On Positive and Unlabeled Learning for Text Classification*

István Nagy T.[1], Richárd Farkas[2], and János Csirik[3]

[1] University of Szeged, Department of Informatics,
6720 Szeged, Árpád tér 2., Hungary
[2] Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung,
Azenbergstrasse 12, D-70174 Stuttgart, Germany
[3] MTA-SZTE Research Group on Artificial Intelligence,
6720 Szeged, Tisza Lajos krt. 103., Hungary
{nistvan,jcsirik}@inf.u-szeged.hu, farkas@ims.uni-stuttgart.de

Abstract. In this paper we present a slightly modified machine learning approach for text classification working exclusively from positive and unlabeled samples. Our method can assure that the positive class is not underrepresented during the iterative training process and it can achieve 30% better F-value when the amount of positive examples is low.

Keywords: semi-supervised learning, positive and unlabeled, PU, text classification

1 Introduction

Classification is a well-studied problem in machine learning. Text classification is the process of assigning predefined category labels to new documents. The classic supervised approach to build a text classifier is to first manually label a set of training documents, which are labeled with the same set of predefined category or class labels as the test set. A classification algorithm is then applied to the training data to build a classifier, which is subsequently employed to assign the predefined classes to instances in the test set (for evaluation) or future instances (in practice).

The main bottleneck of supervised learning is that it needs a large number of labeled training examples for building the accurate model. However, labeling is typically done manually, which is – on the one hand – labor intensive and time consuming, and – on the other hand – may lead to unexpected complications. For example, in practice some of the test or future instances may not belong to any of the predefined classes of the original training set. Furthermore, the test set may contain additional unknown subclasses, or new subclasses may arise as the underlying domain evolves over time. Manual annotation cannot be effectively prepared for these cases. Collecting negative training examples is

* This work was supported in part by the National Innovation Office of the Hungarian government within the framework of the projects BELAMI and MASZEKER.

especially delicate and arduous because negative training examples must be uniformly represented in the universal set excluding the positive class. On the other hand, manually collected negative training examples could be biased because of humans' unintentional prejudice, which could be detrimental to classification accuracy.

In recent years, researchers have studied the concept of using only a small labeled set and a large unlabeled set to help learning. This approach is called semi-supervised learning. It reduces the effort of manual labeling, but negative examples are also needed. The Positive and Unlabeled (PU) learning is a specific semi-supervised learning method. PU learning approaches only need a positive set P and an unlabelled set U, then the algorithm can identify hidden positive documents in the unlabeled set.

Although there are several PU implementations, we were motivated to investigate them in detail because many real life problems can be successfully solved by PU approaches. For example, a company that tries to attract new customers with direct marketing owns a database of their own customers, but has no information on those who are not their customers. In this case, they only have positive examples but no negative examples. If they buy a database containing data on people, they can find people who are similar to their customers, and then can deliberately seek out the bids. In this paper we focus on the test classification use case which is usually used as a sandbox for PU algorithms.

2 Related work

Traditionally, PU learning algorithms are based on a strategy of two steps: first, identify a set of reliable negative (RN) documents from the unlabeled set by using any method for this. Second, build a classifier based on positive and reliable negative data from the unlabeled set. The specific difference between the various algorithms in these two steps is as follows: the 1-DNF or PEBL algorithm [11] first collects words that occur in the positive set P more frequently than in the unlabelled set U. This method has several versions with some modifications. One of them improved the original 1-DNF algorithm [12], which requires the absolute frequency of the feature in the positive data set greater than $\alpha\%$. As a result, they had a smaller but better quality positive word set, which yields a much larger reliable negative document set. The other popular approach to identify reliable negative examples in U is the Spy technique [4]. It is based on the method that first randomly selects a set S of positive documents from P and puts them in U. These documents are called "Spies". They behave similarly to the unknown documents in U. Hence, the algorithm can identify the behavior of the unknown positive documents in U more easily. The Rocchio algorithm is also a popular text classificaton method [4, 2]. In this technique, reliable negative documents are extracted by using the information retrieval method Rocchio. This approach constructs a prototype vector for every class. The classifier is then used to classify documents in U. Those documents that are classified as negative are considered (reliable) negative data, denoted by RN. In the second

step, a classifier is built using expectation-maximization (EM) or Support vector machine (SVM) iteratively.

3 The Proposed Technique

The most popular PU learning algorithms apply a common two-step strategy. We examined the two steps separately. The key element in the first step is the quality of reliable negative documents. To investigate this, we used error rate as follows [12] :

$$Err(\%) = \frac{\#positive\_examples\_in\_RN}{\#positive\_examples\_in\_U}$$

We found that several approaches are good at identifying reliable negative examples from the unlabelled data set. This is proved by the fact that the error rate was less than 1% when we applied Rocchio. Thus, the second step seemed to be more interesting to explore. In the common second step SVM or EM machine learning algorithms were used. Basically, we investigated the iterative SVM method. In the basic iterative SVM method, P and RN were first used to train SVM. Let Q be the remaining unlabeled document set: Q = U - RN. In each iteration we used the actual SVM model to classify Q. Documents which the model classifies as negative were put in RN. The main idea of this approach is that SVM can extract a greater number of possible negative examples in Q. If the current model could not mark any more documents from Q as negative, the algorithm terminated. After the iterations, the final classifier is applied. However, sometimes it proves to be necessary to select a classifier since SVM is sensitive to noise. It may happen that an iteration of SVM extracts too many positive documents from Q and puts them in RN, which may have a negative effect on the performance of the last SVM classifier.

```
while true
 use P and RN to train SVM classifier;
 classify Q using SVM model;
 let W be the set of documents that a current SVM model classified as negative;
 if W =  ∅ then
  exit;
 else
  Q = Q - W;
  RN = RN ∪ W
```

Fig. 1. The two-step approach of [11].

However, we wanted to see how other classification algorithms can perform. In all cases, Rocchio was applied as the first step. In most cases, this approach could identify a reliable negative set which is large enough with minimum error rate.

3.1 The Rocchio Technique

In Rocchio classification, each document is represented as a vector in [7]. Each element in the vector was weighted in term frequency-inverse document frequency (tf-idf). This weight measures how important a word is to a document in a corpus. A classifier is built by constructing positive and negative prototype vectors. In classification, for each test document, it simply uses the cosine measure [7] to compute the similarity of the test document to each prototype vector. The class whose prototype vector is most similar to the test document is assigned to the test document. Documents classified as negative form the negative set RN.

3.2 The Proposed Technique

As [3] emphasized, catching the best iteration during the iteration running process is an important question. The current systems increase the size of only the RN set in the second step. In this case, since only a small positive set is used, in the last iterations the RN set becomes much larger than P, yielding that the positive class is underrepresented and the negative class is overrepresented. Thus, the model may classify all examples as negative, or learn the negative class. To solve this, we increased the P set too. Since the algorithms that we applied could determine the example class probability, they just accepted the prediction in case the prediction probability was higher than α value. Finally we used $\alpha = 0.9$. As shown in Figure 2 during each iteration, both the P and the RN class were increased.

```
while true
 use P and RN to train classifier;
 classify Q using a model;
 let W be the set of documents that a current model classified as negative;
 let S be the set of documents that a current model classified as positive;
 if W = ∅ then
  exit;
 else
  Q = Q - W - S;
  RN = RN ∪ W;
  P = P ∪ S;
```

Fig. 2. The modified iterative method.

Classifiers

In our experiments we used the implementations available in the WEKA [10] library, an open-source data mining software written in Java.

Boosting is [8] a way of improving the performance of a weak learning algorithm. The algorithm first generates a set of classifiers of the same type by applying bootstrapping on the original training data set. These classifiers vote a decision. The final decision is made using a weighted voting schema for each classifier. The resulting model is many times more accurate than the original. Some iterations of Boosting were performed on each model. Further iterations gave only slight improvement in the F-measure (less than 0.05%), thus we decided to perform only 10 iterations in each experiment.

C4.5 is based on the well-known ID3 tree learning algorithm [6]. Axis-parallel hyperplanes are used in classification, and hence learning is very fast. We built decision trees that had at least 2 instances per leaf, and used pruning with subtree raising and a confidence factor of 0.25.

Support Vector Machines (SVM) [9] is the linear function of the form $f(x) = w^t x + b$. Between the weight vector w and the input vector x, $w^t x$ denotes the inner products. SVM is based on the main idea of selecting the hyperplane that separates the space (between the positive and negative classes) while maximizing the smallest margin. In practice we used libSvm[4] and the Weka **SMO** implementation.

Logistic Regression is a predictive model. It was used when the target variable is a categorical variable with two categories. The logistic model formula computes the probability of the selected response as a function of the values of the predictor variables.

4 Experiments and Results

In our experiments, we used the Reuters-21,578 dataset, which has 21,578 documents collected from the Reuters newswire, as our training and testing sample set. Of the 135 categories in Reuters-21,758, only the 10 most frequent ones are used. PU approaches were evaluated in different ways. On the one hand, we used the inductive evaluation: the model identifies or retrieves positive documents from the unlabeled set U. On the other hand, the model was evaluated on the test set with unknown examples. In both evaluation methods one category was employed as the positive class, and the other nine classes as the negative and each category fulfilled once the role of the positive class. For each category,

[4] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

p%	Roc-SVM	SVM	SVM_B	C4.5	C4.5_B	LogReg	LogReg_B	SMO	SMO_B
0.05	**0.30195**	0.0707	0.4130	0.5017	**0.6591**	0.4084	0.4034	0.6141	0.6280
0.1	**0.68731**	0.3727	0.6641	0.7029	0.7463	0.4187	0.4062	**0.7611**	0.7600
0.15	**0.76898**	0.5311	0.7583	0.7300	**0.8089**	0.4082	0.4453	0.7941	0.7936
0.2	**0.79846**	0.7014	0.7727	0.7871	**0.8372**	0.4431	0.4336	0.7892	0.7911
0.3	**0.82053**	0.7932	**0.8145**	0.8027	0.8052	0.4486	0.4449	0.8072	0.8038
0.4	**0.8314**	0.8271	**0.8343**	0.8219	0.8228	0.4657	0.4600	0.8157	0.8161
0.45	**0.82432**	0.8222	**0.8460**	0.8012	0.8174	0.4935	0.4633	0.8074	0.7989
0.5	**0.80254**	0.8188	0.8198	0.8213	**0.8294**	0.5246	0.5262	0.7841	0.7864

Table 1. The modified second step with different learning algorithms on the Reuters-21,578 dataset (F-value results). Roc-SVM: the online available PU learning algorithm, SVM: during the iteration process in the modified second step, libSVM was used as learning algorithm. SVM_B: boosted version at SVM. C4.5: decision tree was used in the second step. C4.5_B: boosted version at C4.5. LogReg: Logistic Regression was used in the second step. LogReg_B: boosted version at Logistic Regression. SMO: SVM implementation in Weka was used in the second step. SMO_B: boosted version at SMO.

30% of the documents were randomly selected as test documents. The rest was used to create the training sets as follows: γ of the documents from the positive class are first selected as the positive set P. The rest of the positive documents (1-γ) and negative documents are used as the unlabeled set U. In the inductive evaluation method the test set was added to U. Since we would like to compare our results with [2] and [4] we determined the values of γ as 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.45 and 0.5. In order to compare our results with existing methods, we evaluated the Roc-SVM approach too. It is available on the Web as part of the LPU system[5]. To evaluate the performance of the classifier we used F-value. It is the harmonic mean of precision and recall.

Table 1 shows the result achieved by different learning algorithms. As the table shows, in the case of the libSVM and the C4.5 algorithms, boosting was able to improve the results. However, in the case of SMO and LogReg, boosting was not effective. Since the boosted C4.5 algorithm achieved the best results in low γ rate, we compare our modified second step method with this learning algorithm with the ROC-SVM. So, we evaluate and compare these two methods in two different ways. Results are shown in Figures 3 and 4.

As the figures show, the boosted C4.5 algorithm with the modified second step achieved better results when γ was low. The biggest difference between the two approaches could be observed when the positive rate was the lowest (0.05). In this case our method could achieve an F-value about 30% better than Roc-SVM. As the positive rate grows, this advantage declines. If the rate of the positive elements is higher than 30%, the two approaches perform nearly identically.

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

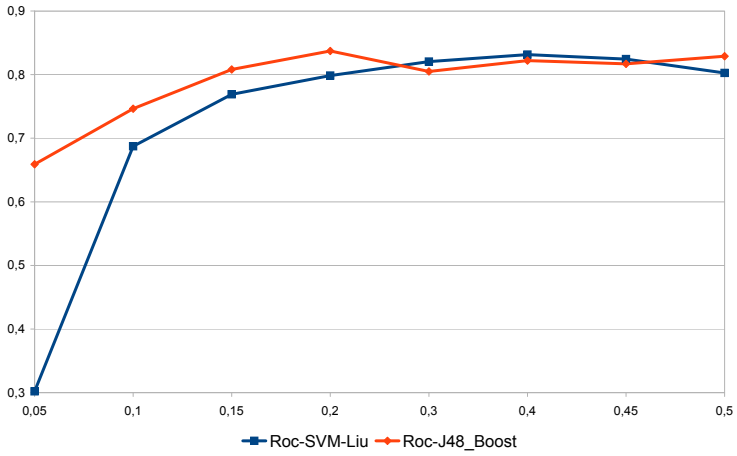

Fig. 3. The result of Roc-SVM and our modified second step with boosted C4.5 learning algorithm in inductive evaluation.

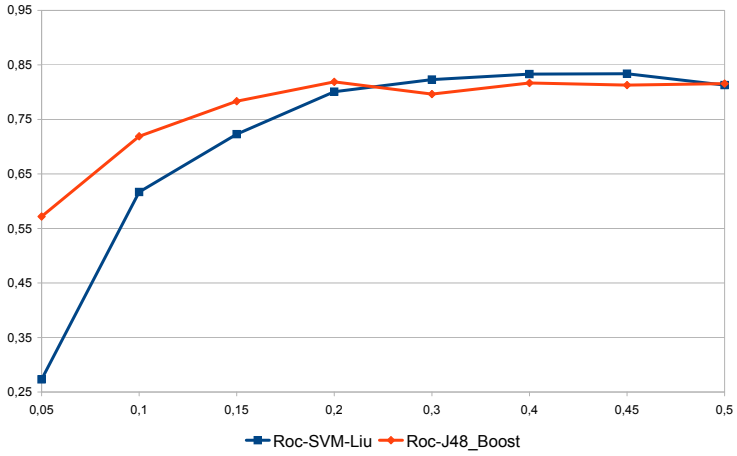

Fig. 4. Evaluating on a separate test set of Roc-SVM and our modified second step with boosted C4.5 learning algorithm.

5 Conclusion

The common PU algorithms are based on a two step strategy. We found that the Rocchio approach could effectively solve the first step. Therefore, we investigated the second step. To improve the iterations we modified the common second step: if the learning algorithm classifies an unlabeled element as positive we put it the positive set P, yielding that the positive class is not underrepresented during the iteration process. The common PU learning algorithms used SVM or EM iteratively. We examined some other learning algorithms, and we found that the boosted C4.5 learning approach achieved more than 30% better F-value when the positive rate was low.

References

1. Agresti, A.: Building and applying logistic regression models. In: An Introduction to Categorical Data Analysis. Wiley, pp. 137–172 (2007)
2. Li, X., Liu, B.: Learning to classify text using positive and unlabeled data. In: Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03). Acapulco, Mexico, Aug 9-15 (2003)
3. Li, X., Liu, B., Ng, S.-K.: Negative Training Data can be Harmful to Text Classification. In: Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-10). MIT, Massachusetts, USA (2010)
4. Liu, B., Dai, Y., Li, X., Lee, W., Yu, P.: Building text classifiers using positive and unlabeled examples. In: Proceedings of the Third IEEE International Conference on Data Mining (ICDM-03). Melbourne, Florida, November 19-22 (2003)
5. Liu, B., Li, X., Lee, W.S., Yu, P.S.: Text Classification by Labeling Words. In: AAAI 2004 (2004)
6. Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann Publishers (1993)
7. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill (1983)
8. Schapire, R.E.: The Strength of Weak Learnability. Machine Learning 5(2), 197–227 (1990)
9. Vapnik, V.N.: Statistical Learning Theory. Wiley (1998)
10. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)
11. Yu, H., Han, J., Chang, K.C.C.: PEBL: Positive Example-Based learning for web page classification using SVM. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 239–248. ACM (2002)
12. Yu, H., Zuo, W., Peng, T.: A New PU Learning Algorithm for Text Classification. In: Proceedings of MICAI, pp. 824–832 (2005)