Learning to Identify Unexpected Instances in the Test Set

Xiao-Li Li
Institute for Infocomm Research,
21 Heng Mui Keng Terrace,
Singapore, 119613
xlli@i2r.a-star.edu.sg

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

See-Kiong Ng
Institute for Infocomm Research,
21 Heng Mui Keng Terrace,
Singapore, 119613
sckng@i2r.a-star.edu.sg

Abstract

Traditional classification involves building a classifier using labeled training examples from a set of predefined classes and then applying the classifier to classify test instances into the same set of classes. In practice, this paradigm can be problematic because the test data may contain instances that do not belong to any of the previously defined classes. Detecting such unexpected instances in the test set is an important issue in practice. The problem can be formulated as learning from positive and unlabeled examples (PU learning). However, current PU learning algorithms require a large proportion of negative instances in the unlabeled set to be effective. This paper proposes a novel technique to solve this problem in the text classification domain. The technique first generates a single artificial negative document A_N. The sets P and $\{A_N\}$ are then used to build a naïve Bayesian classifier. Our experiment results show that this method is significantly better than existing techniques.

1 Introduction

Classification is a well-studied problem in machine learning. Traditionally, to build a classifier, a user first collects a set of training examples that are labeled with predefined or known classes. A classification algorithm is then applied to the training data to build a classifier that is subsequently employed to assign the predefined classes to instances in a test set (for evaluation) or future instances (in practice).

This paradigm can be problematic in practice because some of the test or future instances may not belong to any of the predefined classes of the original training set. The test set may contain additional unknown subclasses, or new subclasses may arise as the underlying domain evolves over time. For example, in cancer classification, the training set consists of data from currently known cancer subtypes. However, since cancer is a complex and heterogeneous disease, and still a perplexing one to-date, it is likely that the test data contain cancer subtypes that are not yet medically classified (they are therefore not covered in the training data). Even if the training data do contain all the current cancer subtypes, new subtypes may be formed at a later stage as the disease evolves due to mutations or other cancer-causing agents. This phenomenon is not uncommon even in the seemingly simpler application domains. For example, in document classification, topics are often heterogeneous and new topics evolve over time. A document classifier built for classifying say, computer science papers, would face the similar problems as the cancer classifier described above. This is because computer science is a heterogeneous and increasingly cross-disciplinary domain; it is also a rapidly evolving one with new topics being created over time.

Thus, a classifier created based on the notion of a fixed set of predefined classes is bound to be inadequate in the complex and dynamic real-world in the long run, requiring the user to manually go through the classification results to remove the unexpected instances. In practice, a competent classifier should learn to identify unexpected instances in the test set so as to automatically set these unclassifiable instances apart. In some applications, this can be important in itself. For example, in the cancer example above, detection of the unexpected instances can alert the scientists that some new medical discovery (a new cancer subtype) may have occurred.

In recent years, researchers have studied the problem of learning from positive and unlabeled examples (or PU learning). Given a positive set P and an unlabelled set U, a PU learning algorithm learns a classifier that can identify hidden positive documents in the unlabeled set U. Our problem of identifying unexpected instances in the test set can be modeled as a PU learning problem by treating all the training data as the positive set P and the test set as the unlabeled set U. A classifier can then be learned using PU learning algorithms to classify the test set to identify those unexpected (or negative) instances before applying a traditional classifier to classify the remaining instances into the original predefined classes.

However, as the current PU techniques operate by trying to identify an adequate set of reliable negative data from the unlabeled set U to learn from, they require a large proportion of unexpected instances in the unlabeled set U to be effective. In practice, the number of unexpected instances in the test data can be very small since they are most likely to be arising from an emerging class. This means that the classifiers built with existing PU learning techniques will

perform poorly due to the small number of unexpected (negative) instances in U.

In this paper, we propose a novel technique called LGN (PU *L*earning by *G*enerating *N*egative examples), and we study the problem using text classification. LGN uses an entropy-based method to generate a single artificial negative document A_N based on the information in P and U, in which the features' frequency distributions correspond to the degrees of "negativeness" in terms of their respective entropy values. A more accurate classifier (we use the naïve Bayesian method) can be built to identify unexpected instances with the help of the artificial negative document A_N. Experimental results on the benchmark 20 Newsgroup data showed that LGN outperforms existing methods dramatically.

2 Related Work

PU learning was investigated by several researchers in recent years. A study of PAC learning from positive and unlabeled examples under the statistical query model was given in [Denis, 1998]. [Liu *et al.*, 2002] reported sample complexity results and showed how the problem may be solved.

Subsequently, a number of practical algorithms [Liu *et al.*, 2002; Yu *et al.*, 2002; Li and Liu, 2003] were proposed. They all conformed to the theoretical results in [Liu *et al.*, 2002] following a two-step strategy: (1) identifying a set of reliable negative documents from the unlabeled set; and (2) building a classifier using EM or SVM iteratively. Their specific differences in the two steps are as follows. S-EM proposed in [Liu *et al.*, 2002] is based on naïve Bayesian classification and the EM algorithm [Dempster, 1977]. The main idea was to first use a spying technique to identify some reliable negative documents from the unlabeled set, and then to run EM to build the final classifier. PEBL [Yu *et al.*, 2002] uses a different method (1-DNF) to identify reliable negative examples and then runs SVM iteratively to build a classifier.

More recently, [Li and Liu, 2003] reported a technique called Roc-SVM. In this technique, reliable negative documents are extracted by using the information retrieval technique Rocchio [Rocchio, 1971], and SVM is used in the second step. In [Fung *et al.*, 2005], a method called PN-SVM is proposed to deal with the situation when the positive set is small. All these existing methods require that the unlabeled set have a large number of hidden negative instances. In this paper, we deal with the opposite problem, i.e. the number of hidden negative instances is very small.

Another line of related work is learning from only positive data. In [Scholkopf, 1999], a one-class SVM is proposed. It was also studied in [Manevitz and Yousef, 2002] and [Crammer, 2004]. One-class SVM builds a classifier by treating the training data as the positive set P. Those instances in test set that are classified as negative by the classifier can be regarded as unexpected instances. However, our experiments show that its results are poorer than PU learning, which indicates that unlabeled data helps classification.

3 The Proposed Algorithm

Given a training set $\{c_i\}$ ($i = 1, 2, \ldots, n$) of multiple classes, our target is to automatically identify those unexpected instances in test set T that do not belong to any of the training classes c_i. In the next subsection (Section 3.1), we describe a baseline algorithm that directly applies PU learning techniques to identify unexpected instances. Then, in Section 3.2, we present our proposed LGN algorithm.

3.1 Baseline Algorithms: PU Learning

To recapitulate, our problem of identifying unexpected instances in the test set can be formulated as a PU learning problem as follows. The training instances of all classes are first combined to form the positive set P. The test set T then forms the unlabeled set U, which contains both positive instances (i.e., those belonging to training classes c_i) and negative/unexpected instances in T (i.e., those not belonging to any training class c_i). Then, PU learning techniques can be employed to build a classifier to classify the unlabeled set U (test set T) to identify negative instances in U (the unexpected instances). Figure 1 gives the detailed framework for generating baseline algorithms based on PU learning techniques.

1. $UE = \Phi$;
2. P = training examples from all classes (treated as positive);
3. $U = T$ (test set, ignore the class labels in T if present);
4. Run an existing PU learning algorithm with P and U to build a classifier Q;
5. **For** each instance $d_i \in U$ (which is the same as T)
6. Use a classifier Q to classify d_i
7. **If** d_i is classified as negative **then**
8. $UE = UE \cup \{d_i\}$;
9. output UE

Figure 1. Directly applying existing PU learning techniques

In the baseline algorithm, we use a set UE to store the negative (unexpected) instances identified. Step 1 initializes UE to the empty set, while Steps 2-3 initialize the positive set P and unlabeled set U as described above. In Step 4, we run an existing PU learning algorithm (various PU learning techniques can be applied to build different classifiers) to construct a classifier Q. We then employ the classifier Q to classify the test instances in U in Steps 5 to 8. Those instances that are classified by Q as negative class are added to UE as unexpected instances. After we have iterated through all the test instances, Step 9 outputs the unexpected set UE.

3.2 The Proposed Technique: LGN

In traditional classification, the training and test instances are drawn independently according to some fixed distribution D over $X \times Y$, where X denotes the set of possible documents in our text classification application, and $Y = \{c_1, c_2, ..., c_n\}$ denotes the known classes. Theoretically, for each class c_i, if its training and test instances follow the same distribution, a classifier learned from the training instances can be used to classify the test instances into the n known classes.

In our problem, the training set Tr with instances from classes $c_1, c_2, ..., c_n$ are still drawn from the distribution D. However, the test set T consists of two subsets, $T.P$ (called *positive instances* in T) and $T.N$ (called *unexpected / negative instances* in T). The instances in $T.P$ are independently drawn

from D, but the instances in $T.N$ are drawn from an unknown and different distribution D_u. Our objective is to identify all the instances drawn from this unknown distribution D_u, or in other words to identify all the hidden instances in $T.N$.

Let us now formally reformulate this problem as a two-class classification problem without labeled negative training examples. We first rename the training set Tr as the positive set P by changing every class label $c_i \in Y$ to "+" (the positive class). We then rename the test set T as the unlabeled set U, which comprises both hidden positive instances and hidden unexpected instances. The unexpected instances in U (or T) are now called *negative instances* with the class label "–" (bear in mind that there are many hidden positive instances in U). A learning algorithm will select a function f from a class of functions F: $X \rightarrow \{+, -\}$ to be used as a classifier that can identify the unexpected (negative) instances from U. The problem here is that there are no labeled negative examples for learning. Thus, it becomes a problem of *learning from positive and unlabeled examples* (PU learning). As discussed in the previous section, this problem has been studied by researchers in recent years, but existing PU techniques performed poorly when the number of negative (unexpected) instances in U is very small. To address this, we will propose a technique to generate artificial negative documents based on the given data.

Let us analyze the problem from a probabilistic point of view. In our text classification problem, documents are commonly represented by frequencies of words w_1, w_2, ..., $w_{|V|}$ that appear in the document collection, where V is called the *vocabulary*. Let w_+ represent a positive word feature that characterizes the instances in P and let w_- represent a negative feature that characterizes negative (unexpected) instances in U. If U contains a large proportion of positive instances, then the feature w_+ will have similar distribution in both P and U. However, for the negative feature w_-, its probability distributions in the set P and U are very different. Our strategy is to exploit this difference to generate an effective set of artificial negative documents N so that it can be used together with the positive set P for a classifier training to identify negative (unexpected) documents in U accurately.

Given that we use the naïve Bayesian framework in this work, before going further, we now introduce naïve Bayesian classifier for text classification.

NAÏVE BAYESIAN CLASSIFICATION

Naïve Bayesian (NB) classification has been shown to be an effective technique for text classification [Lewis, 1994; McCallum and Nigam, 1998]. Given a set of training documents D, each document is considered an ordered list of words. We use $w_{d_i,k}$ to denote the word in position k of document d_i, where each word is from the vocabulary $V = \{w_1, w_2, ..., w_{|V|}\}$. The vocabulary is the set of all words we consider for classification. We also have a set of predefined classes, $C = \{c_1, c_2, ..., c_{|C|}\}$. In order to perform classification, we need to compute the posterior probability, $Pr(c_j|d_i)$, where c_j is a class and d_i is a document. Based on the Bayesian probability and the 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 count of 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.

Finally, assuming that the probabilities of the words are independent given the class, we obtain the NB classifier:

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

In the naive Bayesian classifier, the class with the highest $Pr(c_j|d_i)$ is assigned as the class of the document.

GENERATING NEGATIVE DATA

In this subsection, we present our algorithm to generate the negative data. Given that in a naïve Bayesian framework, the conditional probabilities $Pr(w_t|-)$ (Equation (2)) are computed based on the accumulative frequencies of all the documents in the negative class, a single artificial negative instance A_N would work equally well for Bayesian learning. In other words, we need to generate the negative document A_N in such a way to ensure $Pr(w_+|+) - Pr(w_+|-) > 0$ for a positive feature w_+ and $Pr(w_-|+) - Pr(w_-|-) < 0$ for a negative feature w_-. We use an entropy-based method to estimate if a feature w_i in U has significantly different conditional probabilities in P and in U (i.e, ($Pr(w_i|+)$ and $Pr(w_i|-)$)). The entropy equation is:

$$entropy(w_i) = -\sum_{c \in \{+,-\}} \Pr(w_i \mid c) * \log(\Pr(w_i \mid c)) \quad (4)$$

The entropy values show the relative discriminatory power of the word features: the bigger a feature's entropy is, the more likely it has similar distributions in both P and U (i.e. less discriminatory). This means that for a negative feature w_-, its entropy $entropy(w_-)$ is small as $\Pr(w_-|-)$ (w_- mainly occurring in U) is significantly larger than $\Pr(w_-|+)$, while $entropy(w_+)$ is large as $\Pr(w_+|+)$ and $\Pr(w_+|-)$ are similar. The entropy (and its conditional probabilities) can therefore indicate whether a feature belongs to the positive or the negative class. We generate features for A_N based on the entropy information, weighted as follows:

$$q(w_i) = 1 - \frac{entropy(w_i)}{\max_{j=1,2...,|V|}(entropy(w_j))} \quad (5)$$

If $q(w_i)= 0$, it means that w_i uniformly occurs in both P and U and we therefore do not generate w_i in A_N. If $q(w_i) = 1$, we can be almost certain that w_i is a negative feature and we generate it for A_N, based on its distribution in U. In this way, those features that are deemed more discriminatory will be generated more frequently in A_N. For those features with $q(w_i)$ between the two extremes, their frequencies in A_N are generated proportionally.

We generate the artificial negative document A_N as follows. Given the positive set P and the unlabeled set U, we compute each word feature's entropy value. The feature's frequency in the negative document A_N is then randomly generated following a Gaussian distribution according to $q(w_i) = 1\text{-}entropy(w_i)/max(entropy(w_j), w_j \in V)$. The detailed algorithm is shown in Figure 2.

1. $A_N = \Phi$;
2. P = training documents from all classes (treated as positive);
3. $U = T$ (test set, ignore the class labels in T if present);
4. **For** each feature $w_i \in U$
5. Compute the frequency of w_i in each document d_k $freq(w_i, d_k)$, $d_k \in U$;
6. Let mean $\mu_{w_i} = \frac{\sum_{d_k \in D_{w_i}} freq(w_i, d_k)}{|D_{w_i}|}$, where D_{w_i} is the set of documents in containing w_i
7. Let variance $\sigma_{w_i}^2 = \frac{1}{(|D_{w_i}| - 1)} \sum_{d_k \in D_{w_i}} (freq(w_i, d_k) - \mu_{w_i})^2$;
8. **For** each feature $w_i \in V$
9. Compute $Pr(w_i|+)$, $Pr(w_i|-)$ using Equation (2) assuming that all the documents in U are negative;
10. Let $entropy(w_i) = -\sum_{c \in \{+,-\}} \Pr(w_i | c) * \log(\Pr(w_i | c))$;
11. Let $m = \max(entropy(w_j)), j = 1, ..., |V|$;
12. **For** each feature $w_i \in V$
13. $q(w_i) = 1 - \frac{entropy(w_i)}{m}$;
14. **For** $j = 1$ to $|D_{w_i}| * q(w_i)$
15. Generate a frequency $fnew(w_i)$, using the Gaussian distribution $\frac{1}{\mu_{w_i}\sqrt{2\pi}} e^{\frac{-(x-\mu_{w_i})^2}{2\sigma_{w_i}^2}}$
16. $A_N = A_N \cup \{(w_i, fnew(w_i))\}$
17. Output A_N

Figure 2. Generating the negative document A_N

In the algorithm, Step 1 initializes the negative document A_N (which consists of a set of feature-frequency pairs) to the empty set while Steps 2 and Step 3 initialize the positive set P and the unlabeled set U. From Step 4 to Step 7, for each feature w_i that appeared in U, we compute its frequency in each document, and then calculate the frequency mean and variance in those documents D_{w_i} that contain w_i. These information are used to generate A_N later. From Step 8 to Step 10, we compute the entropy of w_i using $Pr(w_i|+)$ and $Pr(w_i|-)$ (which are computed using Equation (2) by assuming that all the documents in U are negative). After obtaining the maximal entropy value in Step 11, we generate the negative document A_N in Steps 12 to 16. In particular, Step 13 computes $q(w_i)$, which shows how "negative" a feature w_i is in terms of how different the w_i's distributions in U and in P are the bigger the difference, the higher the frequency with which we generate the feature. Steps 14 to 16 is an inner loop and $|D_{w_i}| * q(w_i)$ decides the number of times we generate a frequency for word w_i. Thus, if $q(w_i)$ is small, it means that w_i has occurred in both P and U with similar probabilities, and we generate fewer w_i. Otherwise, w_i is quite likely to be a negative feature and we generate it with a distribution similar to the one in U. In each iteration, Step 15 uses a Gaussian distribution with corresponding μ_{w_i} and σ_{w_i} to generate a frequency $fnew(w_i)$ for w_i. Step 16 places the pair $(w_i, fnew(w_i))$ into the negative document A_N. Finally, Step 17 outputs our generated negative set A_N. Note that the frequency for each feature w_i in A_N may not of an integer value as it is generated by a Gaussian distribution. A_N is essentially a randomly generated aggregated document that summarizes the unlabelled data set, but with the features indicative of positive class dramatically reduced.

BUILDING THE FINAL NB CLASSIFIER

Finally, we describe how to build an NB classifier with the positive set P and the generated single negative document A_N to identify unexpected document instances. The detailed algorithm is shown in Figure 3.

1. $UE = \Phi$;
2. Build a naïve Bayesian classifier Q with P and $\{A_N\}$ using Equations (1) and (2);
3. **For** each document $d_i \in U$
4. Using Q to classify d_i using Equation (3);
5. **If** $(Pr(-|d_i) > Pr(+|d_i))$
6. $UE = UE \cup \{d_i\}$;
7. output UE;

Figure 3. Building the final NB classifier

UE stores the set of unexpected documents identified in U (or test set T), initialized to empty set in Step 1. In Step 2, we use Equations (1) and (2) to build a NB classifier by computing the prior probabilities Pr(+) and Pr(-), and the conditional probabilities of $Pr(w_i|+)$ and $Pr(w_i|-)$. Clearly, $Pr(w_i|+)$ and $Pr(w_i|-)$ can be computed based on the positive set P and the single negative document A_N respectively (A_N can be regarded as the average document of a set of virtual negative documents). However, the problem is how to compute the prior probabilities of Pr(+) and Pr(-). It turns out that this is not a major issue — we can simply assume that we have generated a negative document set that has the same number of documents as the number of documents in the positive set P. We will report experimental results that support this in the next section. After building the NB classifier Q, we use it to classify each test document in U (Steps 3-6). The final output is the UE set that stored all the identified unexpected documents in U.

4 Empirical Evaluation

In this section, we evaluate our proposed technique LGN. We compare it with both one-class SVM (OSVM, we use LIBSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/) and existing PU learning methods: S-EM [Liu *et al*., 2002], PEBL [Yu *et al*., 2002] and Roc-SVM [Li and Liu, 2003]. S-EM and Roc-SVM are publicly available[1]. We implemented PEBL as it is not available from its authors.

4.1 Datasets

For evaluation, we used the benchmark 20 Newsgroup collection, which consists of 11997 documents from 20 different UseNet discussion groups. The 20 groups were also categorized into 4 main categories, "computer", "recreation", "science", and "talk". We first perform the following two sets of experiments:

2-classes: This set of experiments simulates the case in which the training data has two classes, i.e. our positive set P contains two classes. The two classes of data were chosen

[1] http://www.cs.uic.edu/~liub/LPU/LPU-download.html

from two main categories, "computer" and "science", in which the "computer" group has five subgroups, and the "science" group has four subgroups. Every subgroup consists of 1,000 documents.

Each data set for training and testing is then constructed as follows: The positive documents for both training and testing consist of documents from one subgroup (or class) in "computer" and one subgroup (or class) in "science". This gives us 20 data sets. For each class (or subgroup), we partitioned its documents into two standard subsets: 70% for training and 30% for testing. That is, each positive set P for training contains 1400 documents of two classes, and each test set U contains 600 positive documents of the same two classes. We then add negative (unexpected) documents to U, which are randomly selected from the remaining 18 groups.

In order to create different experimental settings, we vary the number of unexpected documents, which is controlled by a parameter α, a percentage of $|U|$, i.e., the number of unexpected documents added to U is $\alpha \times |U|$.

3-classes: This set of experiments simulates the case in which the training data has three different classes, i.e. our positive set P contains three classes of data. We used the same 20 data sets formed above and added another class to each for both P and U. The added third class was randomly selected from the remaining 18 groups. For each data set, the unexpected documents in U were then randomly selected from the remaining 17 newsgroups. All other settings were the same as for the 2-classes case.

4.2 Experimental Results

2-classes: We performed experiments using all possible c_1 and c_2 combinations (i.e., 20 data sets). For each technique, namely, OSVM, S-EM, Roc-SVM, PEBL and LGN, we performed 5 random runs to obtain the average results. In each run, the training and test document sets from c_1 and c_2 as well as the unexpected document instances from the other 18 classes were selected randomly. We varied α from 5% to 100%. Table 1 shows the classification results of various techniques in terms of F-score (for negative class) when α = 5%. The first column of Table 1 lists the 20 different combinations of c_1 and c_2. Columns 2 to 5 show the results of four techniques OSVM, S-EM, Roc-SVM and PEBL respectively. Column 6 gives the corresponding results of our technique LGN.

We observe from Table 1 that LGN produces the best results consistently for all data sets, achieving an F-score of 77.0% on average, which is 54.8%, 32.8%, 60.2% and 76.5% higher than the F-scores of existing four techniques (OSVM, S-EM, Roc-SVM and PEBL) respectively in absolute terms. We also see that LGN is highly consistent across different data sets. In fact, we have checked the first step of the three existing PU learning techniques and found that most of the extracted negative documents were wrong. As a result, in their respective second steps, SVM and EM were unable to build accurate classifiers due to very noisy negative data. Since the S-EM algorithm has a parameter, we tried different values, but the results were similar.

Table 1. Experimental results for α = 5%.

Data Set	OSVM	S-EM	Roc-SVM	PEBL	LGN
graphic-crypt	22.5	46.3	17.2	0.0	**82.1**
graphic-electro	17.9	54.1	15.8	3.5	**78.0**
graphic-med	17.2	39.0	15.3	0.0	**64.9**
graphic-space	22.2	49.5	15.7	0.0	**71.7**
os-crypt	23.6	43.1	18.3	0.0	**82.8**
os-electronics	15.9	39.6	16.3	0.0	**80.2**
os-med	18.6	36.4	16.5	0.0	**75.2**
os-space	20.8	40.5	17.9	0.0	**78.2**
mac.hardware-crypt	23.1	46.0	17.5	1.2	**84.8**
mac.hardware-electro	18.8	42.4	17.5	0.0	**84.3**
mac.hardware-med	18.3	52.6	16.5	0.0	**70.4**
mac.hardware-space	21.3	40.0	17.5	0.0	**77.9**
ibm.hardware-crypt	25.4	46.5	16.9	0.0	**82.9**
ibm.hardware-electro	19.6	47.5	17.1	1.3	**82.4**
ibm.hardware-med	17.4	41.9	16.4	0.0	**74.5**
ibm.hardware-space	21.3	41.5	17.4	1.3	**75.5**
windows-crypt	22.6	54.1	17.3	2.3	**82.0**
windows-electro	19.4	48.2	16.0	0.0	**76.3**
windows-med	20.4	39.9	16.1	1.3	**66.2**
windows-space	18.3	34.4	16.4	0.0	**69.8**
Average	**20.2**	**44.2**	**16.8**	**0.5**	**77.0**

Figure 4 shows the macro-average results of all α values (from 5% to 100%) for all five techniques in the 2-classes experiments. Our method LGN outperformed all others significantly for $\alpha \leq$ 60%. When α was increased to 80% and 100%, Roc-SVM achieved slightly better results than LGN. We also observe that OSVM, S-EM and Roc-SVM outperformed PEBL since they were able to extract more reliable negatives than the 1-DNF method used in PEBL. PEBL needed a higher α (200%) to achieve similar good results.

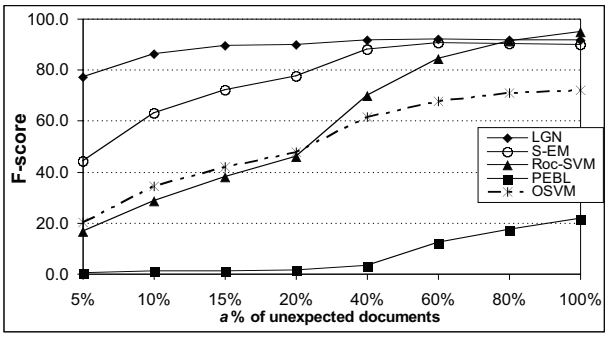

Figure. 4. The comparison results with different percentages of unexpected documents in U in the 2-classes experiments.

3-classes: Figure 5 shows the 3-classes results where LGN still performed much better than the methods when the proportion of unexpected documents is small ($\alpha \leq$ 60%) and comparably with S-EM and Roc-SVM when the proportion is larger. OSVM's results are much worse than S-EM, Roc-SVM and LGN when α is larger, showing that PU learning is better than one-class SVM in the problem. Again, PEBL required a much larger proportion of unexpected documents to produce comparable results.

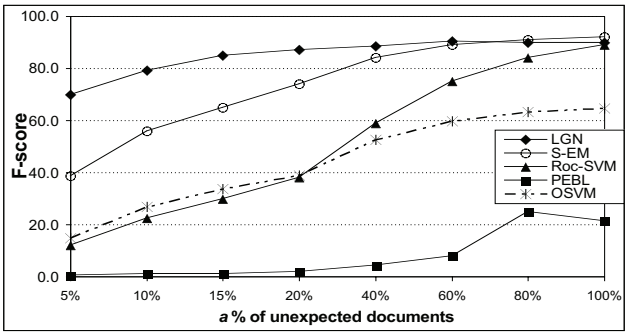

Figure. 5. The comparison results with different percentages of unexpected documents in U in the 3-classes experiments.

In summary, we conclude that LGN is significantly better (with high F-scores) than the other techniques when α is small ($\alpha \leq 60\%$), which indicates that it can be used to effectively extract unexpected documents from the test set even in the challenging scenarios in which their presence in U is non-obvious. The other methods all failed badly when α is small. LGN also performed comparably in the event when the proportion of unexpected instances is large ($\alpha \geq 80\%$).

Finally, we also conducted 10-classes experiments in which ten different classes from both the 20 Newsgroups and Reuter collections (with same experimental setting for the 3-classes) were used. The behaviors of the algorithms for 10 classes were the same as for 2 classes and 3 classes. Using the Reuter collection with 10 classes and α set to 5%, 10%, 15%, 20% and 40%, our algorithm LGN achieved 32.77%, 32.14%, 27.82%, 18.43%, 11.11% higher F-scores respectively than the best results of the existing methods (OSVM, S-EM, Roc-SVM and PEBL). Similarly, using 10 classes from the 20 newsgroup collection, LGN achieved 10.56%, 4.80%, 5.46%, 6.20%, and 4.00% higher F-scores for α =5%, 10%, 15%, 20% and 40% of unexpected documents respectively than the best of the four other existing methods.

Effect of priors: Recall that in Section 3 we have left the prior probabilities as a parameter since we only generate a single artificial negative document. To check the effect of priors, we also varied the prior in our experiments by changing the proportion of negative documents as a percentage of the number of positive documents in P. We tried 40%, 60%, 80% and 100%. The results were virtually the same, with average differences only within ±1%. Thus, we simply choose 100% as the default of our system, which gives us Pr(+) = Pr(-) = 0.5. All the experimental results reported here were obtained using this default setting.

5 Conclusion

In real-world classification applications, the test data may differ from the training data because unexpected instances that do not belong to any of the predefined classes may be present (or emerge in the long run) and they cannot be identified by traditional classification techniques. We have shown here that the problem can be addressed by formulating it as a PU learning problem. However, directly applying existing PU learning algorithms performed poorly as they require a large proportion of unexpected instances to be present in the unlabeled test data, which is often not the case in practice.

We then proposed a novel technique LGN to identify unexpected documents by generating a single artificial negative document to help train a classifier to better detect unexpected instances. Our experimental results in document classification demonstrate that LGN performed significantly better than existing techniques when the proportion of unexpected instances is low. The method is also robust irrespective of the proportions of unexpected instances present in the test set. Although our current experiments were performed in the text classification application using an NB classifier, we believe that the approach is also applicable to other domains. Using a single artificial negative document, however, will not suitable for other learning algorithms. In our future work, we plan to generate a large set of artificial documents so that other learning methods may also be applied.

References

[Crammer and Chechik, 2004] K. Crammer and G. Chechik. A needle in a haystack: local one-class optimization, ICML, 2004.

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

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

[Denis , 2002] F. Denis, R. Gilleron, and M. Tommasi. Text classification from positive and unlabeled examples. IPMU, 2002.

[Fung, 2005] G. Fung, J. Yu, H. Lu, and P Yu. Text Classification without Labeled Negative Documents. ICDE, 2005.

[Lewis and Gale, 1994] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. SIGIR, 1994.

[Li and Liu, 2003] X. Li, and B. Liu. Learning to classify text using positive and unlabeled data. IJCAI, 2003.

[Liu *et al*., 2002] B. Liu, W. Lee, P. Yu, and X. Li. Partially supervised classification of text documents. ICML, 2002.

[Manevitz and Yousef, 2001] L. Manevitz, and M. Yousef. One class SVMs for document classification. Journal of Machine Learning Research, 2, 139–154, 2001.

[McCallum and Nigam, 1998] A. McCallum, and K. Nigam, A comparison of event models for naïve Bayes text classification. AAAI, 1998.

[Muggleton, 2001] S. Muggleton. Learning from the positive data. Machine Learning, 2001.

[Rocchio, 1971] J. Rocchio. Relevant feedback in information retrieval. G. Salton. The smart retrieval system: experiments in automatic document processing, 1971.

[Scholkopf *et al*., 1999] B. Scholkopf, J. Platt, J. Shawe, A. Smola & R. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research, 1999.

[Yu *et al*., 2002] H. Yu, J. Han, and K. Chang. PEBL: Positive example based learning for Web page classification using SVM. KDD, 2002.