estimation. The estimated density ratio could then be used to generate weighted training examples. Dudik et al. (2005) and Bickel and Scheffer (2007) used maximum entropy density estimation, while Huang et al. (2007) proposed kernel mean matching. Sugiyama et al. (2008) and Tsuboi et al. (2008) estimated the weights for the training instances by minimizing the Kullback-Leibler divergence between the test and the weighted training distributions. Bickel et al. (2009) proposed an integrated model. In this paper, we adopt an entirely different approach by dropping the negative training data altogether in learning. Without the negative training data, we use PU learning to solve the problem (Liu et al. 2002; Yu et al. 2002; Denis et al. 2002; Li et al. 2003; Lee and Liu, 2003; Liu et al. 2003; Denis et al. 2003; Li et al. 2007; Elkan and Noto, 2008; Li et al. 2009; Li et al. 2010). We will discuss this learning model further in Section 3.

Another related work to ours is transfer learning or domain adaptation. Unlike our problem setting, transfer learning addresses the scenario where one has little or no training data for the target domain, but has ample training data in a related domain where the data could be in a different feature space and follow a different distribution. A survey of transfer learning can be found in (Pan and Yang 2009). Several NLP researchers have studied transfer learning for different applications (Wu et al. 2009a; Yang et al. 2009; Agirre & Lacalle 2009; Wu et al. 2009b; Sagae & Tsujii 2008; Goldwasser & Roth 2008; Li and Zong 2008; Andrew et al. 2008; Chan and Ng 2007; Jiang and Zhai 2007; Zhou et al. 2006), but none of them addresses the problem studied here.

3 PU Learning Techniques

In traditional supervised learning, ideally, there is a large number of labeled positive and negative examples for learning. In practice, the negative examples can often be limited or unavailable. This has motivated the development of the model of *learning from positive and unlabeled examples*, or PU learning, where *P* denotes a set of positive examples, and *U* a set of unlabeled examples (which contains both hidden positive and hidden negative instances). The PU learning problem is to build a classifier using *P* and *U* in the absence of negative examples to classify the data in *U* or a future test data *T*. In our setting, the test set *T* will also act as the unlabeled set *U*.

PU learning has been investigated by several researchers in the past decade. A study of PAC learning for the setting under the statistical query model was given in (Denis, 1998). Liu et al. reported the sample complexity result and showed how the problem may be solved (Liu et al., 2002). Subsequently, a number of practical algorithms (e.g., Liu et al., 2002; Yu et al., 2002; Li and Liu, 2003) were proposed. They generally follow a two-step strategy: (i) identifying a set of *reliable negative* documents *RN* from the unlabeled set; and then (ii) building a classifier using *P* (positive set), *RN* (reliable negative set) and *U-RN* (unlabelled set) by applying an existing learning algorithm (such as naive Bayesian classifier or SVM) iteratively. There are also some other approaches based on unbalanced errors (e.g., Liu et al. 2003; Lee and Liu, 2003; Elkan and Noto, 2008).

In this section, we first introduce a representative PU learning technique S-EM, and then present a new technique called CR-SVM.

3.1 S-EM Algorithm

S-EM (Liu et al. 2002) is based on naïve Bayesian classification (NB) (Lewis, 1995; Nigam et al., 2000) and the EM algorithm (Dempster et al. 1977). It has two steps. The first step uses a *spy* technique to identify some *reliable negatives* (*RN*) from the unlabeled set *U* and the second step uses the EM algorithm to learn a Bayesian classifier from *P*, *RN* and *U–RN*.

Step 1: Extracting reliable negatives *RN* from *U* using a spy technique

The *spy* technique in S-EM works as follows (Figure 1): First, a small set of positive examples (denoted by *SP*) called "spies" is randomly sampled from *P* (line 2). The default sampling ratio in S-EM is s = 15%. Then, an NB classifier is built using *P–SP* as the positive set and $U \cup SP$ as the negative set (lines 3-5). The NB classifier is applied to classify each $u \in U \cup SP$, i.e., to assign a probabilistic class label $p(+|u)$ (+ means positive) to u. The idea of the spy technique is as follows. Since the spy examples were from *P* and were put into *U* as negatives in building the NB classifier, they should behave similarly to the hidden positive instances in *U*. We thus can use them to find the reliable negative set *RN* from *U*. Using the probabilistic labels of spies in *SP* and an input parameter l (noise level), a probability threshold t is determined. Due to space constraints, we are unable to explain l. Details can be found in (Liu et al. 2002). t is then used to find *RN* from *U* (lines 8-10).

1. $RN \leftarrow \emptyset$; // Reliable negative set
2. $SP \leftarrow Sample(P, s\%)$; // spy set
3. Assign each example in $P - SP$ the class label +1;
4. Assign each example in $U \cup SP$ the class label -1;
5. $C \leftarrow$NB($P - SP$, $U \cup SP$); // Produce a NB classifier
6. Classify each $u \in U \cup SP$ using C;
7. Decide a probability threshold t using *SP* and l;
8. **For** each $u \in U$ **do**
9.   **If** its probability $p(+|u) < t$ **then**
10. $RN \leftarrow RN \cup \{u\}$;

Figure 1. Spy technique for extracting *RN* from *U*

Step 2: Learning using the EM algorithm

Given the positive set *P*, the reliable negative set *RN,* and the remaining unlabeled set *U–RN*, we run EM using NB as the base learning algorithm.

The naive Bayesian (NB) method is an effective text classification algorithm. There are two different NB models, namely, the multinomial NB and the multi-variate Bernoulli NB. In this paper, we use the multinomial NB since it has been observed to perform consistently better than the multi-variate Bernoulli NB (Nigam et al., 2000).

Given a set of training documents *D*, each document $d_i \in D$ is an ordered list of words. We use $w_{d_i,k}$ to denote the word in position k of d_i, where each word is from the vocabulary $V = \{w_1, \ldots, w_{|v|}\}$, which is the set of all words considered in classification. We also have a set of classes $C = \{c_1, c_2\}$ representing positive and negative classes. For classification, we compute the posterior probability $\Pr(c_j|d_i)$. Based on the Bayes rule and multinomial model, we have

$$\Pr(c_j) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|} \quad (1)$$

and with Laplacian smoothing,

$$\Pr(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i) \Pr(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) \Pr(c_j \mid d_i)} \quad (2)$$

where $N(w_t,d_i)$ is the number of times that the word w_t occurs in document d_i, and $\Pr(c_j|d_i) \in \{0,1\}$ depending on the class label of the document. Assuming that probabilities of words are independent given the class, we have the NB classifier:

$$\Pr(c_j \mid d_i) = \frac{\Pr(c_j) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_j)}{\sum_{r=1}^{|C|} \Pr(c_r) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_r)} \quad (3)$$

Each document in *P* is assigned the class label 1;
Each document in *RN* is assigned the class label −1;
Learn an initial NB classifier *f* from *P* and *RN*, using Equations (1) and (2);
Repeat
    For each document d_i in *U-RN* **do** // **E-Step**
        Using the current classifier *f* compute $\Pr(c_j|d_i)$ using Equation (3);
    Learn a new NB classifier *f* from *P, RN* and *U-RN* by computing $\Pr(c_j)$ and $\Pr(w_t|c_j)$, using Equations (1) and (2); // **M-Step**
Until the classifier parameters stabilize
The last iteration of EM gives the final classifier *f* ;
For each document d_i in *U* **do**
    If its probability $\Pr(+|d_i) \geq 0.5$ **then**
        Output d_i as a positive document;
    else Output d_i as a negative document

Figure 2. EM algorithm with the NB classifier

EM (Dempster et al. 1977) is a popular class of iterative algorithms for maximum likelihood estimation in problems with incomplete data. It is often used to address missing values in the data by computing expected values using the existing values. The EM algorithm consists of two steps, the E-step and the M-step. The E-step fills in the missing data, and M-step re-estimated the parameters. This process is iterated till satisfaction (i.e. convergence). For NB, the steps used by EM are identical to those used to build the classifier (equations (3) for the E-step, and equations (1) and (2) for the

M-step). In EM, $\Pr(c_j|d_i)$ takes the value in [0, 1] instead of {0, 1} in all the three equations.

The algorithm for the second step of S-EM is given in Figure 2. Lines 1-3 build a NB classifier f using P and RN. Lines 4-8 run EM until convergence. Finally, the converged classifier is used to classify the unlabeled set U (lines 10-13).

3.2 Proposed CR-SVM

As we will see in the experiment section, the performance of S-EM can be weak in some cases. This is due to the mixture model assumption of its NB classifier (Nigam et al. 2000), which requires that the mixture components and classes be of one-to-one correspondence. Intuitively, this means that each class should come from a distinctive distribution rather than a mixture of multiple distributions. In our setting, however, the negative class often has documents of mixed topics, e.g., representing the broad class of everything else except the topic(s) represented by the positive class.

There are some existing PU learning methods based on SVM which can deal with this problem, e.g., Roc-SVM (Li and Liu, 2003). Like S-EM, Roc-SVM also has two steps. The first step uses Rocchio classification (Rocchio, 1971) to find a set of reliable negatives RN from U. In particular, this method treats the entire unlabeled set U as negative documents and then uses the positive set P and the unlabeled set U as the training data to build a Rocchio classifier. The classifier is subsequently applied to classify the unlabeled set U. Those documents that are classified as negative are then considered as reliable negative examples RN. The second step of Roc-SVM runs SVM iteratively (instead of EM). Unlike NB, SVM does not make any distributional assumption.

However, Roc-SVM does not do well due to the weakness of its first step in finding a good set of reliable negatives RN. This motivates us to propose a new SVM based method CR-SVM to detect a better quality RN set. The second step of CR-SVM is similar to that in Roc-SVM.

Step 1: Extracting reliable negatives *RN* from *U* using Cosine and Rocchio

The first step of the proposed CR-SVM algorithm for finding a RN set consists of two sub-steps:

Sub-step 1 (extracting the potential negative set PN using the *cosine similarity*): Given the positive set P and the unlabeled set U, we extract a set of potential negatives PN from U by computing the similarities of the unlabeled documents in U and the positive documents in P. The idea is that those documents in U that are very dissimilar to the documents in P are likely to be negative.

1. $PN = \varnothing$;
2. Represent each document in P and U as vectors using the TF-IDF representation;
3. **For** each $\mathbf{d}_j \in P$ **do**
4. $\mathbf{pr} = \frac{1}{|P|} * \sum_{j=1}^{|P|} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|_2}$
5. $\mathbf{pr} = \mathbf{pr} / \|\mathbf{pr}\|_2$;
6. **For** each $\mathbf{d}_j \in P$ **do**
7. compute $cos(\mathbf{pr}, \mathbf{d}_j)$ using Equation (4);
8. Sort all the documents $\mathbf{d}_j \in P$ according to $cos(\mathbf{pr}, \mathbf{d}_j)$ in decreasing order;
9. $\omega = cos(\mathbf{pr}, \mathbf{d}_p)$ where $\mathbf{d}_p$ is ranked in the position of $(1- l)*|P|$;
10. **For** each $\mathbf{d}_i \in U$ **do**
11. **If** $cos(\mathbf{pr}, \mathbf{d}_i) < \omega$ **then**
12. $PN = PN \cup \{\mathbf{d}_i\}$

Figure 3. Extracting potential negatives PN from U

The detailed algorithm is given in Figure 3. Each document in P and U is first represented as a vector $\mathbf{d} = (q_1, q_2, \ldots, q_n)$ using the TF-IDF scheme (Salton 1986). Each element q_i (i=1, 2, …, n) in $\mathbf{d}$ represents a word feature w_i. A positive representative vector ($\mathbf{pr}$) is built by summing up the documents in P and normalizing it (lines 3-5). Lines 6-7 compute the similarities of each document $\mathbf{d}_j$ in P with $\mathbf{pr}$ using the cosine similarity, $cos(\mathbf{pr}, \mathbf{d}_j)$.

Line 8 sorts the documents in P according to their $cos(\mathbf{pr}, \mathbf{d}_j)$ values. We want to filter away as many as possible hidden positive documents from U so that we can obtain a very pure negative set. Since the hidden positives in U should have the same behaviors as the positives in P in terms of their similarities to $\mathbf{pr}$, we set their minimum similarity as the threshold value ω which is the minimum similarity before a document is considered as a potential negative document:

$$\omega = \min_{j=1}^{|P|} \cos(\mathbf{pr}, \mathbf{d}_j),\ \mathbf{d}_j \in P \qquad (4)$$

In a noiseless scenario, using the minimum similarity is acceptable. However, most real-life applications contain outliers and noisy artifacts. Using the absolute minimum similarity may be unreliable; the similarity $cos(\mathbf{pr}, \mathbf{d}_j)$ of an outlier docu-

1. $RN = \emptyset$;
2. Represent each document in P, PN and U as vectors using the TF-IDF representation;
3. $\mathbf{p} = \alpha \frac{1}{|P|} \sum_{\mathbf{d}_j \in P} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|} - \beta \frac{1}{|PN|} \sum_{\mathbf{d}_i \in PN} \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|}$;
4. $\mathbf{n} = \alpha \frac{1}{|PN|} \sum_{\mathbf{d}_i \in PN} \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|} - \beta \frac{1}{|P|} \sum_{\mathbf{d}_j \in P} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$;
5. **For** each $\mathbf{d}_i \in U$ **do**
6. **If** $cos(\mathbf{d}_i, \mathbf{n}) > cos(\mathbf{d}_i, \mathbf{p})$ **then**
7. $RN = RN \cup \{\mathbf{d}_i\}$

Figure 4. Identifying *RN* using the Rocchio classifier

ment $\mathbf{d}_j$ in P could be near 0 or smaller than most (or even all) negative documents. It would therefore be prudent to ignore a small percentage l of the documents in P most dissimilar to the representative positive (**pr**) and assume them as noise or outliers. Since we do not know the noise level of the data, to be safe, we use a noise level l = 5% as the default. The final classification result is not sensitive to l as long as it is not too big. In line 9, we use the noise level l to decide on a suitable ω. Then, for each document d_i in U, if its cosine similarity $cos(\mathbf{pr}, \mathbf{d}_i) < \omega$, we regard it as a potential negative and store it in *PN* (lines 10-12).

Our experiment results showed that *PN* is still not sufficient or big enough for accurate PU learning. Thus, we need to do a bit more work to find the final *RN*.

Sub-step 2 (extracting the final reliable negative set *RN* from *U* using Rocchio with *PN*): At this point, we have a positive set *P* and a potential negative set *PN* where *PN* is a purer negative set than *U*. To extract the final reliable negatives, we employ the Rocchio classification to build a classifier *RC* using *P* and *PN* (We do not use SVM here as it is very sensitive to the noise in *PN*). Those documents in *U* that are classified as negatives by *RC* will then be regarded as reliable negatives, and stored in set *RN*.

The algorithm for this sub-step is given in Figure 4. Following the Rocchio formula, a positive and a negative prototype vectors **p** and **n** are built (lines 3 and 4), which are used to classify the documents in *U* (lines 5-7). α and β are parameters for adjusting the relative impact of the positive and negative examples. In this work, we use $\alpha = 16$ and $\beta = 4$ as recommended in (Buckley et al. 1994).

Step 2: Learning by running SVM iteratively

This step is similar to that in Roc-SVM, building the final classifier by running SVM iteratively with the sets *P*, *RN* and the remaining unlabeled set *Q* ($Q = U - RN$).

The algorithm is given in Figure 5. We run SVM classifiers S_i (line 3) iteratively to extract more and more negative documents from *Q*. The iteration stops when no more negative documents can be extracted from *Q* (line 5). There is, however, a danger in running SVM iteratively, as SVM is quite sensitive to noise. It is possible that during some iteration, SVM is misled by noisy data to extract many positive documents from *Q* and put them in the negative set *RN*. If this happens, the final SVM classifier will be inferior. As such, we employ a test to decide whether to keep the first SVM classifier or the final one. To do so, we use the final SVM classifier obtained at convergence (called S_{last}, line 9) to classify the positive set *P* to see if many positive documents in *P* are classified as negatives. Roc-SVM chooses 5% as the threshold, so CR-SVM also uses this threshold. If there are 5% of positive documents (5%*|*P*|) in *P* that are classified as negative, it indicates that SVM has gone wrong and we should use the first SVM classifier (S_1). In our experience, the first classifier is always quite strong; good results can therefore be achieved even without catching the last (possibly better) classifier.

The main difference between Roc-SVM and CR-SVM is that Roc-SVM does not produce *PN*. It simply treats the unlabeled set *U* as negatives for extracting *RN*. Since *PN* is clearly a purer negative set than *U*, the use of *PN* by CR-SVM helps extract a better quality reliable negative set *RN* which subsequently allows the final classifier of CR-SVM to give better results than Roc-SVM.

Note that the methods (S-EM and CR-SVM) are all two-step algorithms in which the first step and the second step are independent of each other. The algorithm for the second step basically needs a good set of reliable negatives *RN* extracted from *U*. This means that one can pick any algorithm for the first step to work with any algorithm for the second step. For example, we can also have CR-EM which uses the algorithm (shown in Figures 3 and 4) of the first step of CR-SVM to combine with the algorithm of the second step of S-EM. CR-EM actually works quite well as it is also able to exploit the more accurate reliable negative set *RN* extracted using cosine and Rocchio.

4 Empirical Evaluation

We now present the experimental results to support our claim that negative training data is not needed and can even harm text classification. We also show the effectiveness of the proposed PU learning methods CR-SVM and CR-EM. The following methods are compared: (1) traditional supervised learning methods SVM and NB which use both positive and negative training data; (2) PU learning methods, including two existing methods S-EM and Roc-SVM and two new methods CR-SVM and CR-EM, and (3) one-class SVM (Schölkop et al., 1999) where only positive training data is used in learning (the unlabeled set is not used at all).

We used LIBSVM[1] for SVM and one-class SVM, and two publicly available[2] PU learning techniques S-EM and Roc-SVM. Note that we do not compare with some other PU learning methods such as those in (Liu et al. 2003, Lee and Liu, 2003 and Elkan and Noto, 2008) as the purpose of this paper is not to find the best PU learning method but to show that PU learning can address our special sample selection bias problem. Our current methods already do very well for this purpose.

4.1 Datasets and Experimental Settings

We used two well-known benchmark data collections for text classification, the Reuters-21578 collection[3] and the 20 Newsgroup collection[4]. Reuters-21578 contains 21578 documents. We used the most populous 10 out of the 135 categories following the common practice of other researchers. 20 Newsgroup has 11997 documents from 20 discussion groups. The 20 groups were also categorized into 4 main categories.

We have performed two sets of experiments, and just used bag-of-words as features since our objective in this paper is not feature engineering.

(1) **Test set has *other topic* documents.** This set of experiments simulates the scenario in which the negative training and test samples have different distributions. We select positive, negative and *other topic* documents for Reuters and 20 Newsgroup, and produce various data sets. Using these data sets, we want to show that PU learning can do better than traditional learning that uses both positive and negative training data.

1. Every document in P is assigned the class label +1;
2. Every document in RN is assigned the label –1;
3. Use P and RN to train a SVM classifier S_i, with i = 1 initially and $i = i+1$ with each iteration (line 3-7);
4. Classify Q using S_i. Let the set of documents in Q that are classified as negative be W;
5. **If** ($W = \emptyset$) **then** *stop*;
6. **else** $Q = Q - W$;
7. $\quad RN = RN \cup W$
8. goto (3);
9. Use the last SVM classifier S_{last} to classify P;
10. **If** more than 5% positives are classified as negative
11. $\quad$ **then** use S_1 as the final classifier;
12. **else** use S_{last} as the final classifier;

Figure 5. Constructing the final classifier using SVM

For the Reuters collection, each of the 10 categories is used as a positive class. We randomly select one or two of the remaining categories as the negative class (denoted by Neg 1 or Neg 2), and then we randomly choose some documents from the rest of the categories as *other topic* documents. These *other topic* documents are regarded as negatives and added to the test set but not to the negative training data. They thus introduce a different distribution to the negative test data. We generated 20 data sets (10*2) for our experiments this way.

The 20 Newsgroup collection has 4 main categories with sub-categories[5]; the sub-categories in the same main category are relatively similar to each other. We are able to simulate two scenarios: (1) the *other topic* documents are similar to the negative class documents (*similar case*), and (2) the *other topic* documents are quite different from the negative class documents (*different case*). This allows us to investigate whether the classification results will be affected when the *other topic* documents are somewhat similar or vastly different from the negative training set. To create the training and test data for our experiments, we randomly select one sub-category from a main category (*cat* 1) as the positive class, and one (or two) sub-category from another category (*cat* 2) as the negative class (again denoted by Neg 1 or Neg 2). For the *other topics*, we randomly choose some docu-

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[2] http://www.cs.uic.edu/~liub/LPU/LPU-download.html
[3] http://www.research.att.com/~lewis/reuters21578.html
[4] http://people.csail.mit.edu/jrennie/20Newsgroups/

[5] The four main categories and their corresponding subcategories are: *computer* (graphics, os, ibmpc.hardware, mac.hardware, windows.x), *recreation* (autos, motorcycles, baseball, hockey), *science* (crypt, electronics, med, space), and *talk* (politics.misc, politics.guns, politics.mideast, religion).

ments from the remaining sub-categories of *cat* 2 for the *similar case*, and some documents from a randomly chosen different category (*cat* 3) (as the *other topic* documents) for the *different case*. We generated 8 data sets (4*2) for the *similar case*, and 8 data sets (4*2) for the *different case*.

The training and test sets are then constructed as follows: we partition the positive (and similarly for the negative) class documents into two standard subsets: 70% for training and 30% for testing. In order to create different experimental settings, we vary the number of the *other topic* documents that are added to the test set as negatives, controlled by a parameter α, which is a percentage of $|TN|$, where $|TN|$ is the size of the negative test set without the *other topic* documents. That is, the number of *other topic* documents added is $\alpha \times |TN|$.

(2) **Test set has no *other topic* documents.** This set of experiments is the traditional classification in which the training and test data have the same distribution. We employ the same data sets as in (1) but without having any *other topic* documents in the test set. Here we want to show that PU learning can do equally well without using the negative training data even in the traditional setting.

4.2 Results with *Other Topic* Documents in Test Set

We show the results for experiment set (1), i.e. the distributions of the negative training and test data are different (caused by the inclusion of *other topic* documents in the test set, or the addition of *other topic* documents to complement existing negatives in the test set). The evaluation metric is the F-score on the positive class (Bollmann and Cherniavsky, 1981), which is commonly used for evaluating text classification.

4.2.1 Results on the Reuters data

Figure 6 shows the comparison results when the negative class contains only one category of documents (Neg 1), while Figure 7 shows the results when the negative class contains documents from two categories (Neg 2) in the Reuters collection. The data points in the figures are the averages of the results from the corresponding datasets.

Our proposed method CR-SVM is shown to perform consistently better than the other techniques. When the size of the *other topic* documents (*x*-axis) in the test set increases, the F-scores of the two traditional learning methods SVM and NB decreased much more dramatically as compared with the PU learning techniques. The traditional learning models were clearly unable to handle different distributions for training and test data. Among the PU learning techniques, the proposed CR-SVM gave the best results consistently. Roc-SVM did not do consistently well as it did not manage to find high quality reliable negatives *RN* sometimes. The EM based methods (CR-EM and S-EM) performed well in the case when we had only one negative class (Figure 6). However, it did not do well in the situation where there were two negative classes (Figure 7) due to the underlying mixture model assumption of the naïve Bayesian classifier. One-class SVM (OSVM) performed poorly because it did not exploit the useful information in the unlabeled set at all.

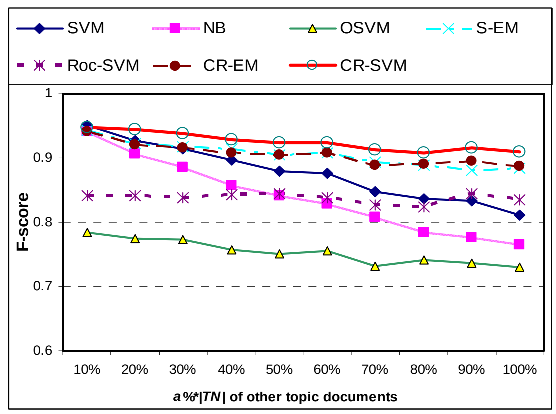

Figure 6. Results of Neg 1 using the Reuter data

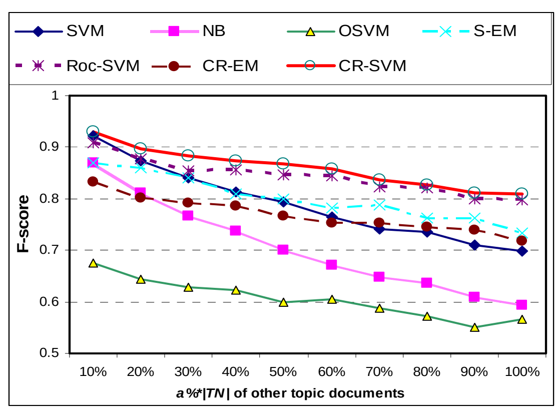

Figure 7. Results of Neg 2 using the Reuter data

4.2.2 Results on 20 Newsgroup data

Recall that for the 20 Newsgroup data, we have two settings: *similar case* and *different case*.

Similar case: Here, the *other topic* documents are similar to the *negative* class documents, as they belong to the same main category.

The comparison results are given in Figure 8 (Neg 1) and Figure 9 (Neg 2). We observe that CR-EM, S-EM and CR-SVM all performed well. EM based methods (CR-EM and S-EM) have a slight edge over CR-SVM. Again, the F-scores of the traditional supervised learning (SVM and NB) deteriorated when more *other topic* documents were added to the test set, while CR-EM, S-EM and CR-SVM were able to remain unaffected and maintained roughly constant F-scores. When the negative class contained documents from two categories (Neg 2), the F-scores of the traditional learning dropped even more rapidly. Both Roc-SVM and One-class SVM (OSVM) performed poorly, due to the same reasons given previously.

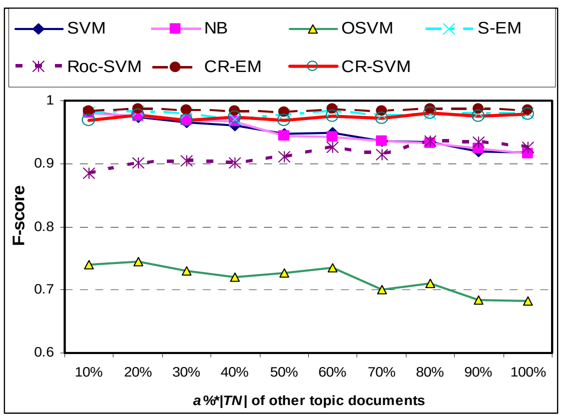

Figure 8. Results of Neg 1, *similar case* – using the 20 Newsgroup data

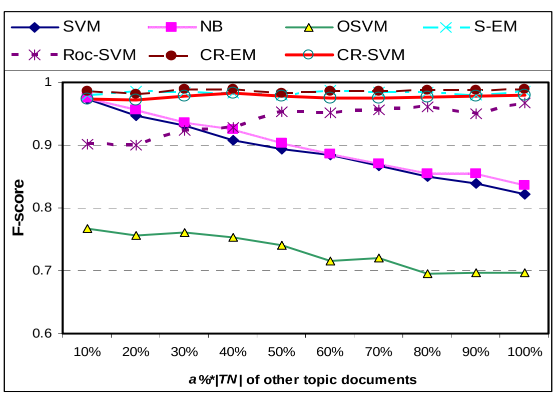

Figure 9. Results of Neg 2, *similar case* – using the 20 Newsgroup data

Different case: In this case, the *other topic* documents are quite different from the negative class documents, since they are originated from different main categories.

The results are shown in Figures 10 (Neg 1) and 11 (Neg 2). The trends are similar to those for the *similar case*, except that the performance of the traditional supervised learning methods (SVM and NB) dropped even more rapidly with more *other topic* documents. As the *other topic* documents have very different distributions from the negatives in the training set in this case, they really confused the traditional classifiers. In contrast, the three PU learning techniques were still able to perform consistently well, regardless of the number of *other topic* documents added to the test data.

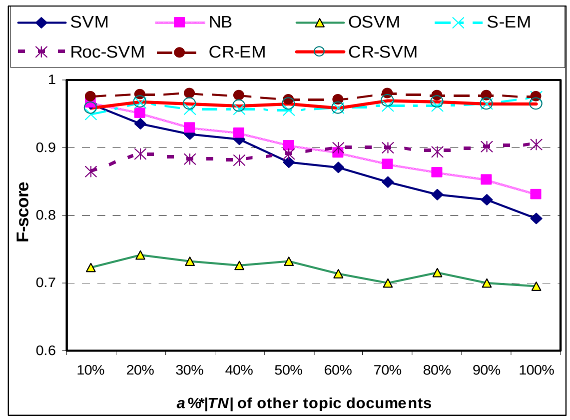

Figure 10. Results of Neg 1, *different case* – using the 20 Newsgroup data

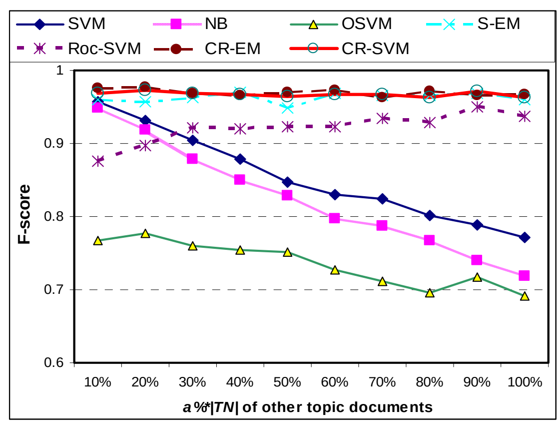

Figure 11. Results of Neg 2, *different case* – using the 20 Newsgroup data

In summary, the results showed that learning with negative training data based on the traditional paradigm actually harms classification when the identical distribution assumption does not hold.

4.3 Results without *Other Topic* Documents in Test Set

Given an application, one may not know whether the identical distribution assumption holds. The above results showed that PU learning is better when it does not hold. How about when the assumption does hold? To find out, we compared the results of SVM, NB, and three PU learning methods using the datasets without any *other topic* documents added to the test set. In this case, the training and test data distributions are the same.

Table 1 shows the results for this scenario. Note that for PU learning, the negative training data were not used. The traditional supervised learning techniques (SVM and NB), which made full use of the positive and negative training data, only performed just about 1-2% better than the PU learning method CR-SVM (which is not statistically significant based on paired *t*-test). This suggests that we can do away with negative training data, since PU learning can perform equally well without them. This has practical importance since the full coverage of negative training data is hard to find and to label in many applications.

From the results in Figures 6–11 and Table 1, we can conclude that PU learning can be used for binary text classification without the negative training data (which can be harmful for the task). CR-SVM is our recommended PU learning method based on its generally consistent performance.

Table 1. Comparison of methods without *other* documents in test set

Methods	Reuters (Neg 1)	Reuters (Neg 2)	20News (Neg 1)	20News (Neg 2)
SVM	0.971	0.964	0.988	0.990
NB	0.972	0.947	0.988	0.992
S-EM	0.952	0.921	0.974	0.975
CR-EM	0.955	0.897	0.983	0.986
CR-SVM	0.960	0.959	0.967	0.974

5 Conclusions

This paper studied a special case of the sample selection bias problem in which the positive training and test distributions are the same, but the negative training and test distributions may be different. We showed that in this case, the negative training data should not be used in learning, and PU learning can be applied to this setting. A new PU learning algorithm (called CR-SVM) was also proposed to overcome the weaknesses of the current two-step algorithms.

Our experiments showed that the traditional classification methods suffered greatly when the distributions are different for the negative training and test data, but PU learning does not. We also showed that PU learning performed equally well in the ideal case where the training and test data have identical distributions. As such, it can be advantageous to discard the potentially harmful negative training data and use PU learning for classification.

In our future work, we plan to do more comprehensive experiments to compare the classic supervised learning and PU learning techniques with different kinds of settings, for example, by varying the ratio between positive and negative examples, as well as their sizes. It is also important to explore how to catch the best iteration of the SVM/NB classifier in the iterative running process of the algorithms. Finally, we would like to point out that it is conceivable that negative training data could still be useful in many cases. An interesting direction to explore is to somehow combine the extracted reliable negative data from the unlabeled set and the existing negative training data to further enhance learning algorithms.

References

Agirre E., Lacalle L.O. 2009. *Supervised Domain Adaption for WSD*. Proceedings of the 12th Conference of the European Chapter for Computational Linguistics (EACL09), pp 42-50.

Andrew A., Nallapati R., Cohen W., 2008. *Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition*, ACL.

Bickel, S., Bruckner, M., and Scheffer. 2009. T. *Discriminative learning under covariate shift*. Journal of Machine Learning Research.

Bickel S. and Scheffer T. 2007. *Dirichlet-enhanced spam filtering based on biased samples*. In Advances in Neural Information Processing Systems.

Bollmann, P.,& Cherniavsky, V. 1981. *Measurement-theoretical investigation of the mz-metric*. Information Retrieval Research.

Buckley, C., Salton, G., & Allan, J. 1994. *The effect of adding relevance information in a relevance feedback environment*, SIGIR.

Blum, A. and Mitchell, T. 1998. *Combining labeled and unlabeled data with co-training*. In Proc. of Computational Learning Theory, pp. 92–10.

Chan Y. S., Ng H. T. 2007. *Domain Adaptation with Active Learning for Word Sense Disambiguation*, ACL.

Dempster A., Laird N. and Rubin D.. 1977. *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society.

Denis F., *PAC learning from positive statistical queries*. ALT, 1998.

Denis F., Laurent A., Rémi G., Marc T. 2003. *Text classification and co-training from positive and unlabeled examples*. ICML.

Denis, F, Rémi G, and Marc T. 2002. *Text Classification from Positive and Unlabeled Examples.* In Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems.

Downey, D., Broadhead, M. and Etzioni, O. 2007. *Locating complex named entities in Web Text.* IJCAI.

Dudik M., Schapire R., and Phillips S. 2005. *Correcting sample selection bias in maximum entropy density estimation*. In Advances in Neural Information Processing Systems.

Elkan, C. and Noto, K. 2008. *Learning classifiers from only positive and unlabeled data*. KDD, 213-220.

Goldwasser, D., Roth D. 2008. *Active Sample Selection for Named Entity Transliteration*, ACL.

Heckman J. 1979. *Sample selection bias as a specification error*. Econometrica, 47:153–161.

Huang J., Smola A., Gretton A., Borgwardt K., and Scholkopf B. 2007. *Correcting sample selection bias by unlabeled data*. In Advances in Neural Information Processing Systems.

Jiang J. and Zhai C. X. 2007. *Instance Weighting for Domain Adaptation in NLP*, ACL.

Lee, W. S. and Liu, B. 2003. *Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression*. ICML.

Lewis D. 1995. *A sequential algorithm for training text classifiers*: *corrigendum and additional data*. SIGIR Forum, 13-19.

Li, S., Zong C., 2008. *Multi-Domain Sentiment Classification*, ACL.

Li, X., Liu, B. 2003. *Learning to classify texts using positive and unlabeled data*, IJCAI.

Li, X., Liu, B., 2005. *Learning from Positive and Unlabeled Examples with Different Data Distributions*. ECML.

Li, X., Liu, B., 2007. *Learning to Identify Unexpected Instances in the Test Set*. IJCAI.

Li, X., Yu, P. S., Liu B., and Ng, S. 2009. *Positive Unlabeled Learning for Data Stream Classification*, SDM.

Li, X., Zhang L., Liu B., and Ng, S. 2010. *Distributional Similarity vs. PU Learning for Entity Set Expansion*, ACL.

Liu, B, Dai, Y., Li, X., Lee, W-S., and Yu. P. 2003. *Building text classifiers using positive and unlabeled examples*. ICDM, 179-188.

Liu, B, Lee, W-S, Yu, P. S, and Li, X. 2002. *Partially supervised text classification.* ICML, 387-394.

Nigam, K., McCallum, A., Thrun, S. and Mitchell, T. 2000. *Text classification from labeled and unlabeled documents using EM*. Machine Learning, 39(2/3), 103–134.

Pan, S. J. and Yang, Q. 2009. *A survey on transfer learning*. IEEE Transactions on Knowledge and Data Engineering, Vol. 99, No. 1.

Rocchio, J. 1971. Relevant feedback in information retrieval. In G. Salton (ed.). *The smart retrieval system: experiments in automatic document processing*,

Englewood Cliffs, NJ, 1971.Sagae K., Tsujii J. 2008. *Online Methods for Multi-Domain Learning and Adaptation*, EMNLP.

Salton G. and McGill M. J. 1986. *Introduction to Modern Information Retrieval*.

Schölkop f B., Platt J.C., Shawe-Taylor J., Smola A.J., and Williamson R.C. 1999. *Estimating the support of a high-dimensional distribution*. Technical report, Microsoft Research, MSR-TR-99-87.

Shimodaira H. 2000. *Improving predictive inference under covariate shift by weighting the log-likelihood function*. Journal of Statistical Planning and Inference, 90:227–244.

Sugiyama M. and Muller K.-R. 2005. *Input-dependent estimation of generalization error under covariate shift*. Statistics and Decision, 23(4):249–279.

Sugiyama M., Nakajima S., Kashima H., von Bunau P., and Kawanabe M. 2008. *Direct importance estimation with model selection and its application to covariate shift adaptation*. In Advances in Neural Information Processing Systems.

Tsuboi J., Kashima H., Hido S., Bickel S., and Sugiyama M. 2008. *Direct density ratio estimation for large-scale covariate shift adaptation*. In Proceedings of the SIAM International Conference on Data Mining, 2008.

Wu D., Lee W.S., Ye N. and Chieu H. L. 2009. *Domain adaptive bootstrapping for named entity recognition*, ACL.

Wu Q., Tan S. and Cheng X. 2009. *Graph Ranking for Sentiment Transfer*, ACL.

Yang Q., Chen Y., Xue G., Dai W., Yu Y. 2009. *Heterogeneous Transfer Learning for Image Clustering via the SocialWeb*, ACL

Yu, H., Han, J., K. Chang. 2002. *PEBL: Positive example based learning for Web page classification using SVM*. KDD, 239-248.

Zadrozny B. 2004. *Learning and evaluating classifiers under s ample selection bias*, ICML.

Zhou Z., Gao J., Soong F., Meng H. 2006. *A Comparative Study of Discriminative Methods for Reranking LVCSR N-best Hypotheses in Domain Adaptation and Generalization*. ICASSP.