TPN^2: Using positive-only learning to deal with the heterogeneity of labeled and unlabeled data

Nikolaos Trogkanis [1], Georgios Paliouras [2]

[1] School of Electrical and Computer Engineering, NTUA, Greece
tronikos@gmail.com
[2] Inst. of Informatics and Telecommunications, NCSR "Demokritos", Greece
paliourg@iit.demokritos.gr

Abstract. This paper introduces TPN^2, the runner up method in both tasks of the ECML-PKDD Discovery Challenge 2006 on personalized spam filtering. TPN^2 is a classifier training method that bootstraps positive-only learning with fully-supervised learning, in order to make the most of labeled and unlabeled data, under the assumption that the two are drawn from significantly different distributions. Furthermore, the unlabeled data themselves are separated into subsets that are assumed to be drawn from multiple distributions. For that reason, TPN^2 trains a different classifier for each subset, making use of all unlabeled data each time.

Keywords: one-class learning, positive-only learning, semi-supervised learning, multi-strategy learning, spam filtering

1 Introduction

The topic of the ECML-PKDD Discovery Challenge 2006 was personalized spam filtering. The goal was to train a personalized spam/ham classifier for each user that correctly classifies the emails in the user's inbox. Despite their personalization, it was assumed that the classifiers will be trained and used on the mailing server. Therefore, training cannot rely on messages labeled by the individual users. An obvious surrogate for training data is the use of publicly available sources, such as mailing lists and newsgroups and emails received through "spam traps"[1]. Such data have been used for benchmarking spam filters in the past, e.g. the Ling-spam[2] corpus. In addition to these labeled data, the personal emails of the users are assumed to be available for training, but without labels. These unlabeled data are available in large volumes and can be used to improve the classifiers in a semi-supervised fashion.

In semi-supervised learning a small set of labeled examples of every class and a large unlabeled set are used for building the classifier. Semi-supervised learning has been shown to be particularly beneficial in training text classifiers, such as spam

[1] Spam traps are email addresses published visually invisible for humans but get collected by the web crawlers of spammers.
[2] Ling-spam is available at: http://www.iit.demokritos.gr/skel/i-config/downloads/

filters, e.g. [1], [3], [9]. However, all of these techniques assume that labeled and unlabeled examples are generated from the same distribution. This assumption may be violated in practice, and when this happens these methods perform poorly.

One such example is the ECML-PKDD 2006 competition, where the labeled public data are very different from the emails received by individual users. Clearly, the unlabeled data are closer to the data expected to be processed by the filter in operation. Therefore, their use is even more important than in the usual semi-supervised learning scenario. At the same time, the use of the labeled data is essential, but has to be done with care, in order to avoid misleading the training process.

Addressing this dual problem, we chose to rely mostly on the large amounts of unlabeled emails in the user's inboxes. We used the labeled training data only to help us label a small part of the unlabeled data, sufficient to bootstrap the semi-supervised learning process. In doing that, we also took into account a natural asymmetry between the spam and the ham classes, namely that spam is much less personal than ham. Therefore, the discrepancy between the labeled and the unlabeled data is expected to be much higher for ham than for spam. Thus, we named spam the positive class and applied a positive-only learning approach on the unlabeled data.

To solve the problem of learning from positive and unlabeled examples, a few algorithms have been proposed in the past few years. One class of algorithms is based on a two-step strategy. This class includes SEM (Spy Expectation Maximization) [6], PEBL (Positive Example Based Learning) [11] and Roc-SVM (Rocchio – Support Vector Machines) [4]. These algorithms aim to iteratively discover the true negative examples, while maintaining correctly-classified the positive ones. It has been shown theoretically that this approach can lead to a good classifier [6]. In addition to these two-step algorithms, there are other methods that aim to estimate the proportion of negative to positive examples in unlabeled data and use that to bias the training process, e.g. PNB (Positive Naive Bayes) [2] and biased-SVM [5].

One final aspect of our approach was the utilization of multiple different subsets of unlabeled data. These subsets correspond to the inboxes of different users. Clearly the inbox of a user should weigh more in the training of that user's personalized filter. However, the inboxes of other users can also provide useful information. For that reason, we performed a weighted aggregation of the inboxes, giving more weight to the inbox of the current user. The inbox of the current user is taken as "foreground" and the other inboxes as "background" data. The weight of foreground data varied according to the dissimilarity of the user's inbox from other inboxes.

In summary the contribution of our TPN^2 method comprises:

- the use of fully-supervised learning to bootstrap positive-only learning on data from a different distribution;
- the weighted aggregation of foreground and background unlabeled data.

The rest of the paper is organized as follows. After presenting in section 2 the algorithms that we used, in section 3 we present the TPN^2 method. Then, in section 4, we empirically evaluate the proposed technique on the two tasks of the ECML-PKDD Discovery Challenge 2006. Finally, we provide our conclusions from this work, together with suggestions for future work in section 5.

2 Description of Existing Algorithms

As explained in section 1, the proposed method combines both fully-supervised and semi-supervised learning. In the fully-supervised stage, the common Naive Bayes algorithm, following the multinomial model was used, while in the semi-supervised stages a version of PNB (Positive Naive Bayes) was combined with PEBL (Positive Example Based Learning) and Roc-SVM (Rocchio – Support Vector Machines). These four algorithms are presented briefly in this section.

2.1 Naive Bayes Multinomial (NBM)

Given a set D of labeled documents, let us denote by PD (respectively ND) the set of positive documents (respectively negative documents) in the set D. Considering bag-of-words representation of the documents, each document is represented as the vector $d = \langle x_t \rangle_{t=1..|V|}$, where $|V|$ the size of the vector of features X_t taking values x_t. Each feature corresponds to a word and the value that it takes in a vector is a function of the number of occurrences of the word in document d. In the simplest case, the feature function indicates only the presence or absence of a word from a document. In that case the document vector contains binary features.

Bayes classifiers assign an unclassified document to the most probable class, using Bayes theorem:

$$\arg\max_j \{p(c_j \mid d)\} = \arg\max \{p(c_j)\, p(d \mid c_j)\} \quad (1)$$

Naive Bayes calculates the required a-posteriori and a-priori probabilities as frequencies on the training data, under the simplifying assumption that the probability of a document given a class can be expressed as the product of the individual probabilities of its feature values x_t given the class. In other words, features are assumed to be independent given the class.

According to [7], there are two models of the Naive Bayes classifier that are mostly used for text classification. These are the multi-variate Bernoulli and the multinomial. Despite its initial use for handling word frequencies, the multinomial model has recently been shown to perform better than the multi-variate Bernoulli even when ignoring frequencies and translating the document vectors into binary ones, e.g. [10] and [8]. For this reason, we have opted for the multinomial model, which elaborates equation 1 as follows:

$$\arg\max_j \left\{ p\left(c_j \mid d\right) \right\} = \arg\max_j \left\{ p\left(c_j\right) p\left(|d|\right) |d|! \prod_{t=1}^{|V|} \frac{p\left(w_t \mid c_j\right)^{x_t}}{x_t!} \right\} \quad (2)$$

where class and word probability estimates were calculated as frequencies on the training data, using Laplace smoothing to avoid zero probabilities. We have tested the method with both types of document vector, i.e., frequencies and binary, and arrived at similar conclusions to what has been reported in the literature, i.e. that binary document vectors lead to better performance. Therefore, we focus on binary vectors in the rest of the paper.

2.2 Naive Bayes Multinomial Positive (Positive Naive Bayes – PNB)

The second algorithm that we used is a representative of the second class of positive-only learning methods mentioned in section 1, i.e. those that estimate the proportion of negative to positive examples in unlabeled data and use that to bias the training process. In particular, we adopt the approach proposed in [2] for PNB (Positive Naive Bayes), using the multinomial model for Naive Bayes.

According to PNB, we assume to be given an estimate $\hat{p}(pos)$ of the positive class probability $p(pos)$, a set *PD* of positive documents together with a set *UD* of unlabeled documents, the Naive Bayes Multinomial Positive classifier classifies a document $d = \langle x_t \rangle_{t=1...|V|}$ as explained in section 2.1, calculating the class probabilities as follows:

$$p(c_0 \equiv pos) = \hat{p}(pos), \; p(c_1 \equiv neg) = 1 - \hat{p}(pos) \quad (3)$$

and the word probability estimates, using Laplace smoothing as follows:

$$p(w_t \mid pos) = \frac{1 + \#(w_t, PD)}{|V| + \#(PD)} \quad (4)$$

$$p(w_t \mid neg) = \frac{\hat{p}(w_t) - p(w_t \mid pos) \cdot pr(pos)}{1 - pr(pos)} \quad (5)$$

where $\hat{p}(w_t) = \frac{\#(w_t, UD)}{\#(UD)}$.

2.3 Positive Example Based Learning (PEBL)

PNB was combined with two positive-only learners belonging in the first class mentioned in section 1, i.e. those that iteratively search for the true negative examples in the unlabeled ones, while maintaining correctly-classified the positive examples. The first of the two algorithms that we tested was PEBL [11], which adopts the following strategy:

1. identify a set of reliable negative documents from the unlabeled set (strong negative);
2. apply a common classifier learning algorithm, such as SVM, on the positive and the strong negative to obtain a classifier;
3. apply the classifier on the unseen data that are not in the strong-negative set;
4. add the new negatives to the strong-negative set and retrain the SVM;
5. stop the iterative process when no new negatives are found by the classifier and consider the remaining examples as positive.

In the first step, i.e. the one that identifies the first set of strong negatives, PEBL uses the 1-DNF method. This method rejects any unlabeled examples that contain words that appear very commonly in the positive examples. Clearly, this is a very strict criterion, but also one that reduces the chances of mislabeling positive examples as negative in the first step.

2.4 Roc-SVM

The second of the two-step algorithms that we used was Roc-SVM [4], which follows a similar approach to PEBL, consisting of two steps: (1) extracting some reliable negative documents from the unlabeled set, (2) applying SVM iteratively to build a classifier.

In the first step Roc-SVM uses a more elaborate method than PEBL, a Rocchio classifier is built on the positive and unlabeled data, assuming that all unlabeled are negative, and then this classifier is applied on the unlabeled data to identify strong negative. The resulting labeled data are used to train the SVM classifier in step 2, following the same iterative procedure as in PEBL.

The second difference of Roc-SVM to PEBL is that it does not trust the final SVM in the iterative process to be the best-trained one, as this classifier is often affected by noise. Instead it chooses either that, or the one produced in the first iteration, depending on how well the final one classifies the positive examples. The SVM produced in the first iteration is often a very good one, due to the way in which the strong negative examples are chosen.

3 Proposed Method

The TPN^2 method addresses the problem of training a classifier in the presence of some labeled and many unlabeled data derived from different distributions. An example of that is the use of labeled public email and personal mailboxes in the Challenge. The method deals with this problem, by training a fully-supervised classifier on the labeled data and using that to select a small number of good positive examples in the unlabeled. These strong positives are used to bootstrap a positive-only learner. An implicit assumption made here is that the positive examples are less different in the labeled and the unlabeled data than the negative ones. This is likely to be the case with spam (positive) vs. ham (negative) emails.

The unlabeled data may also comprise a number of similar but different subsets, such as the mailboxes of different users. In order to make the best use of the unlabeled data, TPN^2 trains a separate classifier for each subset, e.g. each user, using at the same time all unlabeled data. However, it weighs the current subset more, treating that as "foreground" data, while treating the rest of the unlabeled data as "background" data.

Our method consists of four stages: (1) creating a weighted mixture of the different subsets of unlabeled data, (2) training a fully-supervised classifier to select the strongest positive from the unlabeled examples, (3) iteratively extend the positive set with a positive-only learner, and (4) using a two-step positive-only learner to refine the final classifier. The four stages are described in more detail below, while the pseudocode for TPN^2 is presented in table 1.

Stage 1: Weighted mixture of foreground and background unlabeled data
In this stage the set of unlabeled data is weighted. The method focuses on one of the distinct subsets in the dataset, treating that as foreground data and awarding its members with w times more weight than the rest of the unlabeled (background) data.

This is implemented by simply using each example of the foreground data w times, instead of just once. The value of the weight w is user-defined, but we will show a heuristic method for choosing it at the end of this section.

Stage 2: Selection of strong positives from the unlabeled examples
In this stage, fully-supervised learning is used to train a classifier on the labeled data. After experimentation, we chose the Naive Bayes classifier, using the multinomial model (NBM) for this purpose. Once the classifier is trained, it is applied on the unlabeled data, allowing us to choose a small number of strong positive examples. Strong positives are the examples classified as positive by NBM with confidence greater than a user-defined threshold.

Stage 3: Iterative extension of the strong positive set
Given a set of good positive examples and many unlabeled ones, we apply a positive-only learner (PNB) to identify more positive. We assume here that the labeled data have provided more information about the positive class and there are thus very few false positives among the selected set of strong positive examples.[3] Using the initial set of strong positive examples, PNB builds a classifier that is applied on all unlabeled examples. Those examples that are classified as positive will make the new positive set, which is used in turn by PNB to build a new classifier. At the end of stage 3, we will have a positive set containing most of the positive examples of the unlabeled set and very few false positives.

Stage 4: Refine the positive-only trained classifier
Having identified most of the positive examples, we refine the classifier using a different positive-only learner that focuses on finding strong negative examples. We have tested both PEBL and Roc-SVM in that role.

The algorithm presented in table 1, has three parameters that need to be defined by the user:

1. Positive class probability p. This was provided for the challenge and it is p=0.5.
2. Confidence threshold for NBM, above which a positive example is considered strong positive. Given the fact that Naive Bayes tends to push probabilities estimates to 0 and 1, we have opted for a strict value for this threshold, i.e. 0.99.
3. Foreground data weight w. For the selection of this parameter the following heuristic is proposed: Test for increasing values of w and keep the lowest value that leads to the maximum number of identified positive emails in E_i just before entering the final stage. This heuristic pushes the assumption of minimum false positive rate to the extreme.

[3] This was actually proven when we were given the true labels of the challenge data. About half of the messages identified as ham by the initial classifier were false.

Table 1. Pseudocode description of the TPN[2] method.

Input:

- labeled training emails, T
- unlabeled subsets, E_1, …, E_n
- foreground subset, E_i
- foreground weight, w
- positive (spam) class probability, p

Output:

- classifier

Algorithm:

$E := \sum_{j \neq i} E_j + w \cdot E_i$; // ...(1)

```
NBM := construct_NBM (T); // ...(2)
POS := NBM.classify(E); // ...(3)
do {
        POS_OLD := POS;
        U := E – POS; // ...(4)
        PNB := construct_PNB(POS, U, p); // ...(5)
        POS := PNB.classify(E); // ...(6)
} while (POS ≠ POS_OLD); // …(7)
U := E – POS;
PEBL := construct_PEBL(POS, U); // ...(8)
return PEBL;
```

Notes:

…(1) E is a weighted mixture of all unlabeled data (the foreground data E_i is added w times)
…(2) Naive Bayes Multinomial (NBM) learns from T
…(3) NBM is used to extract the strongest positive examples from the unlabeled ones
…(4) remove the strong positives from the unlabeled examples
…(5) train PNB on strong positives and remaining unlabeled, using positive class probability[4]
…(6) use the trained PNB to classify all unlabeled and keep the positive
…(7) continue iteratively, until no more positive can be found
…(8) run PEBL (or Roc-SVM) on positive and remaining unlabeled

[4] In the version of the algorithm that participated in the challenge, we used a more pessimistic estimate of the positive class probability for PNB. The use of p, as shown here, led to considerably better results than all of the reported results in the challenge. We would like to thank the reviewer of the paper for this simplifying suggestion.

4 Experimental Results

4.1 Experimental Set-up

This section uses the Challenge data to study the behavior of TPN^2 under varying conditions and parameter values. In particular we wanted to study:

1. The performance of TPN^2 in the two different tasks of the challenge. Table 2 presents the main properties of the two tasks. In Task A, the size of labeled training data from public corpora is large and so is the size of unlabeled data per user. The aim here is to be able to train a personalized filter from each user's data separately. In contrast, Task B requires the use of information from the unlabeled data of other users. The labeled data is very limited and so is the number of emails available for each user. Presumably, the users share enough common characteristics to be able to utilize unlabeled data from all inboxes when training a personalized filter.
2. The choice of value for the parameter w, i.e. the relative weight of the user's own data (foreground data) to the rest of the unlabeled data (background data). This value is expected to be smaller and closer to 1, the closer the user's data are to the norm.
3. The effect of the various training stages and corresponding algorithms that we used.

Table 2. Number of emails and inboxes for each task of the challenge.

	Task A	Task B
Number of labeled training emails	4000	100
Number of emails within one evaluation inbox	2500	400
Number of inboxes for evaluation	3	15

The data was provided in feature vector format and therefore the use of text analysis or heuristics that are commonly used in spam filtering was not possible. The performance of the methods was assessed by the AUC (Area Under Curve) method, which measures the area under the ROC (Receiver Operating Characteristics) curve. The ROC curve is usually a plot of sensitivity against 1-specifity. In this case it was a plot of the true positive rate (correctly identified spam) against the false positive rate (incorrectly identified spam).

4.2 Choosing the Weight of Foreground Data

In section 3, we presented a heuristic for choosing the value of w, based on the unlabeled data only. This section presents the results of this heuristic (Tables 3 and 4). The tables present the value of w chosen by the heuristic, the optimal, according to the AUC score, choice of w in the range [1,30] if we were given the labels of all unlabeled data, the number of unlabeled examples assigned to the positive class by the method trained with the heuristic w: |POS|=(TP+FP) in E_i just before entering the

final stage, the number of false positive examples: FP in E_i, and the performance of TPN^2 using PEBL with the heuristic and the optimal values of w.

Table 3. Task A performance of the method, using PEBL in stage 4 and setting the value of w with the proposed heuristic vs. the optimal value.

Inbox	w (heur)	w (opt)	|POS|	FP	AUC (heur)	AUC (opt)
task_a_u00	5	27	905	2	0.936654	0.939847
task_a_u01	13	19	992	2	0.948652	0.949384
task_a_u02	21	15	1157	8	0.991288	0.991479
Average					**0.958865**	**0.960237**

Table 4. Task B performance of the method, using PEBL in stage 4 and setting the value of w with the proposed heuristic vs. the optimal value.

Inbox	w (heur)	w (opt)	|POS|	FP	AUC (heur)	AUC (opt)
task_b_u00	1	4	181	2	0.9852	0.9915
task_b_u01	9	25	181	1	0.986375	0.9904
task_b_u02	27	2	183	1	0.9857	0.9876
task_b_u03	1	9	197	16	0.981	0.9945
task_b_u04	1	26	144	9	0.929975	0.94415
task_b_u05	1	1	122	16	0.853175	0.853175
task_b_u06	18	28	122	5	0.87295	0.88445
task_b_u07	1	1	184	4	0.98505	0.98505
task_b_u08	8	5	198	7	0.99365	0.995975
task_b_u09	1	19	187	4	0.976325	0.985975
task_b_u10	1	26	165	13	0.925125	0.970925
task_b_u11	1	20	158	4	0.939425	0.9521
task_b_u12	1	17	188	1	0.9861	0.9921
task_b_u13	1	1	124	5	0.946575	0.946575
task_b_u14	3	3	159	2	0.9404	0.9404
Average					**0.952468**	**0.960992**

The first observation is that the chosen value of w is much more variable in task A, than task B. In task B, in 10 out of the 15 mailboxes the heuristic chooses to give the same weight to foreground and background data. Practically, this means that 10 out of the 15 classifiers in task B are identical. This is an indication of the similarity between the unlabeled data of different users in task B. In contrast, the values of w chosen in task A are high, focusing the training process on the data of the user, rather than the background data.

Another observation is that the choice of w with the heuristic method is quite good in most cases. Although the choices are not so close to the ones we would choose if we were given the labels of unlabeled data, the optimal w does not lead to much better performance.

In order to study the sensitivity of TPN^2 to the choice of w, figure 1 presents the AUC performance using PEBL for varying w in the two tasks. For the sake of comprehensibility, figure 1 presents results for only four indicative datasets of task B.

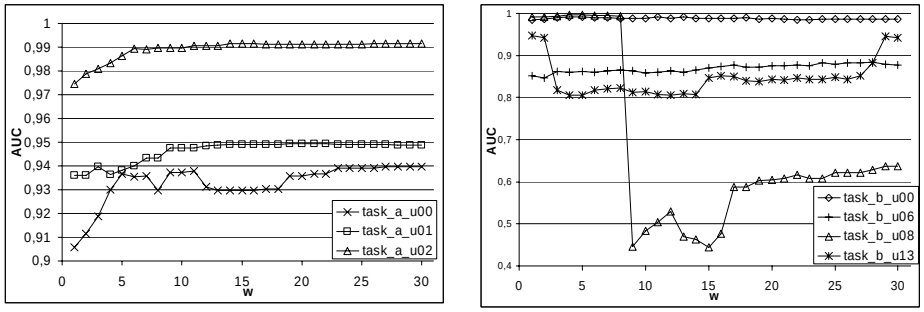

Fig. 1. Performance for inboxes of Task A and Task B varying w.

The figure shows that the method is relatively insensitive to the choice of w in most of the datasets. This is also confirmed by most of task B datasets that are not shown here. Nevertheless, a careful examination of the curves shows that the choice of w is important. One obvious argument for that is the steep and sudden change decrease in performance for dataset task_b_u08 when w changes from 8 to 9. More importantly, the shape of the curves is very different in task A than task B. Choosing a low value for w in task A could hurt the performance seriously.

Finally, it is also not always true that a large value of w is better, as shown by the performance in task_b_u08. Even in task A, where the curves in figure 1 seem to indicate that there is no difference for any big value of w, the performance slightly decreases after the optimal value of w. For w → infinity, which means no influence of the background inboxes ($E=E_i$), we have the following values of AUC for each inbox: 0.895408, 0.909718, and 0.875867, which are much lower than the best we achieved.

Therefore, one needs to choose w carefully, although its exact value can vary without significant loss of performance in most cases. The proposed heuristic works reasonably well, although there could be room for improvement.

4.3 Performance of the Algorithms used in Different Stages

This subsection examines the contribution of each of the three learning stages to the performance of TPN^2. Tables 5 and 6 present the results obtained in each stage for each dataset. All of the results are obtained using the value of w chosen by the heuristic of section 3.

Table 5. Performance in the three learning stages for Task A.

	Stage 2	Stage 3	Stage 4	
Inbox	NBM	PNB	PEBL	Roc-SVM
task_a_u00	0.818971	0.864881	0.936654	0.924884
task_a_u01	0.874001	0.901113	0.948652	0.945581
task_a_u02	0.897548	0.967851	0.991288	0.987226
Average	**0.863507**	**0.911282**	**0.958865**	**0.952564**

Table 6. Performance in the three learning stages for Task B.

	Stage 2	Stage 3	Stage 4	
Inbox	NBM	PNB	PEBL	Roc-SVM
task_b_u00	0.493075	0.948963	0.9852	0.981825
task_b_u01	0.456338	0.952325	0.986375	0.980175
task_b_u02	0.7228	0.954787	0.9857	0.9856
task_b_u03	0.707225	0.984637	0.981	0.980975
task_b_u04	0.77165	0.878	0.929975	0.926025
task_b_u05	0.617887	0.761825	0.853175	0.830475
task_b_u06	0.569925	0.768138	0.87295	0.8687
task_b_u07	0.563175	0.974075	0.98505	0.986025
task_b_u08	0.520763	0.986625	0.99365	0.9943
task_b_u09	0.431412	0.964063	0.976325	0.9776
task_b_u10	0.6655	0.9127	0.925125	0.936025
task_b_u11	0.714988	0.905338	0.939425	0.9368
task_b_u12	0.634975	0.967538	0.9861	0.988375
task_b_u13	0.66315	0.853562	0.946575	0.942425
task_b_u14	0.56955	0.904675	0.9404	0.9378
Average	**0.606828**	**0.914483**	**0.952468**	**0.950208**

As expected, the performance of the fully-supervised classifier (NBM) is much better in task A than task B, since the labeled dataset available in task A is larger. However, with the use of PNB, our method is able to reach approximately the same level of performance in stage 3. Then, the improvement in the fourth stage is essentially the same for both tasks and both of the algorithms that we tested. Thus, the main conclusion is that the proposed method can compensate for the lack of labeled data, by iteratively searching for strong positive examples in the unlabeled data set. Furthermore, the use of a two-step positive-only learner in the last stage is important, when a substantial set of strong positives has been established.

5 Conclusions

In this paper, we introduced the TPN^2 method, which tackles the problem of learning from labeled and unlabeled that are derived from different distributions. The method adopts a four-stage approach combining fully-supervised and positive-only learning methods. The underlying assumption is that the positive examples are more similar in the labeled and unlabeled data than the negative ones. Based on this assumption, the core of the method iteratively selects strong positive examples from the unlabeled data, starting from the ones most confidently identified by a classifier trained on the labeled data. Furthermore, the method handles unlabeled data comprising of different subsets. In that case, the method builds a separate classifier for each subset, using the whole set of unlabeled data, but focusing more on the current subset.

The proposed method participated in the ECML/PKDD Discovery Challenge 2006, the subject of which was the construction of personalized spam filters. The challenge

defined two tasks, in which a set of public email data was given as labeled and a number of personal inboxes as unlabeled data. The two tasks posed a different proportion of labeled and unlabeled data, as well as a different number of personal inboxes. The proposed method obtained the second place in both tasks, as it is particularly suitable for the scenario of the challenge, i.e. spam (positive) email is more homogeneous in the two datasets (public and private) than ham (negative) email.

The paper contains a selection of the results obtained in the various experiments with the parameters of the method, focusing particularly on the choice of algorithms for the four stages of the method and the choice of weight for the foreground data. Despite the good results, a number of extensions seem interesting, such as the use of different weights for different subsets of the unlabeled background data. Additionally, a different configuration of the positive-only learners could be used to reduce the risk of error amplification by the iterative use of the same search bias. Finally, the method should be tested on other problems, which may violate its underlying assumptions.

References

1. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. Proceedings of the Workshop on Computational Learning Theory, COLT-98, (1998) 92-100.
2. Denis, F., Gilleron, R., Tommasi, M.: Text classification from positive and unlabeled examples. Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU, (2002).
3. Joachims, T.: Transductive inference for text classification using support vector machines. Proceedings of ICML-99, 16th International Conference on Machine Learning, (1999) 200-209.
4. Li, X., Liu, B.: Learning to classify text using positive and unlabeled data. Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI-03, (2003).
5. Liu B., Dai Y., Li X., Lee W., Yu P.: Building text classifiers using positive and unlabeled examples. Proceedings of the Third IEEE International Conference on Data Mining, ICDM-03, (2003).
6. Liu, B., Lee, W. S., Yu, P., Li, X.: Partially supervised classification of text documents. Proceedings of the Nineteenth International Conference on Machine Learning, ICML-02, (2002).
7. McCallum, A., Nigam, K.: A comparison of event models for naïve Bayes text classification. AAAI-98 Workshop on Learning for Text Categorization, (1998).
8. Metsis V., Androutsopoulos I., Paliouras G.: Spam Filtering with Naive Bayes - Which Naive Bayes?. Proceedings of the 3rd Conference on Email and Anti-Spam, CEAS-06, (2006).
9. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, 39(2/3), (2000) 103-134.
10. Schneider, K.-M.: On Word Frequency Information and Negative Evidence in Naive Bayes Text Classification. España for Natural Language Processing, EsTAL, (2004).
11. Yu, H., Han, J., Chang, K.: PEBL: Positive example based learning for Web page classification using SVM. Proc. ACM SIGKDD International Conference on Knowledge Discovery in Databases, KDD-02, (2002).