# Learning to Classify Documents with Only a Small Positive Training Set

Xiao-Li Li[1], Bing Liu[2], and See-Kiong Ng[1]

[1] Institute for Infocomm Research, Heng Mui Keng Terrace, 119613, Singapore
[2] Department of Computer Science, University of Illinois at Chicago, IL 60607-7053
xlli@i2r.a-star.edu.sg, liub@cs.uic.edu, skng@i2r.a-star.edu.sg

**Abstract.** Many real-world classification applications fall into the class of positive and unlabeled (PU) learning problems. In many such applications, not only could the negative training examples be missing, the number of positive examples available for learning may also be fairly limited due to the impracticality of hand-labeling a large number of training examples. Current PU learning techniques have focused mostly on identifying reliable negative instances from the unlabeled set *U*. In this paper, we address the oft-overlooked PU learning problem when the number of training examples in the positive set *P* is small. We propose a novel technique LPLP (Learning from Probabilistically Labeled Positive examples) and apply the approach to classify product pages from commercial websites. The experimental results demonstrate that our approach outperforms existing methods significantly, even in the challenging cases where the positive examples in *P* and the hidden positive examples in *U* were not drawn from the same distribution.

## 1 Introduction

Traditional supervised learning techniques typically require a large number of labeled examples to learn an accurate classifier. However, in practice, it can be an expensive and tedious process to obtain the class labels for large sets of training examples. One way to reduce the amount of labeled training data needed is to develop classification algorithms 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 and a set *U* of unlabeled examples (which contains both hidden positive and hidden negative examples), we build a classifier using *P* and *U* to classify the data in *U* as well as future test data. We call this the PU learning problem.

Several innovative techniques (e.g. [1], [2], [3]) have been proposed to solve the PU learning problem recently. All of these techniques have focused on addressing the lack of labeled negative examples in the training data. It was assumed that there was a sufficiently large set of positive training examples, and also that the positive examples in *P* and the hidden positive examples in *U* were drawn from the same distribution. However, in practice, obtaining a large number of positive examples can be rather difficult in many real applications. Oftentimes, we have to do with a fairly small set of positive training data. In fact, the small positive set may not even adequately represent the whole positive class, as it is highly likely that there could be hidden positives in *U* that may not be similar to those few examples in the small positive set *P*. Moreover, the examples in the positive set *P* and the hidden positive examples in the unlabeled set *U* may not even be generated or drawn from the same distribution. A classifier trained merely on the few available examples in *P* would thus be incompetent in recognizing all the hidden positives in *U* as well as those in the test sets.

In this work, we consider the problem of learning to classify documents with only a small positive training set. Our work was motivated by a real-life business intelligence application of classifying web pages of product information. The richness of information easily available on the World Wide Web has made it routine for companies to conduct business intelligence by searching the Internet for information on related products. For example, a company that sells computer printers may want to do a product comparison among the various printers currently available in the market. One can first collect sample printer pages by crawling through all product pages from a consolidated e-commerce web site (e.g., amazon.com) and then hand-label those pages containing printer product information to construct the set *P* of positive examples. Next, to get more product information, one can then crawl through all the product pages from other web sites (e.g., cnet.com) as *U*. Ideally, PU learning techniques can then be applied to classify all pages in *U* into printer pages and non-printer pages. However, we found that while the printer product pages from two websites (say, amazon.com and cnet.com) do indeed share many similarities, they can also be quite distinct as the different web sites invariably present their products (even similar ones) in different styles and have different focuses. As such, directly applying existing methods would give very poor results because 1) the small positive set *P* obtained from one site contained only tens of web pages (usually less than 30) and therefore do not adequately represent the whole positive class, and 2) the features from the positive examples in *P* and the hidden positive examples in *U* are not generated from the same distribution because they were from different web sites.

In this paper, we tackle the challenge of constructing a reliable document (web page) classifier based on only a few labeled positive pages from a single web site and then use it to automatically extract the hidden positive pages from different web sites (i.e. the unlabeled sets). We propose an effective technique called LPLP (Learning from Probabilistically Labeling Positive examples) to perform this task. Our proposed technique LPLP is based on probabilistically labeling training examples from *U* and the EM algorithm [4]. The experimental results showed that LPLP significantly outperformed existing PU learning methods.

## 2 Related Works

A theoretical study of PAC learning from positive and unlabeled examples under the statistical query model was first reported in [5]. Muggleton [6] followed by studying the problem in a Bayesian framework where the distribution of functions and examples are assumed known. [1] reported sample complexity results and provided theoretical elaborations on how the problem could be solved. Subsequently, a number of practical PU learning algorithms were proposed [1], [3] and [2]. These PU learning algorithms all conformed to the theoretical results presented in [1] by following a common two-step strategy, namely: (1) identifying a set of reliable negative documents from the unlabeled set; and then (2) building a classifier using EM or SVM iteratively. The specific differences between the various algorithms in these two steps are as follows. The S-EM method proposed in [1] was based on naïve Bayesian classification and the EM algorithm [4]. 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. The PEBL method [3] uses a different method (1-DNF) for identifying reliable negative examples and then runs SVM iteratively for classifier building. More recently, [2] reported a technique called Roc-SVM. In this technique, reliable negative documents were extracted by using the information retrieval technique Rocchio [7]. Again, SVM is used in the second step. A classifier selection criterion is also proposed to catch a good classifier from iterations of SVM. Despite the differences in algorithmic details, the above methods all focused on extracting reliable negative instances from the unlabeled set.

More related to our current work was the recent work by Yu [8], which proposed to estimate the boundary for the positive class. However, the amount of positive examples they required was around 30% of the whole data, which was still too large for many practical applications. In [9], a method called PN-SVM was proposed to deal with the case when the positive set is small. However, it (like all the other existing algorithms of PU learning) relied on the assumption that the positive examples in $P$ and the hidden positives in $U$ were all generated from the same distribution. For the first time, our LPLP method proposed in this paper will address such common weaknesses of current PU learning methods, including handling challenging cases where the positive examples in $P$ and the hidden positive examples in $U$ were not drawn from the same distribution.

Note that the problem could potentially be modeled as a one-class classification problem. For example, in [10], a one-class SVM that uses only positive data to build a SVM classifier was proposed. Such approaches are different from our method in that they do not use unlabeled data for training. However, as previous results reported in [2] have already showed that they were inferior for text classification, we do not consider them in this work.

## 3 The Proposed Technique

Figure 1 depicts the general scenario of PU learning with a small positive training set $P$. Let us denote a space $\psi$ that represents the whole positive class, which is located above the hyperplane $H_2$. The small positive set $P$ only occupies a relatively small subspace $SP$ in $\psi$ ($SP \subseteq \psi$), shown as the oval region in the figure. The examples in the unlabelled set $U$ consists of both hidden positive examples (represented by circled "+") and hidden negative examples (represented by "-"). Since $P$ is small, and $\psi$ contains positive examples from different web sites that present similar products in different styles and focuses, we may not expect the distributions of the positive examples in $P$ and those of the hidden positive examples in $U$ to be the same. In other words, the set of hidden positive examples that we are trying to detect may have a very small intersection or is even disjoint with $SP$. If we naively use the small set $P$ as the positive training set and the entire unlabelled set $U$ as the negative set, the resulting classifier (corresponds to hyperplane $H_1$) will obviously perform badly in identifying the hidden positive examples in *U*.

On the other hand, we can attempt to use the more sophisticated PU learning methods to address this problem. Instead of merely using the entire unlabelled set *U* as the negative training data, the first step of PU learning extracts some reliable negatives from *U*. However, this step is actually rather difficult in our application scenario as depicted in Figure 1. Since the hidden positive examples in *U* are likely to have different distributions from those captured in the small positive set *P*, not all the training examples in *U* that are dissimilar to examples in *P* are negative examples. As a result, the so-called reliable negative set that the first step of PU learning extracts based on dissimilarity from *P* would be a very noisy negative set, and therefore not very useful for building a reliable classifier.

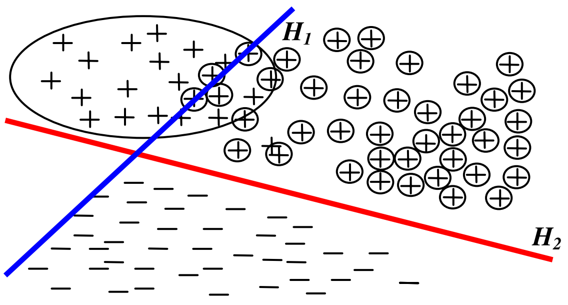

**Fig. 1.** PU learning with a small positive training set

Let us consider the possibility of extracting a set of likely positive examples (*LP*) from the unlabeled set *U* to address the problem of *P*'s being not sufficiently representative of the hidden positive documents in *U*. Unlike *P*, the distribution of *LP* will be similar with the other hidden positive examples in *U*. As such, we could expect that a more accurate classifier can be built with the help of set *LP* (together with *P*). Pictorially, the resulting classifier would correspond to the optimal hyperplane $H_2$ shown in Figure 1. Instead of trying to identify a set of noisy negative documents from the unlabeled set *U* as existing PU learning techniques do, our proposed technique LPLP therefore focuses on extracting a set of likely positive documents from *U*.

While the positive documents in *P* and the hidden positive documents in *U* were not drawn from the same distribution, they should still be similar in some underlying feature dimensions (or subspaces) as they belong to the same class. For example, the printer pages from two different sites, say amazon.com and cnet.com, would share the representative word features such as "printer", "inkjet", "laser", "ppm" etc, though their respective distributions may be quite different. Particularly, the pages from cnet.com whose target readers are more technically-savvy may contain more frequent mentioning of keyword terms that correspond to the technical specifications of printers than those pages from amazon.com whose primary focus is to reach out to the less technically-inclined customers. However, we can safely expect that the basic keywords (representative word features) that describe computer printers should be presented in both cases. In this work, we therefore assume that the representative word features for the documents in *P* should be similar to those for the hidden positive documents in *U*. If we can find such a set of representative word features (*RW*) from the positive set *P*, then we can use them to extract other hidden positive documents from *U*.

We are now ready to present the details of our proposed technique LPLP. In Section 3.1, we first introduce a method to select the set of representative word features *RW* from the given positive set *P*. Then, in Section 3.2, we extract the likely positive documents from *U* and probabilistically label them based on the set *RW*. With the help of the resulting set *LP*, we employ the EM algorithm with a good initialization to build an accurate classifier to identify the hidden positive examples from *U* in Section 3.3.

## 3.1 Selecting a Set of Representative Word Features from P

As mentioned above, we expect the positive examples in *P* and the hidden positive examples in *U* share the same representative word features as they belong to the same class. We extract a set *RW* of representative word features from the positive set *P* containing the top *k* words with the highest $s(w_i)$ scores. The scoring function *s*() is based on TFIDF method [11] which gives high scores to those words that occur frequently in the positive set *P* and not in the whole corpus $P \bigcup U$ since *U* contains many other unrelated documents. Figure 2 shows the detailed algorithm to select the keyword set *RW*.

```
1. $RW = \Phi$, $F = \Phi$;
2. For each word feature $w_i \in P$
3.     If (stopwords( $w_i$ )!=true)
4.         $F = F \cup$ {stemming( $w_i$ )};
5. $total$=0;
6. For each word feature $w_i \in F$
7.     $N(w_i, P) = \sum_{j=1}^{|P|} \frac{N(w_i, d_j)}{\max_{w_i}(N(w_i, d_j))}$;
8.     $total$ += $N(w_i, P)$;
9. For each word feature $w_i \in F$
10.    $s(w_i) = \frac{N(w_i,P)}{total} * \log \frac{|P|+|U|}{df(w_i,P)+df(w_i,U)}$;
11. Rank the word feature's $s(w_i)$ from big to small into a list $L$;
12. Pr$_{TOP}$= the $k$-th $s(w_i)$ in the list $L$, $w_i \in F$;
13. For $\forall w_i \in F$
14.     If ( $s(w_i)$ >=Pr$_{TOP}$)
15.         $RW = RW \cup \{ w_i \}$;
```

**Fig. 2.** Selecting a set of representative word features from *P*

In step 1, we initialize the representative word set *RW* and unique feature set *F* as empty sets. After removing the stop words (step 3) and performing stemming (step 4) [11], all the word features are stored into the feature set *F*. For each word feature $w_i$ in the positive set *P*, steps 6 to 8 compute the accumulated word frequency (in each document $d_j$, the word $w_i$'s frequency $N(w_i,d_j)$ is normalized by the maximal word frequency of $d_j$ in step 7). Steps 9 to 10 then compute the scores of each word feature, which consider both $w_i$'s probabilities of belonging to a positive class and its inverted document frequency, where $df(w_i, P)$ and $df(w_i, U)$ are $w_i$'s document frequencies in *P* and *U* respectively. After ranking the scores into the rank list *L* in step 11, we store into *RW* those word features from *P* with top *k* scores in *L*.

## 3.2 Identifying *LP* from *U* and Probabilistically Labeling the Documents in *LP*

Once the set *RW* of representative keywords is determined, we can regard them together as a representative document (*rd*) of the positive class. We then compare the similarity of each document $d_i$ in *U* with *rd* using the cosine similarity metric [11], which automatically produces a set *LP* of probabilistically labeled documents with $\Pr(d_i|+) > 0$. The algorithm for this step is given in Figure 3. In step 1, the likely positive set *LP* and the remaining unlabelled set *RU* are both initialized as empty sets. In steps 2 to 3, each unlabeled document $d_i$ in *U* is compared with *rd* using the cosine similarity. Step 4 stores the largest similarity value as *m*. For each document $d_i$ in *U*, if its cosine similarity $sim(rd, d_i)>0$, we assign a probability $Pr(+|d_i)$ that is based on the ratio of its similarity and *m* (step 7) and we store it into the set *LP* in step 8. Otherwise, $d_i$ is included in *RU* instead (step 10). The documents in *RU* have zero similarity with *rd* and can be considered as a purer negative set than *U*.

```
1. $LP = \Phi$; $RU = \Phi$;
2. For each $d_i \in U$
3.     $sim(rd, d_i) = \frac{\sum_j w_{rd,j} * w_{d_i,j}}{\sqrt{\sum_j w_{rd,j}^2}\sqrt{\sum_j w_{d_i,j}^2}}$;
4. Let $m = \max_{d_i}(sim(rd, d_i))$, $d_i \in U$;
5. For each $d_i \in U$
6.     If ($sim(rd, d_i) > 0$)
7.         $Pr(+|d_i) = sim(rd, d_i)/m$;
8.         $LP = LP \cup \{d_i\}$;
9.     Else
10.        $RU = RU \cup \{d_i\}$;
```

**Fig. 3.** Probabilistically labeling a set of documents

Note that in step 7, the hidden positive examples in *LP* will be assigned high probabilities while the negative examples in *LP* will be assigned very low probabilities. This is because the representative features in *RW* were chosen based on those words that occurred frequently in *P* but not in the whole corpus $P \cup U$. As such, the hidden positive examples in *LP* should also contain many of the features in *RW* while the negative examples in *LP* would contain few (if any) of the features in *RW*.

## 3.3 The Classification Algorithm

Next, we employ the naïve Bayesian framework to identify the hidden positives in *U*. Given a set of training documents *D*, each document is considered an list of words and each word in a document is from the vocabulary $V = < w_1, w_2, \ldots, w_{|v|} >$. We also have a set of predefined classes, $C=\{c_1, c_2, \ldots, c_{|C|}\}$ For simplicity, we will consider two class classification in this discussion, i.e. $C=\{c_1, c_2\}$, $c_1$="+" and $c_2$="-". To perform classification for a document $d_i$, we need to compute the posterior probability, $Pr(c_j|d_i)$, $c_j \in \{+,-\}$. Based on the Bayesian probability and the multinomial model [12], we have

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

and with Laplacian smoothing,

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

Here, $Pr(c_j|d_i) \in \{0,1\}$ depending on the class label of the document.

Assuming that the probabilities of words are independent given the class, we obtain the naïve Bayesian classifier:

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

In the naive Bayesian classifier, the class with the highest $Pr(c_j|d_i)$ is assigned as the class of the document. The NB method is known to be an effective technique for text classification even with the violation of some of its basic assumptions (e.g class conditional independence) [13] [14] [1].

The Expectation-Maximization (EM) algorithm is a popular class of iterative algorithms for problems with incomplete data. It iterates over two basic steps, the Expectation step, and the Maximization step. The Expectation step basically fills in the missing data, while the Maximization step then estimates the parameters. When applying EM, equations (1) and (2) above comprise the Expectation step, while equation (3) is used for the Maximization step. Note that the probability of the class given the document now takes the value in [0, 1] instead of {0, 1}.

```
1. For each $d_i \in RU$,
2.     $Pr(+ \mid d_i) = 0$;
3.     $Pr(- \mid d_i) = 1$;
4. $PS = LP \cup P$ (or $LP$);
5. For each $d_j \in PS$
6.     If $d_j \in P$
7.         $Pr(+ \mid d_j) = 1$;
8.         $Pr(- \mid d_j) = 0$;
9.     Else
10.        $Pr(+ \mid d_j) = sim(rd, d_j) / m$;
11.        $Pr(- \mid d_j) = 0$;
12. Build an NB-C classifier C using PS and RU based on equations (1), (2);
13. While classifier parameters change
14.     For each $d_i \in PS \cup RU$
15.         Compute $Pr(+|d_i)$ and $Pr(-|d_i)$ using NB-C, i.e., equation (3);
16.         Update $Pr(c_j)$ and $Pr(w_t|c_j)$ by replacing equations (1) and (2)
            with the new probabilities produced in step 15 (a new NB-C is being
            built in the process)
```

**Fig. 4.** The detailed LPLP algorithm

The ability of EM to work with missing data is exactly what we need here. Let us regard all the positive documents to have the positive class value "+". We want to know the class value of each document in the unlabeled set *U*. EM can help to properly assign a probabilistic class label to each document $d_i$ in the unlabeled set, i.e., $Pr(+|d_i)$ or $Pr(-|d_i)$. Theoretically, in EM algorithm, the probabilities of documents in *U* will converge after a number of iterations [4]. However, a good initialization is important in order to find a good maximum of the likelihood function. For example, if we directly use *P* as positive class and *U* as negative class (initially), then EM algorithm will not build an accurate classifier as the negative class would be too noisy (as explained previously). Thus, in our algorithm, after extracting likely positive set *LP*, we re-initialize the EM algorithm by treating the probabilistically labeled *LP* (with/without *P*) as positive documents. The resulting classifier is more accurate because 1) *LP* has the similar distributions with other hidden positive documents in *U*, and 2) the remaining unlabeled set *RU* is also much purer than *U* as a negative set.

The detailed LPLP algorithm is given in Figure 4. The inputs to the algorithm are *LP*, *RU* and *P*. Steps 1 to 3 initialize the probabilities for each document $d_i$ in *RU*, which are all treated as negative documents initially. Step 4 sets the positive set *PS*; there are two possible ways to achieve this: we either (1) combine *LP* and *P* as *PS*, or (2) use only *LP* as *PS*. We will evaluate the effect of the inclusion of *P* in *PS* in the next section. Steps 5 to 11 will assign the initial probabilities to the documents in *P* (if *P* is used) and *LP*. Each document in *P* is assigned to the positive class while each document in *LP* is probabilistically labeled using the algorithm in Figure 3. Using *PS* and *RU*, a NB classifier can be built (step 12). This classifier is then applied to the documents in ($LP \cup RU$) to obtain the posterior probabilities ($Pr(+|d_i)$ and $Pr(-|d_i)$) for each document (step 15). We can then iteratively employ the revised posterior probabilities to build a new (and better) NB classifier (step 16). The EM process continues until the parameters of the NB classifier converge.

## 4 Evaluation Experiments

In this section, we evaluate the proposed LPLP technique under different settings and compare it with existing methods, namely, Roc-SVM [2] and PEBL [3]. Roc-SVM is available on the Web as a part of the LPU system[1]. We implemented PEBL ourselves as it is not publicly available. The results of S-EM [1] were not included here because the performance was generally very poor due to its reliance on similarity of positive documents in *P* and in *U,* as expected.

### 4.1 Datasets

Our evaluation experiments were done using product Web pages from 5 commercial Web sites: Amazon, CNet, PCMag, J&R and ZDnet. These sites contained many introduction/description pages of different types of products. The pages were cleaned using the web page cleaning technique in [15], i.e., navigation links and advertisements have been detected and eliminated. The data contained web pages of the following product categories: Notebook, Digital Camera, Mobile Phone, Printer and TV. Table 1 lists the number of pages from each site, and the corresponding product categories (or classes). In this work, we treat each page as a text document and we do not use hyperlinks and images for classification.

**Table 1.** Number of Web pages and their classes.

| | **Amazon** | **CNet** | **J&R** | **PCMag** | **ZDnet** |
|---|---|---|---|---|---|
| **Notebook** | 434 | 480 | 51 | 144 | 143 |
| **Camera** | 402 | 219 | 80 | 137 | 151 |
| **Mobile** | 45 | 109 | 9 | 43 | 97 |
| **Printer** | 767 | 500 | 104 | 107 | 80 |
| **TV** | 719 | 449 | 199 | 0 | 0 |

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

Note that we did not use standard text collections such as Reuters[2] and 20 Newsgroup data[3] in our experiments as we want to highlight the performance of our approach on data sets that have different positive distributions in *P* and in *U*.

## 4.2 Experiment Settings

As mentioned, the number of available positively labeled documents in practice can be quite small either because there were few documents to start with, or it was tedious and expensive to hand-label the training examples on a large scale. To reflect this constraint, we experimented with different number of (randomly selected) positive documents in *P*, i.e. $|P|$ = 5, 15, or 25 and allpos. Here "allpos" means that all documents of a particular product from a Web site were used. The purpose of these experiments is to investigate the relative effectiveness of our proposed method for both small and large positive sets.

We conducted a comprehensive set of experiments using all the possible *P* and *U* combinations. That is, we selected every entry (one type of product from each Web site) in Table 1 as the positive set *P* and use each of the other 4 Web sites as the unlabeled set *U*. Three products were omitted in our experiments because their $|P|<10$, namely, Mobile phone in J&R (9 pages), TV in PCMag (no page), and TV in ZDnet (no page). A grand total of 88 experiments were conducted using all the possible *P* and *U* combinations of the 5 sites. Due to the large number of combinations, the results reported below are the average values for the results from all the combinations.

To study the sensitivity of the number of representative word features used in identifying likely positive examples, we also performed a series of experiments using different numbers of representative features, i.e. $k$ = 5, 10, 15 and 20 in our algorithm.

## 4.3 Experimental Results

Since our task is to identify or retrieve positive documents from the unlabeled set *U*, it is appropriate to use *F* value to evaluate the performance of the final classifier. *F* value is the harmonic mean of precision (*p*) and recall (*r*), i.e. $F=2*p*r/(p+r)$. When either of *p* or *r* is small, the *F* value will be small. Only when both of them are large, *F* value will be large. This is suitable for our purpose as we do not want to identify positive documents with either too small a precision or too small a recall. Note that in our experiment results, the reported *F* values give the classification (retrieval) results of the positive documents in *U* as *U* is also the test set.

Let us first show the results of our proposed technique LPLP under different experimental settings. We will then compare it with two existing techniques.

The bar chart in Figure 5 shows the *F* values (Y-axis) of LPLP using different numbers of positive documents (X-axis) and different numbers of representative words (4 data series). Recall that we had presented two options to construct the positive set *PS* in step 4 of the LPLP algorithm (Figure 4). The first option is to add the extracted likely positive documents (*LP*) to the original set of positive documents *P*, represented in Figure 5 by "with *P*". The second option is to use only the extracted likely positive documents as the positive data in learning, i.e., dropping the original

[2] http://www.research.att.com/~lewis/reuters21578.html
[3] http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes/20_newsgroups.tar.gz

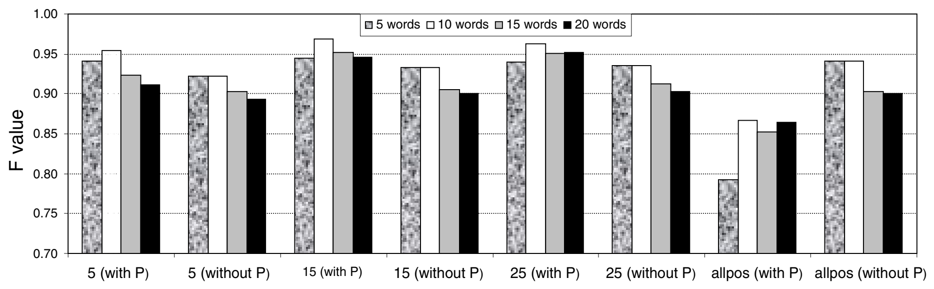

**Fig. 5.** F values of LPLP with different numbers of positive documents

positive set *P* (since it is not representative of the hidden positive documents in *U*). This option is denoted by "without P" in Figure 5.

**Inclusion of *P* for constructing *PS*:** If there were only a small number of positive documents (|*P*| = 5, 15 and 25) available, we found that using option 1 (with *P*) to construct the positive set for the classifier is better than using option 2 (without *P*), as expected. However, interestingly, if there is a large number of positive documents (allpos in Figure 5), then option 1 is actually inferior to option 2. The reason is that the use of a big positive set *P*, which is not representative of the positive documents in *U*, would have introduced too much negative influence on the final classifier (many hidden positive examples in *U* will be classified as negative class). However, when the given positive set *P* is small, its potential negative influence is much less, and it will therefore help to strengthen the likely positive documents by providing more positive data. This is a subtle and rather unexpected trade-off here.

**Number of positive documents in *P*:** From Figure 5, we also observe that the number of the given positive documents in *P* does not influence the final results a great deal. The reason for this is that the computed likely positive documents from *U* are actually more effective positive documents for learning than the original positive documents in *P*. This is a very compelling advantage of our proposed technique as this means that the user does not need to label or to find a large number of positive examples for effective learning. In fact, as discussed above, we also notice that even without using any original positive document in *P*, the results were very good as well.

**Number of representative word features:** The results in Figure 5 also showed that there is no need to use many representative words for detecting positive documents. In general, 5-15 representative words would suffice. Including the less representative word features beyond the top *k* most representative ones would introduce unnecessary noise in identifying the likely positive documents in *U*.

Next, we compare the results of our LPLP technique with those of the two best existing techniques mentioned earlier, namely, Roc-SVM [2] and PEBL [3]. Figure 6 shows two series of results. The first series, marked "*P*", showed the classification results of all three methods using all positive documents ("allpos"), without the use of the likely positive documents *LP* as suggested in this paper. In other words, learning was done using only *P* and *U*. Note that for the strictest comparison what we have shown here are the best possible results for Roc-SVM and PEBL, which may not be obtainable in practice because it is hardly possible to determine which SVM iteration would give the best results in these algorithms (both Roc-SVM and PEBL algorithms run SVM many times). In fact, their results at convergence were actually much worse.

We can see that PEBL performed better than both LPLP and Roc-SVM. However, the absolute *F* value of PEBL is still very low (0.54). Note also that because of the use of "allpos" for training, the LPLP's result for this was obtained without using the likely positive set *LP* (it is now the EM standard algorithm), hence it was unable to perform as well as it should have.

The second series in Figure 6 shows the comparative results of using the extracted likely positive documents instead of *P* for learning. Here, our LPLP algorithm performs dramatically better (*F*=0.94) even against the best possible results of PEBL (*F*=0.84) and Roc-SVM (*F*=0.81). Note that here PEBL and Roc-SVM also use the likely positive documents *LP* extracted from *U* by our method (we boosted the PEBL and Roc-SVM for the purpose of comparison). The likely positives were identified from *U* using 10 representative words selected from *P*. Unlike our LPLP algorithm, both Roc-SVM and PEBL do not take probabilistically labels, but only binary labels. As such, for these two algorithms, we chose the likely positive documents from *U* by requiring each document (*d*) to contain at least 5 (out of 10) selected representative words. All the likely positive documents identified were then treated as positive documents, i.e., $Pr(+|d) = 1$. We also tried using other numbers of representative words in *RW* and found that 5 words performed well for these two algorithms with our datasets. We can see that with the use of the likely positives (set *LP*) identified by our method (instead of *P*), the classification results of these two existing algorithms have also improved dramatically as well. In fact, by using *LP* instead of *P*, the previously weaker Roc-SVM has caught up so substantially that the best possible result of PEBL is only slightly better than that of Roc-SVM now.

Finally, in Figure 7, we show the comparative results when the number of the positive documents is small, which is more often than not the case in practice. Again, we see that our new method LPLP performed much better than the best possible

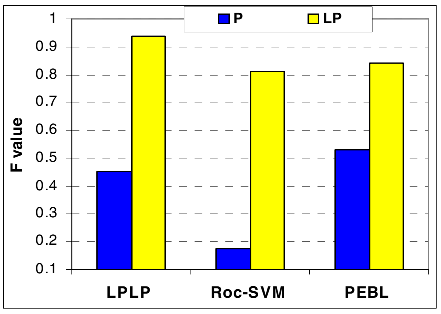

**Fig. 6.** F values of LPLP, the best results of Roc-SVM and PEBL using all positive documents

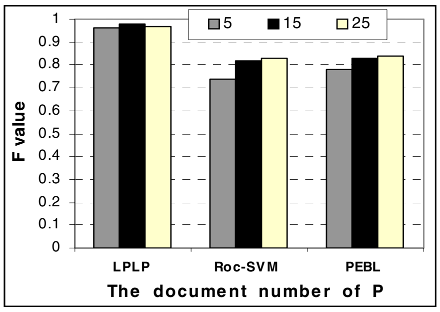

**Fig. 7.** F values of LPLP, the best results of Roc-SVM and PEBL using *P* together with *LP*

results of the two existing methods Roc-SVM and PEBL (which may not be obtainable in practice, as explained earlier) when there were only 5, 15, or 25 positive documents in *P*. As explained earlier, including *P* together with *LP* (for all the three techniques) gave better results when *P* is small.

In summary, the results in Figures 6 and 7 showed that the likely positive documents *LP* extracted from *U* can be used to help boost the performance of classification techniques for PU learning problems. In particular, LPLP algorithm benefited the most and performed the best. This is probably because of its ability to handle probabilistic labels and is thus better equipped to take advantage of the probabilistic (and hence potentially noisy) *LP* set than the SVM-based approaches.

## 5 Conclusions

In many real-world classification applications, it is often the case that not only the negative training examples are hard to come by, but the number of positive examples available for learning can also be fairly limited as it is often tedious and expensive to hand-label large amounts of training data. To address the lack of negative examples, many PU learning methods have been proposed to learn from a pool of positive data (*P*) without any negative data but with the help of unlabeled data (*U*). However, PU learning methods still do not work well when the size of positive examples is small.

In this paper, we address this oft-overlooked issue for PU learning when the number of positive examples is quite small. In addition, we consider the challenging case where the positive examples in *P* and the hidden positive examples in *U* may not even be drawn from the same distribution. Existing techniques have been found to perform poorly in this setting. We proposed an effective technique LPLP that can learn effectively from positive and unlabeled examples with a small positive set for document classification. Instead of identifying a set of reliable negative documents from the unlabeled set *U* as existing PU techniques do, our new method focuses on extracting a set of likely positive documents from *U*. In this way, the learning can rely less on the limitations associated with the original positive set *P*, such as its limited size and potential distribution differences. Augmented by the extracted probabilistic *LP* set, our LPLP algorithm can build a much more robust classifier. We reported experimental results with product page classification that confirmed that our new technique is indeed much more effective than existing methods in this challenging classification problem. In our future work, we plan to generalize our current approach to solve similar classification problems other than document classification.

## References

1. Liu, B., Lee, W., Yu, P., Li, X.: Partially Supervised Classification of Text Documents. In: ICML, pp. 387–394 (2002)
2. Li, X., Liu, B.: Learning to Classify Texts Using Positive and Unlabeled Data. In: IJCAI, pp. 587–594 (2003)
3. Yu, H., Han, J., Chang, K.C.-C.: PEBL: Positive Example Based Learning for Web Page Classification Using SVM. In: KDD, pp. 239–248 (2002)

4. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society (1977)
5. Denis, F.: PAC Learning from Positive Statistical Queries. In: ALT, pp. 112–126 (1998)
6. Muggleton, S.: Learning from Positive Data. In: Proceedings of the sixth International Workshop on Inductive Logic Programming, pp. 358–376. Springer, Heidelberg (1997)
7. Rocchio, J.: Relevance feedback in information retrieval. In: Salton, G. (ed.) The SMART Retrieval System: Experiments in Automatic Document Processing (1971)
8. Yu, H.: General MC: Estimating boundary of positive class from small positive data. In: ICDM, pp. 693–696 (2003)
9. Fung, G.P.C., et al.: Text Classification without Negative Examples Revisit. IEEE Transactions on Knowledge and Data Engineering 18(1), 6–20 (2006)
10. Schölkopf, B., et al.: Estimating the Support of a High-Dimensional Distribution. Neural Comput 13(7), 1443–1471 (2001)
11. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval (1986)
12. McCallum, A., Nigam, K.: A comparison of event models for naïve Bayes text classification. In: AAAI Workshop on Learning for Text Categorization (1998)
13. Lewis, D.D.: A sequential algorithm for training text classifiers: corrigendum and additional data. In: SIGIR Forum, 13–19 (1995)
14. Nigam, K., et al.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning 39(2-3), 103–134 (2000)
15. Yi, L., Liu, B., Li, X.: Eliminating noisy information in Web pages for data mining. In: KDD, pp. 296–305 (2003)