Automated Retraining Methods for Document Classification and their Parameter Tuning

Stefan Siersdorfer   Gerhard Weikum

MPI–I–2005–5–002   August 2005

FORSCHUNGSBERICHT   RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Stuhlsatzenhausweg 85   66123 Saarbrücken   Germany

Authors' Addresses

Stefan Siersdorfer, Gerhard Weikum
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
{stesi,weikum}@mpi-sb.mpg.de

Abstract

This paper addresses the problem of semi-supervised classification on document collections using retraining (also called self-training). A possible application is focused Web crawling which may start with very few, manually selected, training documents but can be enhanced by automatically adding initially unlabeled, positively classified Web pages for retraining. Such an approach is by itself not robust and faces tuning problems regarding parameters like the number of selected documents, the number of retraining iterations, and the ratio of positive and negative classified samples used for retraining. The paper develops methods for automatically tuning these parameters, based on predicting the leave-one-out error for a re-trained classifier and avoiding that the classifier is diluted by selecting too many or weak documents for retraining. Our experiments with three different datasets confirm the practical viability of the approach.

Chapter 1

Introduction

1.1 Motivation

Automatic document classification is useful for a wide range of applications such as organizing web, intranet, or portal pages into topic directories, filtering news feeds or mail, focused crawling on the web or in intranets, and many more. In some applications, the availability of good training data for the classifier is the key bottleneck. As an example, consider a personalized or community information tool that uses thematically focused crawling [10] to build and maintain a directory or index for browsing, search, or recommendations. The quality of this approach stands or falls with the amount and quality of the training data that the underlying classifier has available. But this training data has to be provided by the user, and it is unrealistic to expect humans to spend extensive time on this task. Thus, focused crawling is often bound to start with a few dozen or at most a few hundred training documents (e.g., taken from bookmark collections).

To overcome the training bottleneck, semi-supervised learning techniques could be applied. In our given setting, the classifier could be bootstrapped by training it with whatever explicitly class-labeled training data are available and used for making decisions about the classes of previously unseen, unlabeled test documents retrieved by the crawler. These decisions would have a certain degree of uncertainty, depending on the classifier's statistical learning model. However, some of the test documents are usually accepted for their corresponding classes with high statistical confidence, and these could then be selected for retraining the classifier, now with considerably more training documents. Obviously, this simple idea does not provide a robust solution, for the automatically selected, additional training data may also increase the classifier's uncertainty and may eventually lead to an unintended topic drift. In this paper we address the issue of how to make such a semi-supervised classifier robust and practically viable.

1.2 Contribution

There are various approaches, like Transductive SVM, EM-iterated Bayesian Classifiers, or Co-Training [29, 24, 7], that successfully use information from initially *unlabeled* documents to improve classification results. However these methods come with parameters, which have a crucial influence on the quality of the classification results and need to be *tuned manually* on a per application basis.

To this end we propose a retraining algorithm that performs *automatic parameter tuning*. When our method considers adding a batch of initially unlabeled documents to the training set, it predicts the resulting improvement (or degeneration) of the classifier's accuracy by performing a leave-one-out validation. The training set is extended, by selecting the unlabeled documents with highest classification confidence and then retraining the classifier, only as long as the predictor expects an improvement. To avoid extensive leave-one-out validations, which are resource-intensive, the predictor invokes the validation merely after a certain number of iterations and rather uses a spline interpolation technique for less expensive estimation.

A particularly subtle but important point in this procedure is that one should often select different numbers of positive and negative samples for retraining, depending on the ratio in the underlying corpus. In the case of a focused Web crawler, usually a much larger fraction of negative (i.e., thematically uninteresting) documents is seen and may lead to a wrong classifier bias unless such corrective steps are taken. This point holds for most other applications too (e.g., news filtering), and it is often inherent in the fact that a classifier discriminates one positive topic against many other negative topics.

The novel contributions of this paper are the following:

1. We develop a robust, practically viable procedure for automated retraining of classifiers with careful selection of initially unlabeled documents.

2. We perform comprehensive experiments that evaluate our retraining procedure against state-of-the-art semi-supervised classification methods like EM-iterated Bayesian classifiers, Transductive SVMs, and Spectral Graph Transduction.

1.3 Related Work

There is a considerable prior of work on classification using unlabeled data (also called semi-supervised learning), see [27] for an overview. Naive Retraining where new documents with highest classification confidence are iteratively added to the training set, is, e.g., described in [10]; but these methods perform often

worse than the underlying base learning method. A more enhanced EM (Expectation
Maximization)-based variant for Bayesian Classifiers is proposed in [24] and applied to text classification. For Transductive SVM [16, 29] and Semi-Supervised SVM [5] unlabeled samples are taken into account (opposite to standard SVM) in a modified optimization problem (standard SVM cannot use unlabeled samples at all). Co-training [7] splits the feature space into conditionally independent dimensions and performs retraining on the corresponding classifiers. Recent graph-based semi-supervised learning algorithms work by formulating the assumption that "nearby" points, and points in the same structure should have similar labels [19, 30, 17]. In [6] semi-supervised learning is combined with ensemble classification methods. An approach for the case that only positive (and no negative) training data plus unlabeled data are available is described in [21]. In [3] semi-supervised learning is used for text summarization; in [31] a retraining method with user feedback as a stopping criterion is used for image retrieval. However, to our knowledge, none of these methods deals with the problem of automatically tuning their parameters.

The issue of asymmetric distribution of documents among different classes is addressed, e.g., in [20, 8, 13], and the problem of automated parameter tuning has been considered in the field of machine learning, e.g., in [18], but, to our knowledge, not in the context of retraining.

1.4 Outline

The rest of the paper is organized as follows. Section 2 gives some technical background on document classification. In Section 3 we describe our retraining approach with automatic parameter tuning. Section 4 presents our experimental results.

Chapter 2

Technical Basics

Classifying text documents into thematic categories usually follows a supervised learning paradigm and is based on training documents that need to be provided for each topic. Both training documents and test documents, which are later given to the classifier, are represented as multidimensional feature vectors. In the prevalent bag-of-words model the features are derived from word occurrence frequencies, e.g. based on tf*idf feature weights [4, 23]. Often feature selection algorithms are applied to reduce the dimensionality of the feature space and eliminate "noisy", non-characteristic features, based on information-theoretic measures for feature ordering (e.g., relative entropy or information gain).

Feature vectors of topic labeled text documents (e.g., capturing $tf \cdot idf$ weights of terms) are used to train a classification model for each topic, using probabilistic (e.g., Naive Bayes) or discriminative models (e.g., SVM). Linear support vector machines (SVMs) construct a hyperplane $\vec{w} \cdot \vec{x} + b = 0$ that separates the set of positive training examples from a set of negative examples with maximum margin δ, as illustrated in Figure 2.1 . This training requires solving a quadratic optimization problem whose empirical performance is somewhere between quadratic and cubic in the number of training documents [9]. For a new, previously unseen, (test) document $\vec{d}$ the SVM merely needs to test whether the document lies on the "positive" side or the "negative" side of the separating hyperplane. The decision simply requires computing a scalar product of the vectors $\vec{w}$ and $\vec{d}$. SVMs have been shown to perform very well for text classification (see, e.g., [12, 14]).

Unlike the inductive SVM setting, for Transductive SVM (TSVM) [16, 29], a hyperplane is computed that separates *both* taining and (unlabeled) test data with maximum margin, as illustrated in Figure 2.2.

The most widely used technique for empirically estimating the classifier quality is *cross-validation* [23] on a set of independent data samples with known topic memberships (aka. class labels). The partitioning is systematically varied by dividing the overall data into 4 groups and investigating each of the four choices

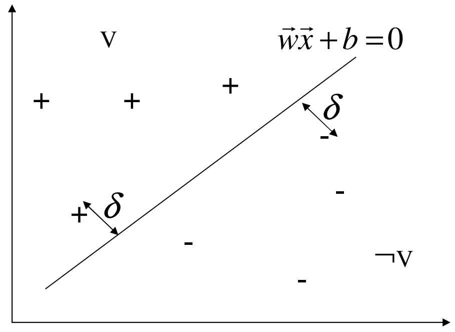

Figure 2.1: Linear SVM Classifier

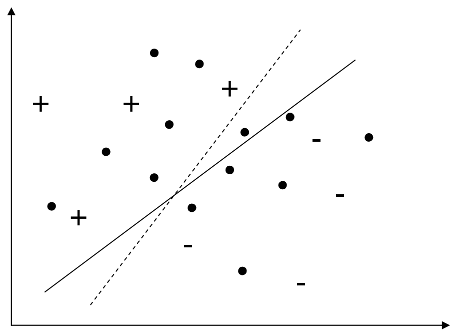

Figure 2.2: Linear TSVM Classifier (dashed hyperplane)

for using one group as test data and the other 3 groups for training; the empirical results are finally averaged over all choices. An important variation is *leave-one-out validation* [23]. Here the n documents of a data collection are divided by the ratio $(n-1):1$. Both methods are also popular for predicting a classifier's quality. Leave-one-out prediction is more accurate than prediction based on cross-validation but requires training the classifier n times, unless special properties of the classifier's underlying model could be exploited. For SVMs [15] has proposed a more efficient estimation technique known as the $\xi\alpha$ estimator, but it gives only approximate results and turned out to be too crude in our experiments.

In this paper we consider only binary classifiers that make a decision for a single topic, based on positive and negative training examples.

Chapter 3

Retraining and its Parameter Tuning

3.1 A Simple Base Algorithm

Consider a training set T and a set of unlabeled data U. We can perform retraining by iteratively building a classifier C on T, classifying the documents in U and adding the documents with the highest classification confidence. Classification confidence could be estimated, e.g., by the distance from the separating hyperplane in the SVM case or by the probability of accepting a document for a class.

This procedure can be described more precisely by the following algorithm:

```
Input: training set T = T_0
       set of unlabeled Data U = U_0

for i = 1, ..., m do
    build classifier C on T
    classify U
    U_pos := top-p positively classified docs
    U_neg := top-n negatively classified docs
    T = T + U_pos + U_neg
    U = U - U_pos - U_neg
```

This algorithm provides us with a tradeoff. On one hand, a higher number of training examples could potentially improve the classification accuracy; on the other hand, there are potentially incorrectly labeled documents among the docs in U_{pos} and U_{neg}, which can dilute the training set. The algorithm thus has two important tuning parameters:

1. the number m of iterations
2. the ratio p/n between new positively classified and negatively classified docs used for retraining

In the following we show how we can automatically tune these parameters. Note that the total number of selected documents for each retraining step, $r := p + n$ could be considered as an additional tuning parameter. However, we can simply choose it sufficiently small to be on the conservative side.

3.2 Tuning the Number of Iterations

Because of the tradeoffs mentioned above, a higher number of iterations do not necessarily imply a lower error. Our idea now is to approximate this error curve on the test set U by an estimated error curve.

For a retraining step we can build an error estimator by performing leave-one-out validation of the current classifier C on the original training set T_0, i.e., the part of the training set that consists of the manually labeled documents (which are correct with perfect confidence). This *partial* leave-one-out can be described by the following pseudo code:

```
Input: initial training set T = T_0
       L = set of automatically labeld data
           used for retraining
       C = classifier trained on T+L
Output: estimated error for C

classifiedDocs = 0
incorrectlyClassified = 0

for all docs d in T do
     build classifier C_L1o  on T+L-{d}
     if ( realClass(d) != classsify(d,C_L1o) )
          incorrectlyClassified++
     classifiedDocs++

return incorrectlyClassified/classifiedDocs
```

For a set of sample estimates

$$\{(i_0, estError(i_0)), \ldots, (i_l, estError(i_l))\}, \tag{3.1}$$

where the i_j values are the iteration numbers and $estError(i_j)$ is the estimated error, we can now approximate the overall error curve by fitting the sample estimates, as illustrated in Figure 3.1.

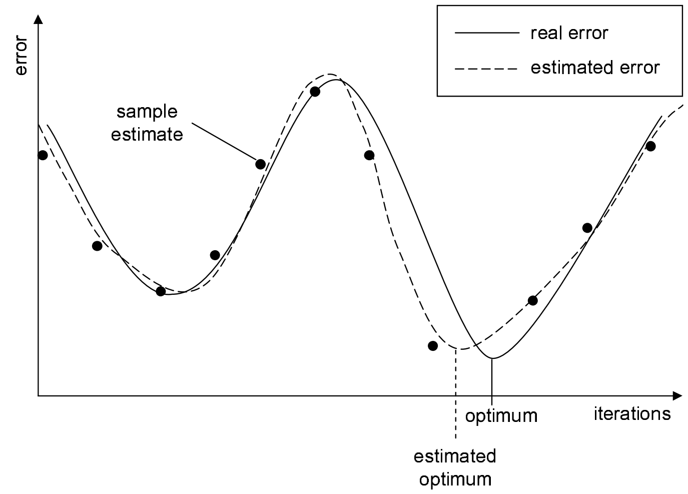

Figure 3.1: Optimal Number of Iterations: Approximation by Estimated Error Curve

There are various approaches to this curve fitting. In our experiments we obtained good performance using cubic splines. Cubic splines are used in many areas, e.g., bio medicine, signal processing, and computer graphics [11, 25, 28]. In our experiments we also tested other approaches like linear splines, and error estimation by the less time consuming k-fold-cross-validation instead of leave-one-out.

Having approximated the error estimation curve $S(x)$, we choose the retraining classifier C in the iteration i with minimum $S(i)$ (see Figure 3.1). Choosing the number of supporting points for the fitting is an efficiency issue. The more supporting points the better the approximation but the higher the overall cost for computing the estimator values.

The classifier can be optimized in the same way for other quality measures like the F-measure (the harmonic mean of precision and recall).

3.3 Tuning the Ratio of Positive and Negative Samples

For an effective classification the training set should be an appropriate representation of the test set. For binary classification, it is especially helpful if the ratio between positive and negative documents is approximately the same for the test and the training set. For example, Bayesian classifiers take the prior class probabilities explicitly into account. For SVM a badly proportionalized training set can also lead to a disadvantageous bias [8]. The assumption of having a training set with the same ratio ratio of positive and negative documents as a test set is not at all self-guaranteed or easy to satisfy in practice. Typically a human, collecting training documents, would rather choose roughly the same number of documents for each class, even if there are significant (but a priori unknown) differences in the real world.

The idea is to overcome this problem by adjusting the training set such that it better represents the test set. To do so, in each iteration of our retraining algorithm we approximate the ratio between positive and negative documents by applying the current classifier to the set of initially unlabeled data U_0 (test data). Among a small number r of new retraining documents we choose the number of positive and negative documents, n and p, such that the difference between the overall ratio of positive and negative training docs and the estimated ratio on the unlabeled data is minimized.

More formally let t_{pos} be the number of positive, t_{neg} be the number of negative training documents in the current iteration, v_{pos} be the number of unlabeled documents classified as positive by the current classifier C, and v_{neg} be the number of documents classified as negative. Then we choose the number of *newly* added positive and negative documents for retraining, p and n, such that the ratio $(t_{pos}+p) : (t_{neg}+n)$ between the overall number of positive and negative training documents provides the best approximation for the ratio $v_{pos} : v_{neg}$ of positive and negative test documents estimated by the current classifier:

$$p = \text{arg min}_{x \in \{0,\ldots,r\}} \left| \frac{t_{pos} + x}{t_{neg} + r - x} - \frac{v_{pos}}{v_{neg}} \right| \tag{3.2}$$

and

$$n = r - p \tag{3.3}$$

As an illustrative example consider the case that the *intitial* (manually labeled) training set consists of 100, and the set of initially unlabeled (test) data, that we aim to classifiy automatically, consists of 200 documents. Furthermore assume that, after i iterations, 100 of the test documents were automatically classified during previous iterations, and were, along with the initial training set, used to build the current classifier. Assume that the initial training set contains 50 positive and 50 negative examples and the 100 additional retraining documents after i iterations contain 60 examples classified as pos and 40 examples classified as neg. Thus after the i-th iteration, the ratio $\frac{t_{pos}}{t_{neg}}$ on the total number of examples used for training the current classifier is $\frac{50+60}{50+40} \approx 1.22$.

For the $i+1$-th iteration, we apply the current classifier (obtained in the $i-th$ iteration) to the the 200 initially unlabeled documents, resulting in, say, 120 positively and 80 negatively classified documents, i.e. $\frac{v_{pos}}{v_{neg}} = \frac{120}{80} = 1.5$. In the next step of the $i+1$-th iteration, we add $r = 10$ new retraining documents, such that the ratio $pos : neg$ for the overall number of training documents used for the next classifier provides the best approximation for $\frac{v_{pos}}{v_{neg}}$. In our example this holds for $p = 10$ positive and $n = 0$ negative retraining documents. For this choice the fraction of positive and negative examples for the new classifier is $\frac{50+60+10}{50+40+0} \approx 1.33$, a better approximation than the ratio 1.22 after iteration i.

3.4 The Enhanced Retraining Algorithm

With the parameter tuning methods described above, our retraining algorithm now works as follows: We retrain as long as documents for retraining are available. In each retraining iteration we add a small number r of documents to the training set, determining the ratio between new positive and negative training documents as described in Section 3.3. Every $stepsize$ iterations we compute and save an error estimator. We apply curve fitting to the estimated error, and choose the classifier corresponding to the minimum estimated error (see Section 3.2).

The following pseudo code summarizes our modified retraining algorithm:

```
Input: training set T = T_0
       set of unlabeled Data U = U_0
       stepsize

set of classifiers C-Set = empty
set of supporting points Support-Set = empty
iteration number i = 0;
while (U is not empty) do
   build classifier C on T
   add (i,C) to C-Set
   estimate p and n \\see Section 3.3
   classify U
   U_pos := top-p positively classified docs
   U_neg := top-n negatively classified docs
   T = T + U_pos + U_neg
   U = U - U_pos - U_neg
   if (i mod stepsize = 0)
      estimate error estError of C by leave-one-out on T_0
      add (i,estError) to Support-Set
   i++

compute interpolating curve S on Support-Set \\see Section 3.2
choose j which minimizes S(i)
return Classifier c from C-Set with iteration number = j
```

Chapter 4

Experiments

4.1 Setup

We performed a series of experiments with real-life data from the following sources.

1. The Newsgroups collection at [1]. This collection contains 17,847 postings collected from 20 Usenet newsgroups. from topics like 'rec.autos', 'sci.space', etc.

2. The Reuters articles [22]. This is the most widely used test collection for text categorization research. The collection contains 21,578 Reuters newswire articles; 12,904 of them are subdivided into categories ('earn', 'grain', 'trade', etc.).

3. The Internet Movie Database (IMDB) at [2]. Documents of this collection are short movie descriptions that include the storyboard, cast overview, and user comments. This collection contains 20 topics according to particular movie genres ('drama', 'horror' etc.). Only movies were considered that have a unique genre (avoiding movies that belong to, say, both 'drama' and 'science fiction'). Thereof it holds a total of 34,681 documents.

For every data collection we considered each class with at least 300 documents. We obtained 20 classes for Newsgroups, 8 for Reuters and 9 for IMDB. For each class we randomly chose 100 documents as positive training examples and 100 negative examples from all other classes. For validation we considered two cases:

1. the symmetric case: we chose equal numbers of positive and negative test documents for each class (200 per class)

Method	Newsg. avg(error)	IMDB avg(error)	Reuters avg(error)	Newsg. avg(F1)	IMDB avg(F1)	Reuters avg(F1)
SVM	0.106 ± 0.0067	0.262 ± 0.0144	0.06 ± 0.0082	0.891	0.734	0.941
TSVM	$\mathbf{0.1 \pm 0.0066}$	0.261 ± 0.0143	0.059 ± 0.0082	0.9	0.739	0.941
TSVM+est	0.105 ± 0.0067	0.262 ± 0.0144	$\mathbf{0.059 \pm 0.0082}$	**0.892**	0.734	0. **942**
EM-Bayes	0.129 ± 0.0073	$\mathbf{0.245 \pm 0.014}$	0.072 ± 0.009	0.884	0. **764**	0.93
SGT	0.119 ± 0.0071	0.277 ± 0.0146	0.088 ± 0.0098	0.882	0.727	0.914
RetCsplL1o	0.123 ± 0.0072	0.263 ± 0.0144	0.068 ± 0.0087	0.867	0.723	0.934
RetCsplCv	0.116 ± 0.007	0.257 ± 0.0143	0.062 ± 0.0083	0.877	0.729	0.939
RetLsplL1o	0.123 ± 0.0072	0.256 ± 0.0143	0.066 ± 0.0086	0.871	0.731	0.936
RetLsplCv	0.115 ± 0.007	0.258 ± 0.0143	0.062 ± 0.0083	0.879	0.731	0.939
RetCv	0.111 ± 0.0069	0.254 ± 0.0142	0.061 ± 0.0083	0.882	0.735	0.94

Figure 4.1: Macro-averaged Results for **Symmetric** Test Set: Baseline and Retraining Methods

Method	Newsg. avg(error)	IMDB avg(error)	Reuters avg(error)	Newsg. avg(F1)	IMDB avg(F1)	Reuters avg(F1)
SVM	0.097 ± 0.0035	0.246 ± 0.0075	0.075 ± 0.0049	0.726	0.481	0.783
TSVM	0.364 ± 0.0056	0.401 ± 0.0086	0.362 ± 0.0089	0.434	0.376	0.437
TSVM+est	0.096 ± 0.0035	0.249 ± 0.0075	0.076 ± 0.0049	0.728	0.475	0.78
EM-Bayes	0.202 ± 0.0047	0.267 ± 0.0077	0.093 ± 0.0054	0.596	0.498	0.75
SGT	0.216 ± 0.0048	0.329 ± 0.0082	0.167 ± 0.0069	0.543	0.402	0.606
RetCsplL1o	$\mathbf{0.077 \pm 0.0031}$	0.207 ± 0.0071	$\mathbf{0.058 \pm 0.0043}$	**0.749**	0.497	**0.818**
RetCsplCv	0.08 ± 0.0032	0.211 ± 0.0071	0.059 ± 0.0044	0.749	0.496	0.817
RetLsplL1o	0.081 ± 0.0032	0.212 ± 0.0071	0.058 ± 0.0043	0.744	0.49	0.813
RetLsplCv	0.083 ± 0.0032	0.209 ± 0.0071	0.06 ± 0.0044	0.744	0.491	0.812
RetCv	0.084 ± 0.0032	$\mathbf{0.204 \pm 0.007}$	0.059 ± 0.0044	0.745	**0.499**	0.816

Figure 4.2: Macro-averaged Results for **Asymmetric** Test Set: Baseline and Retraining Methods

2. the asymmetric case: we chose the number of positive and negative test documents in a ratio of $1 : 6$ (i.e., 200:1200).

In all experiments, the standard bag-of-words model [4] (using term frequencies to build L1-normalized feature vectors, stemming with the algorithm of Porter [26], and deletion of stopwords) was used for document representation. We used binary classifiers so as to recognize documents from one specific topic against all other topics; this setup was repeated for every topic.

For each data collection we computed the macro-averaged error (i.e., the average ratio of incorrectly classified documents to the number of test documents) along with the 95 percent confidence interval and the macro-averaged F1 value (the harmonic mean of precision and recall).

4.2 Results

We compared the following classification methods:

1. Standard linear SVM (**SVM**)
2. Standard linear TSVM. Here the fraction f of unlabeled examples to be classified into the positive class is a selectable parameter. As default setting we used the ratio between the positive and the negative examples in the training data. (**TSVM**)
3. Linear TSVM with parameter where the ratio f between positive and negative test documents was set according to the SVM classification (Method 1) on the test documents. (**TSVM+est**)
4. The augmented EM-iterated Bayesian classifier with weighting of the unlabeled data as described in [24]. Here we determined the weighting parameter λ by leave-one-out validation (considering the values between 0 and 1 with a step width of 0.2), choosing the λ with the lowest estimated error. (**EM-Bayes**)
5. Spectral Graph Transduction as described in [17] (**SGT**)
6. Our retraining approach with linear SVM (Method 1) as the underlying base classifier and 10 new retraining documents per iteration and
   (a) error/F1 prediction by leave-one-out estimation invoked after every 10 iterations and cubic spline interpolation (**RetCsplL1o**)
   (b) error/F1 prediction by leave-one-out eatimation invoked after every 10 iterations and linear spline interpolation (**RetLsplL1o**)
   (c) error/F1 prediction by 5-fold cross-validation invoked after every 10 iterations and cubic spline interpolation (**RetCsplCv**)
   (d) error/F1 prediction by 5-fold cross-validation invoked after every 10 iterations and linear spline interpolation (**RetLsplCv**)
   (e) error/F1 prediction by 5-fold cross-validation invoked after every iteration - and no interpolation (**RetCv**)

For SVM and TSVM we used the popular *SVMlight* implementation [14] with parameter C = 1000 (tradeoff between training error and margin). For the Spectral Graph Transductor we used the *SGTlight* implementation with parameterization as desribed in [17].

The average results for the symmetric and asymmetric test sets are shown in Figures 4.1 and 4.2. Detailed results for the different classes are shown in Figures 1 through 3 in the appendix. (The best values are highlighted in boldface.)

The main observations are:

- For the *symmetric* test case, there is no clear winner. For IMDB the EM-Bayes Method perfoms best, for Newsgroups and Reuters the TSVM algorithm is the best algorithm. An explanation for the good performance of standard TSVM is, that, for the symmetric case, the parameter f agrees completely with the real ratio between positive and negative documents in the test set.

- In the *asymmetric* test case, our retraining algorithm clearly provides the best performance on all three datasets. For example, on the IMDB data, which is the hardest test case in terms of the absolute accuracy that was achievable, we reduce the error from approximately 25-27 percent (for SVM and TSVM with estimator and for EM-iterated Bayes) to 20.7 percent, quite a significant gain. The very bad performance of standard TSVM can be explained by the big gap between the parameter f, estimated on the training set, and the real ratio between positive and negative documents in the asymmetric test set.

As we regard the asymmetric test case, significantly more unacceptable test documents than acceptable ones, as the far more realistic setting (e.g. in focused crawling, news filtering, etc.), we conclude that the newly proposed retraining method is the clear winner and outperforms the previously known state-of-the-art algorithms by a significant margin.

Chapter 5

Conclusion and Future Work

This work has been motivated by the fact that the availability of training data is often a critical point in real applications. This has led us to a semi-supervised learning approach with iterative retraining using initially unlabeled data. An additional difficulty that real applications often pose is the imbalance between positive and negative documents in the corpus that creates a mismatch with the ratio of positive and negative training samples and may result in a wrong bias of the classifier. In Web applications, and for large-scale intranet corpora, this is a typical situation and creates a major impediment to the previously proposed state-of-the-art techniques for semi-supervised classification. Our method successfully addresses these practically relevant issues, which were largely disregarded by prior work, and significantly outperforms the other methods in terms of classification accuracy.

Generalizing beyond the scope of this paper, it seems that supervised learning methods, regardless of their great success in many areas of computer science, face an unsurmountable training bottleneck in application areas like Web mining. This shifts the focus to unsupervised and semi-supervised techniques, and we believe that this calls for new approaches not only at the level of mathematical models and algorithms but also at the level of scalable and robust system engineering in order to meet the needs of real-life applications.

Bibliography

[1] The 20 newsgroups data set. *http://www.ai.mit.edu/ jrennie/20Newsgroups/*.

[2] Internet movie database. *http://www.imdb.com*.

[3] Massih-Reza Amini and Patrick Gallinari. The use of unlabeled data to improve supervised learning for text summarization. In *SIGIR '02*, pages 105–112. ACM Press, 2002.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[5] Kristin P. Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *NIPS 1999*, pages 368–374. MIT Press, 1999.

[6] Kristin P. Bennett, Ayhan Demiriz, and Richard Maclin. Exploiting unlabeled data in ensemble methods. In *SIGKDD*, pages 289–296. ACM Press, 2002.

[7] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Workshop on Computational Learning Theory*, 1998.

[8] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Training text classifiers with SVM on very few positive examples. *Technical Report MSR-TR-2003-34, Microsoft Corp.*, 2003.

[9] C.J.C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 1998.

[10] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[11] E.Q. Chen and C.F. Lam. Predictor-corrector with cubic spline method for spectrum estimation in compton scatter correction of spect. *Computers in biology and medicine, 1994, vol. 24, no. 3, pp. 229, Ingenta*.

[12] S. Dumais and H. Chen. Hierarchical classification of Web content. *SIGIR*, 2000.

[13] H. Guo and H. L. Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explorations, 30 - 39, 2004*.

[14] T. Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. *ECML*, 1998.

[15] T. Joachims. Estimating the generalization performance of an SVM efficiently. *ECML*, 2000.
[16] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML'99, 200 - 209*, 1999.
[17] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pages 290–297, 2003.
[18] R. Kohavi and G. John. Automatic parameter selection by minimizing estimated error. In *Machine Learning*, 1995.
[19] Balaji Krishnapuram, David Williams, Ya Xue, Alexander Hartemink, Lawrence Carin, and Mario Figueiredo. On semi-supervised classification. In *NIPS*. MIT Press, 2005.
[20] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *ICML'97, Nashville, TN, U.S.A., 179-186*, 1997.
[21] W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML'03, Washingtion USA*, 2003.
[22] David D. Lewis. Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318. Defense Advanced Research Projects Agency, Morgan Kaufmann, February 1991.
[23] C.D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
[24] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Intelligence*, 39(2/3), 2000.
[25] E.I. Okanla and P.A. Gaydecki. A real-time audio frequency cubic spline interpolator. *Signal processing, 1996, vol. 49, no. 1, pp. 45, Ingenta*.
[26] M. Porter. An algorithm for suffix stripping. *Automated Library and Information Systems*, 14(3).
[27] M. Seeger. Learning with labeled and unlabeled data. *Tech. Rep., Institute for Adaptive and Neural Computation, University of Edinburgh, UK*, 2001.
[28] Chris Seymour and Keith Unsworth. Interactive shape preserving interpolation by curvature continuous rational cubic splines. *Appl. Math. 102 (1999), no. 1, 87–117*.
[29] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
[30] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*. MIT Press, 2004.
[31] Z. Zhou, K. Chen, and Y. Jiang. Exploiting unlabeled data in content-based image retrieval. In *ECML'03, Pisa, Italy*, 2004.

class	**SVM** error	**TSVM** error	**TSVM+est** error	**EM-Bayes** error	**SGT** error	**RetCsplL1o** error
alt.atheism	0.108	0.361	0.105	0.238	0.229	**0.071**
comp.graphics	0.116	0.36	0.113	0.258	0.186	**0.093**
comp.os.ms-windows.misc	0.085	0.357	0.084	0.171	0.115	**0.081**
comp.sys.ibm.pc.hardware	0.117	0.36	0.12	0.218	0.174	**0.113**
comp.sys.mac.hardware	0.106	0.363	0.109	0.222	0.285	**0.102**
comp.windows.x	0.084	0.36	0.089	0.298	0.191	**0.071**
misc.forsale	0.058	0.369	0.056	0.099	0.246	**0.051**
rec.autos	0.079	0.361	0.077	0.148	0.272	**0.066**
rec.motorcycles	0.066	0.364	0.064	0.324	0.248	**0.049**
rec.sport.baseball	0.07	0.363	0.071	0.089	0.237	**0.069**
rec.sport.hockey	0.048	0.361	0.048	0.084	0.129	**0.044**
sci.crypt	0.061	0.361	0.063	0.104	0.199	**0.056**
sci.electronics	0.199	0.363	0.201	0.201	0.307	**0.176**
sci.med	0.141	0.364	0.138	0.093	0.219	**0.064**
sci.space	0.074	0.37	**0.07**	0.354	0.182	0.074
soc.religion.christian	0.07	0.361	0.066	0.361	0.189	**0.066**
talk.politics.guns	0.096	0.369	0.094	0.162	0.251	**0.069**
talk.politics.mideast	0.066	0.364	0.066	0.128	0.167	**0.036**
talk.politics.misc	0.142	0.367	0.142	0.231	0.199	**0.092**
talk.religion.misc	0.153	0.371	0.153	0.257	0.288	**0.103**

class	**SVM** F1	**TSVM** F1	**TSVM+est** F1	**EM-Bayes** F1	**SGT** F1	**RetCsplL1o** F1
alt.atheism	0.708	0.438	0.716	0.544	0.524	**0.771**
comp.graphics	0.665	0.44	0.674	0.51	0.551	**0.679**
comp.os.ms-windows.misc	0.725	0.444	**0.73**	0.617	0.681	0.725
comp.sys.ibm.pc.hardware	**0.675**	0.44	0.667	0.559	0.561	0.659
comp.sys.mac.hardware	**0.7**	0.436	0.692	0.553	0.467	0.699
comp.windows.x	0.742	0.44	0.729	0.488	0.571	**0.754**
misc.forsale	0.808	0.427	0.812	0.72	0.493	**0.819**
rec.autos	0.757	0.438	0.761	0.65	0.498	**0.768**
rec.motorcycles	0.787	0.433	0.796	0.463	0.513	**0.826**
rec.sport.baseball	**0.776**	0.436	0.772	0.753	0.51	0.769
rec.sport.hockey	0.845	0.438	0.845	0.77	0.674	**0.855**
sci.crypt	0.811	0.438	0.806	0.728	0.548	**0.824**
sci.electronics	0.541	0.436	0.535	**0.557**	0.449	0.555
sci.med	0.644	0.433	0.651	**0.747**	0.549	0.684
sci.space	0.767	0.424	**0.78**	0.439	0.583	0.767
soc.religion.christian	0.787	0.438	**0.8**	0.438	0.576	0.791
talk.politics.guns	0.723	0.427	0.727	0.63	0.513	**0.768**
talk.politics.mideast	0.801	0.433	0.801	0.687	0.602	**0.866**
talk.politics.misc	0.641	0.429	0.641	0.548	0.547	**0.718**
talk.religion.misc	0.622	0.422	0.622	0.521	0.459	**0.684**

Figure 1: Asymmetric Test Set: Detailed Results for Newsgroups

class	**SVM** error	**TSVM** error	**TSVM+est** error	**EM-Bayes** error	**SGT** error	**RetCsplL1o** error
Action	0.267	0.403	0.283	0.281	0.456	**0.239**
Adventure	**0.309**	0.416	0.312	0.284	0.421	0.324
Comedy	0.349	0.434	0.355	**0.304**	0.396	0.32
Documentary	0.128	0.364	0.125	0.107	0.174	**0.094**
Drama	0.324	0.43	0.328	0.404	0.308	**0.259**
Horror	0.217	0.397	0.223	0.246	0.324	**0.199**
Sci-Fi	0.179	0.38	0.176	0.182	0.186	**0.114**
Thriller	0.308	0.41	0.311	0.492	0.482	**0.22**
Western	0.13	0.377	0.127	0.106	0.217	**0.094**

class	**SVM** F1	**TSVM** F1	**TSVM+est** F1	**EM-Bayes** F1	**SGT** F1	**RetCsplL1o** F1
Action	0.445	0.373	0.412	**0.475**	0.333	0.435
Adventure	0.393	0.353	0.387	**0.42**	0.304	0.383
Comedy	0.359	0.324	0.349	**0.402**	0.338	0.341
Documentary	0.664	0.433	0.672	0.7	0.561	**0.718**
Drama	0.357	0.331	0.349	**0.369**	0.35	0.364
Horror	0.486	0.382	0.473	**0.493**	0.371	0.484
Sci-Fi	0.573	0.409	0.58	0.576	0.525	**0.626**
Thriller	0.392	0.362	0.386	0.337	0.323	**0.41**
Western	0.657	0.413	0.664	0.712	0.516	**0.715**

Figure 2: Asymmetric Test Set: Detailed Results for IMDB

class	**SVM** error	**TSVM** error	**TSVM+est** error	**EM-Bayes** error	**SGT** error	**RetCsplL1o** error
acq	0.107	0.36	0.113	**0.083**	0.205	0.096
crude	0.031	0.361	0.031	0.094	0.149	**0.026**
earn	0.049	0.369	0.049	0.057	0.15	**0.031**
grain	0.083	0.363	0.086	**0.056**	0.187	0.064
interest	0.098	0.359	0.098	0.13	0.146	**0.08**
money-fx	0.079	0.363	0.081	0.115	0.191	**0.062**
trade	0.064	0.359	0.065	0.136	0.151	**0.04**
wheat	0.091	0.364	0.088	0.069	0.156	**0.063**

class	**SVM** F1	**TSVM** F1	**TSVM+est** F1	**EM-Bayes** F1	**SGT** F1	**RetCsplL1o** F1
acq	0.71	0.44	0.695	**0.773**	0.572	0.722
crude	0.897	0.438	0.897	0.746	0.629	**0.912**
earn	0.836	0.427	0.836	0.804	0.607	**0.885**
grain	0.759	0.436	0.751	**0.803**	0.568	0.794
interest	0.738	0.442	0.738	0.684	0.655	**0.77**
money-fx	0.772	0.436	0.768	0.711	0.568	**0.798**
trade	0.811	0.442	0.807	0.675	0.624	**0.865**
wheat	0.741	0.433	0.749	**0.8**	0.628	0.797

Figure 3: Asymmetric Test Set: Detailed Results for Reuters