I N F O R M A T I K

Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning

Stefan Siersdorfer Sergej Sizov
Gerhard Weikum

MPI–I–2004–5–001 May 2004

FORSCHUNGSBERICHT RESEARCH REPORT

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

Stuhlsatzenhausweg 85 66123 Saarbrücken Germany

Authors' Addresses

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

Abstract

Automatic text classification methods come with various calibration parameters such as thresholds for probabilities in Bayesian classifiers or for hyperplane distances in SVM classifiers. In a given application context these parameters should be set so as to meet the relative importance of various result quality metrics such as precision versus recall. In this work we consider classifiers that can accept a document for a topic, reject it, or abstain. We aim to meet the application's goals in terms of accuracy (i.e., avoid false acceptances or rejections) and loss (i.e., limit the fraction of documents for which no decision is made). To this end we investigate restrictive forms of SVM classifiers and we develop meta methods that split the training data into subsets for independently trained classifiers and then combine the results of these classifiers. These techniques tend to improve accuracy at the expense of document loss. We develop estimators that help to predict the accuracy and loss for a given setting of the methods' tuning parameters, and a methodology for efficiently deriving a setting that meets the application's goals. Our experiments confirm the practical viability of the approach.

Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning

Stefan Siersdorfer Sergej Sizov Gerhard Weikum

May 11, 2004

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 [10]. There exists a great variety of classification methods such as Naive Bayes, kNN, Rocchio, linear SVM, etc. [21, 10, 13], all of which operate on a high-dimensional feature space usually constructed from word occurrence frequencies in documents (and possibly some additional input such as anchor texts in hyperlink neighbors, neighbor topics, etc.). All methods face inherent *tradeoffs* regarding the result quality metrics: precision, which is the fraction of documents that are automatically placed under some topic and do indeed belong there (as per a human expert's assessment), can be improved by making the classifier more conservative, but this way recall, which is the fraction of positively classified documents among all documents that the human expert would place under the given topic, usually becomes worse.

Whether precision or recall is more important is application dependent, and this raises the need for tuning classification methods towards the goals of the application. To this end we exploit the fact that many methods have certain calibration parameters anyway by which we can control their degrees of making more conservative or more speculative decisions. For example, a Bayesian classifier may accept a given document for some topic only if the probability of the document belonging there exceeds some specific threshold. Similarly, an SVM (support vector machine) classifier may choose to accept only documents whose positive distance from the separating hyperplane is above some threshold. In this work we will also discuss families of "committee-based" meta methods (combining results from multiple classifiers) that come with explicitly designed parameters of this kind.

We consider classifiers for a given topic that make a ternary decision on a newly seen document: they can accept the document for the topic, reject it, or abstain if there is neither sufficiently strong evidence for acceptance nor for rejection. The third option is important as it makes a key difference for constructing meta classifiers that combine the results of different classifiers (e.g., in a quorum consensus manner). Now the quality metrics of interest are primarily the classification *error*, which is the fraction of erroneously accepted or erroneously rejected documents, and the document *loss*, which is the fraction of documents for which the classifier or meta classifier makes no decision at all (i.e., abstains). Alternatively to error we may consider the *accuracy* which is one minus the error. Note that there is a difference between loss and reduced recall: a document that is not accepted by the committee-based meta classifier is not necessarily completely dropped and may be directed to a specialized, computationally more expensive classifier (in the extreme case a human expert) so that recall may still be high, whereas documents that none of the underlying classifiers accepts are definitely dropped and would affect recall.

Like with precision and recall, there is an unavoidable tradeoff between error (or its complement accuracy) and loss and the relative importance of the two metrics depends on the application's goals. For example, in the context of a focused crawler [10] that starts with very few training documents and initially aims to find more "archetype" documents that are characteristic for the topic(s) of interest and can be used to improve the classifier in some semisupervised learning approach (see, e.g., [25]), the emphasis is on accuracy to avoid topic drift and ensure that only good archetypes are accepted, so in this case the classifier should be tuned to act very conservatively. Similarly, retrieval ranking for a "power user" of a domain-specific search engine could incorporate classifier confidence, so that improved classifier accuracy would result in better precision of the top-ten search results. On the other hand, if we want to build up an intranet document warehouse on some topic(s), we want to capture as many documents as possible and thus want to bound (or even minimize) the loss. In general, the goal would be to optimize one of the two metrics under the constraint that the other metric is bounded. Other objective functions are conceivable such as optimizing a linear combination of the two metrics, but in this work we will focus on the following goal: minimize the error under the constraint that the document loss stays above a specified lower bound. The problem addressed in this work is how to tune a classifier, or meta classifier to the goals of the application. We are especially interested in applications with relatively few (e.g., tens or hundreds but not thousands of) training documents. For topic-specific focused crawling or personalized information search this situation is the norm.

1.2 Contribution

The contributions of this work are the following:

1. We develop a *methodology for tuning* a classifier or committee-based meta classifier to the *error and loss goals* of a given application. As base methods we consider linear SVM and a simple but robust centroid separation method, but our approach to meta classification, estimation, and tuning would apply to other base classifiers (e.g., Naive Bayes) as well.

2. We show how to leverage *split-based meta methods* for the purpose of goal-oriented tuning. We investigate under which conditions it is beneficial to split a larger set of training documents into subsets for independent training of multiple classifiers whose decisions for previously unseen document are then combined in a quorum consensus manner.

3. We develop *estimators* for predicting the error and loss of a specific meta classifier setup (with specific parameter settings). We use special care in ensuring that the estimations and the training phase for the classifiers under consideration are efficient, so that interactive exploration of document collections with repeated re-training is feasible.

All of the classifiers that we consider are binary (or actually, because of the abstention option, ternary) in the sense that they make a decisions only about a document's membership in a single topic class. Multiway and hierarchical classifiers, e.g., for populating a tree-structured topic directory, can be built out of the binary building blocks [14, 3].

Our tuning methodology proceeds in three steps:

1. We first consider various base methods with default settings of tuning parameters (e.g., SVM without distance thresholding) and estimate their accuracy and loss using the empirical technique of cross validation [20]. For the split-based meta method this step is inexpensive, for it can be based on the trained classifiers for k independent partitions using the training subsets of $k-1$ classifiers as test data for the k^{th} classifier.

2. We assess the result quality of alternative parameter settings (e.g., with thresholds set to specific values) by analytically aggregating estimates from step 1. This does not require any re-training with modified parameters or on restricted document sets, and is therefore very efficient.

3. We enumerate (a subspace of) the combinatorial space of possible parameter settings (which is potentially very large for the meta methods) , using heuristics to identify candidates with good estimates. Each candidate is assessed using the estimation techniques of step 2, and the best ones are reported. Because of the heuristics for pruning the search space, this step is also fairly efficient.

The value of this approach lies in improving the engineering of document classification methods for advanced applications. We do not invent any novel foundations or algorithms for classification, but our methodology helps applications to make judicious choices for given goals in the presence of accuracy-versus-loss (and possibly other) tradeoffs.

1.3 Related Work

There is a plethora of work on text document classification using all kinds of probabilistic and discriminative models [10]. The emphasis of this body of work has been on the mathematical and algorithmic approaches, and the engineering aspects of how to cope with tradeoffs and how to tune a classifier with regard to properties of the training data and, most importantly, specific application goals have been largely neglected (exceptions being, e.g., [5, 12, 26], which address different settings and are only marginally related to our work, however).

The machine learning literature has studied a variety of meta methods such as bagging, stacking, or boosting [8, 27, 19, 15], and even combinations of heterogeneous learners (e.g., [28]). Our notion of a meta method is closest to bagging (see, e.g., [8]). Co-training [6] and its variants is another technique of which our approach may be reminiscent. However, co-training splits the feature space into conditionally independent dimensions and gives the complete training set to all classifiers, whereas our approach splits the training data itself and feature space engineering is orthogonal to our method. To our knowledge none of the prior work on bagging and related techniques has considered the parameter tuning of such methods towards application-specific quality goals.

The approach of intentionally splitting a training set for meta learning has been investigated by [11]. However, that work has focused on the efficiency versus accuracy tradeoff; so the improvements in efficiency were achieved at the expense of reduced accuracy. In contrast, our approach preserves and even improves high accuracy, and the measure that we are trading this for is document loss. The notion of loss in a ternary decision model, on the other hand, has not received wide attention. The recent paper [23] studied the accuracy-loss tradeoff in a ROC curve model (for a recommender system), but has not looked at how to systematically engineer and tune methods for judicious application choices regarding this trade-off.

For SVM classifiers some isolated tuning issues have been considered in the literature. The popular SVM Light software package by [17] provides various kinds of thresholds and variations of SVM training (e.g., SVM regression, transductive SVMs, etc.), but there is no systematic discussion of how to adjust these tuning knobs for a given application. [7] have proposed to introduce a bias for the separating hyperplane towards negative training samples, and advocated that this is beneficial when the number of positive training samples is very low.

1.4 Outline

Chapter 2 gives some technical background on document classification. Chapter 3 discusses how various methods can be tuned to be either more conservative or more speculative, thus emphasizing either accuracy or loss and presents the estimators that we use for predicting how a classifier or meta classifier will perform with given settings of its tuning parameters. At the end of that chapter we present a procedure for exploring the parameter space and choosing a goal-compatible setting. Chapter 4 presents our experimental results.

Chapter 2

Technical Background

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, 20]. 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. 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, 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., [14, 17]). Other discriminative classifiers (e.g., based on Fisher discriminants) trade off some accuracy for speed [13], but we restrict ourselves to linear SVMs in this work.

We consider also a much simpler centroid classifier that separates the centroids of positive and negative training sets with maximum margin. Obviously this method, which can be regarded as a variant of the Rocchio family of classifiers [16], is much faster than SVMs in the training phase, as its centroid and hyperplane computation are linear in the number of training documents. As for the decision phase for test documents, there is no difference between an SVM and a centroid classifier. These two linear discrimination techniques are illustrated in Figure 2.1: the separating hyperplanes are shown as a thick dashed line for SVM and a thin dotted line for Centroid.

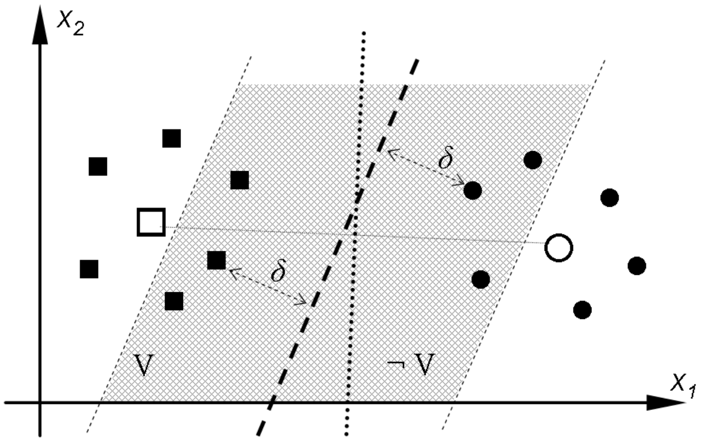

Figure 2.1: Illustration of SVM and Centroid Classifiers

None of the above classifiers consistently outperforms all other techniques. Further note that there are tradeoffs regarding the quality of the classifiers' decisions and their efficiency for training and decision-making. In addition to using one of these classifiers, multiple classifiers can be combined using a meta classifier approach [8, 27, 15], for example, by voting on the final decision (including weighted voting, see, e.g., [26]). Such a setup is interesting not only to combine different algorithmic techniques, but mostly for combining classifiers that have been trained with different training sets or for different feature spaces.

In this work we consider only binary classifiers that make a decision for a single topic, based on positive and negative training examples. Classifiers for hierarchical topic directories can be easily built out of such modules (see, e.g., [14]). Suppose a classifier is given n test documents, accepts a documents for the topic, and rejects r documents. Further assume that among the a accepted documents a^+ do indeed belong to the topic according of the intellectual assessment of a human user and a^- do not belong there. Analogously let r^- denote the number of rejected documents that do indeed not belong to the topic and r^+ the number of erroneously rejected documents. Then the measures for assessing the classifier's quality are: $precision = a^+/(a^+ + a^-)$, $recall = a^+/(a^+ + r^+)$, $accuracy = (a^+ + r^-)/n$, $error = (a^- + r^+)/n = 1 - accuracy$.

The most widely used technique for empirically measuring these quality metrics is *cross-validation* [20] on a set of independent data samples with known topic memberships (aka. class labels). Many benchmarks (e.g., ModApte for the Reuters document collection) use a ratio of 3:1 for partitioning the given data into a training set and a test set and report the performance on the test set. The partitioning is systematically varied by dividing the overall data into 4 groups and investigating each of the four choices 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* [20]. 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 re-training the classifier n times, unless special properties of the classifier's underlying model could be exploited. For SVMs [18] 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. For sufficiently precise estimations we use full-fledged leave-one-out or k-fold cross-validation .

Chapter 3

Tunable Classifiers and Meta Classifiers

3.1 Making Classifiers Restrictive and Tunable

The idea of restrictive classification is to avoid making a decision about a test document at all if that decision can be made only with relatively low confidence. So out of a given set of unlabeled data U, our method chooses a subset S of documents that are either accepted or rejected for the given topic label, and abstains on the documents in $U - S$. The quality measures precision, recall, accuracy, and error is computed on the subset S (i.e., the denominator n in the formulas of Section 2 is the cardinality of S), and we call the ratio $|U - S|/|U|$ the document *loss*.

We can use confidence measures to make simple methods restrictive. For SVMs or the Centroid method a natural confidence measure is the distance of a test document vector from the separating hyperplane. So we can tune these methods by requiring that accepted or rejected documents have a distance above some threshold, and abstain otherwise. The threshold is our tuning parameter. This approach is illustrated in Figure 2.1: all test documents that fall in the shaded region are dismissed.

Given an application-acceptable loss of L percent, we can make a classifier restrictive by dismissing the L percent of the test documents with the lowest confidence values.

3.2 Restrictive Meta Classifiers

For meta classification we are given a set $V = \{v_1, \ldots, v_k\}$ of k binary classifiers with results $R(v_i, d)$ in $\{+1, -1, 0\}$ for a document d, namely, +1 if d is accepted for the given topic by v_i, -1 if d is rejected, and 0 if v_i abstains. We can combine these results into a meta result: $Meta(d) = Meta(R(v_1, d), \ldots, R(v_k, d))$ in $\{+1, -1, 0\}$ where 0 means abstention. A family of such meta methods is the linear classifier combination with thresholding [24]. Given thresholds t_1 and t_2, with $t_1 > t_2$, and weights $w(v_i)$ for the k underlying classifiers we compute $Meta(d)$ as follows:

$$Meta(d) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) > t_1 \\ -1 & \text{if } \sum_{i=1}^{n} R(v_i, d) \cdot w(v_i) < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

This meta classifier family has some important special cases, depending on the choice of the weights and thresholds:
1) voting [8]: Meta returns the result of the majority of the classifiers.
2) unanimous decision: if all classifier give us the same result (either +1 or -1), Meta returns this result, 0 otherwise.
3) weighted averaging [26]: Meta weighs the classifiers by using some predetermined quality estimator, e.g., a leave-one-out estimator for each v_i.

The restrictive and tunable behavior is achieved by the choice of the thresholds: we dismiss the documents where the linear result combination lies between t_1 and t_2. In the rest of the work we will consider only the unanimous-decision meta classifier as the simplest of the above cases in order to demonstrate the feasibility of our approach. The approach itself carries over to more sophisticated instantiations of the meta classifier framework.

3.3 k-split Meta Classifier

One possibility to obtain a set V of k different classifiers is to split the training set T_0 into k disjoint subsets $T_1, \ldots, T_k$ and build one classifier v_i for each set T_i. Then we can easily construct a meta classifier, coined the k-split meta classifier.

Why would such a partitioning of the training data be useful at all? There are two aspects to consider:

- First, it may help to make the overall classification procedure to become more robust and reduce its generalization error. The rationale for this is that subsets of T_0 may be good enough for effectively training a classifier and that the consensus or averaging step over all classifiers then helps to counteract possible overfitting effects and thus makes the meta classifier more robust. Obviously, there are limitations to this desirable but not always achievable effect; we would expect some optimal choice of k beyond which further splitting becomes detrimental (and with very small training sets the optimal k would often be 1, i.e., not splitting at all).

- Second, the time for training a classifier often depends in some super-linear way on the cardinality of the training set T (e.g., more than quadratic with SVM). So we can improve the training efficiency by learning with subsets of T_0. This is particularly intriguing for applications that require interactive re-training such as personalized focused crawlers where user feedback after some initial results can help to calibrate the focus.

A similar approach has been studied by Chan [11], but his classifiers were not restrictive and tunable, so that he obtained efficiency gains at the expense of significantly increasing the classification error. In contrast, our method trades efficiency for loss, but keeps accuracy high or even improves it.

The naturally arising next question is how to search the tuning parameter space for the most appropriate value of k. Before we turn to this issue in Section 3.5, we first discuss, in Section 3.4, how to estimate the accuracy and loss for a fixed setting of k and the other tuning parameters.

A natural alternative to splitting the training set into disjoint partitions would be to allow overlapping partitions and not necessarily using all training documents. This could be easily implemented using *random sampling* to create a training partition, where the number k of partitions and their size m in terms of training documents are tuning parameters. Note that the sampling procedure is allowed to select duplicates in different partitions, hence the name random resampling. This method has the same need for parameter tuning that the k-split approach faces. We will study random resampling as a competitor to k-split meta classification in our experiments.

3.4 Estimators for Accuracy and Loss

3.4.1 Estimators for Single Classifiers

For tuning the use of a classifier in an application it is desirable to have a priori estimators for the accuracy and loss of the classification method given its training data. While such estimators entirely based on analytical models are conceivable, our experience has led us to believe that some degree of empirical estimation is inevitable for sufficiently accurate predictions, at least as a baseline upon which analytical reasoning can build. We will see that good estimators for the k-split meta classifiers can be constructed without (repeatedly) performing time-consuming leave-one-out estimation on the full training set. Likewise, for restrictive variants of the base classifiers SVM and Centroid we can avoid having to fully recompute (with repeated invocation of the training procedure) the leave-one-out estimator for every setting of the threshold parameters.

SVM requires an initial full-fledged leave-one-out estimator which involves n times retraining, where n is the cardinality of T_0, or at least an initial cross-validation with m partitions of size n/m and m times retraining. For the Centroid method a more efficient technique is feasible: we compute the two centroids that result from all documents in T_0. For each leave-one-out choice we incrementally "subtract" the left-out test document yielding the adjusted centroids and the adjusted hyperplane in time $O(1)$ and classify the document. So just like the entire training procedure for Centroid has run-time $O(n)$, the complete leave-one-out estimator (over all n choices) can be carried out in the same $O(n)$ time.

For the k-split meta methods we will need to estimate the accuracy of each of the underlying classifiers trained with subsets $T_1, \ldots, T_k$. We can exploit this situation by estimating the accuracy $acc(T_i)$ via cross-validation on the complementary subsets $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_k$. For a robust estimator we take the average over all $acc(T_i)$ values. In the experiments we refer to this estimator as the ***CV estimator***.

Finally, the value of $acc(T_k)$ can be estimated by

$$acc(T_k) \sim \frac{1}{k} \sum_{i=1}^{k} acc(T_k^i) \qquad (3.2)$$

We notice that this step does not require expensive leave-one-out estimations, and it is carried out over training sets that are significantly smaller than T_0. However, depending on the application situation we may like to avoid having to re-run the estimations on all candidate choices of k, as this would still require the training and cross-validation of k classifiers for every value of k. Rather we would prefer analytically reasoning, with very low computational cost, about the quality degradation that we are likely to see when training with some subset T_i instead of T_0. Our rationale is the following: the accuracy for training with T_i, $acc(T_i)$, should really be the same as for T_0 itself, $acc(T_0)$, if T_i is still a representative sample for the same stochastic feature distribution for which T_0 is our baseline sample; conversely, if the distribution in T_i deviates significantly from that of T_0 we should see some degradation in accuracy. Deviations $diff$ between probability distributions can be measured by the Kullback-Leibler divergence (relative entropy) $KL(T_i, T_0)$.

We performed extensive experiments to study the viability of such heuristic estimators, and also looked at variations and other entropy- or χ^2-based measures for deviation. It turned out that the correlation coefficient between our $diff$ metric and the resulting accuracy was consistently above 0.9 for a number of experiments with k-splits on $Newsgroups$ and $Reuters$ datasets; so $acc(T_i)$ does indeed deteriorate approximately linearly with increasing $KL(T_i, T_0)$. Our estimator for

$acc(T_i)$ is based on a simple linear regression using two (or possibly more) data points for acc and KL derived from two (or more) k-split partitionings. This way we fit the coefficients a and b in the following equation:

$$acc(T_i) = a \cdot KL(T_i, T_0) + b \tag{3.3}$$

Note that this procedure requires $acc(T_i)$ estimators, using the averaged cross-validation technique outlined above, for only two choices of k, and we can choose relatively large values of k (e.g., 10 and 20) so that the training and cross-validation of k classifiers on relatively small training sets is fairly inexpensive. Further note that this computation is carried out outside the actual search procedure for the best possible k, and that we do not need to be particularly judicious about choosing our two sample points for k. We refer to this alternative estimator as the *KL estimator* in our experiments.

The KL divergence itself is estimated by

$$KL(T_i, T_0) = \sum_j f_j(T_i) \cdot log_2 \frac{f_j(T_i)}{f_j(T_0)} \tag{3.4}$$

where $f_j(S)$ is the relative frequency of documents in document set S that contain feature j (i.e., a word stem). This computation is linear in the cardinality of the training set T_i. For a robust estimate we actually average the KL values over all subsets T_i of a given k-split partitioning.

One complication that arises with this approach is that positive and negative training examples follow radically different distributions, and we have to make sure that our KL-based distance measure captures this. To this end we actually compute the KL divergence for positive and negative samples separately and take their maximum for the scaling factor in the accuracy prediction:

$$diff(T_k, T_0) = max\{KL_{pos}(T_k, T_0), KL_{neg}(T_k, T_0)\} \tag{3.5}$$

$$acc(T_k) = a \cdot diff(T_k, T_0) + b \tag{3.6}$$

Figures 3.1, 3.2 illustrate the viability of this analytic approximation technique.

The run-time cost of constructing the accuracy estimator for a basic SVM classifier, using leave-one-out estimation, is between $O(n^3)$ and $O(n^4)$ (n times retraining each with complexity of typically between $O(n^2)$ and $O(n^3)$) with n denoting the cardinality of the complete training set T_0. With $m - fold$ cross-validation instead of leave-one-out the cost is between $O(mn^2)$ and $O(mn^3)$. The $\xi\alpha$ estimator of [18] would require only a single training procedure (thus typically running in time $O(n^2)$ to $O(n^3)$), but is way too crude to be useful in our framework. For the Centroid method, training and leave-one-out estimation are

Test	pos. samples	neg. samples	k	correlation KL - acc
Reuters:				
money-fx vs. acq	700	700	1..20	-0.93
earn vs. trade	500	500	1..10	-0.92
Newsgroups:				
rec.autos vs. rec.motorcycles	700	700	1..20	-0.97
talk.politics.guns vs. talk.politics.mideast	700	700	1..20	-0.98

Figure 3.1: Correlation between $diff(T_i, T_0)$ and $accuracy(T_i)$

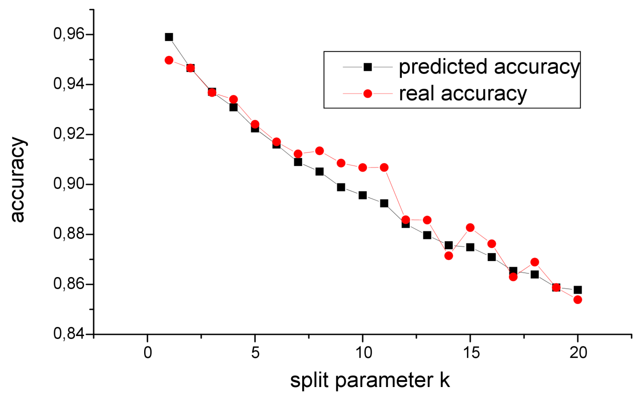

Figure 3.2: Accuracy Prediction by KL for Newsgroups Topics ”rec.cars” and ”rec.motorcycles”

combined and have only $O(n)$ run-time cost. Finally, for estimating the accuracy of an SVM classifier trained on some subset T_i we need time $O(n)$ for the KL computation (needed only in the KL estimator) plus the a priori probing cost, with training, cross-validation, and regression (needed in both the CV and the KL estimators), which in total is between $O((n/k)^2)$ and $O((n/k)^3)$ where k is the smaller one of our two probing points. With k being 10 or larger, this is a substantial savings compared to the basic SVM estimator.

3.4.2 Estimators for Restrictive Classifiers

For a basic classifier like SVM or Centroid, the leave-one-out decision step gives us a set of tuples
$(d, confidence, isCorrect)$ where d is the left-out-document, $confidence$ is the classification confidence (i.e., hyperplane distance or probability), and $isCorrect$

is a Boolean value that tells us if the classifier trained with the $n-1$ remaining documents correctly classifies d (value 1) or not (value 0). Note that the confidence value may not be exported by the built-in estimators of existing software packages such as SVM light; in this case we either need to re-implement the leave-one-out estimator on top or modify the software or switch to a package that makes these values easily accessible.

Given a threshold for the confidence, it is now easy to compute the adjusted accuracy. We only need to restrict the summation over the $isCorrect$ values that form the basis of the average accuracy estimation to those $isCorrect$ values for which confidence exceeds the threshold, and the denominator for accuracy then is the count of all tuples with confidence higher than the threshold. Likewise, loss simply is the count of the tuples with confidence below the threshold divided by n.

3.4.3 Estimators for k-split Meta Classifiers

Now we explain how to construct an estimator for loss and accuracy (or equivalently error) of a k-split meta classifier with unanimous decision, given the estimators for the underlying classifiers $\{v_1, ..., v_k\}$.

Let $T_0 = \{T_1 \cup \ldots \cup T_k\}$ be our partitioning of the overall training data and let v_i be the classifier trained on T_i. We associate a Bernoulli random variable X_i with each v_i, where $X_i = 1$ if v_i classifies a document correctly, 0 otherwise.

A simple, but unfortunately oversimplified, approach would be to estimate the accuracy, $P[X_i = 1]$, or equivalently error, $P[X_i = 0]$, for every v_i by 1) assuming that X_i and X_j are pairwise independent for different v_i and v_j, and 2) using the KL-based prediction model of Section 3.4.1. Unfortunately, the independence assumption leads to fairly inaccurate estimations for both accuracy and document loss. Instead we pursue a more sophisticated approach that takes the correlations between classifiers into account.

To this end we run an additional cross-validation upfront. We consider three mutually disjoint subsets T_1, T_2, T_3 of T_0 where T_1 and T_2 serve to train classifiers v_1 and v_2 and T_3 is held-back test data to assess v_1 and v_2 such that neither v_1 nor v_2 has seen this test data before. Note that the three subsets may be easily derived by a random partitioning of a medium-sized random subset of T_0; we do not need to take all of T_0 into consideration thus reducing the computational cost. The training procedure gives us data points (x_1, x_2) for the joint distribution of (X_1, X_2). Thus we obtain an estimator for the covariance

$$cov(X_1, X_2) = \frac{1}{n-1} \cdot \sum_j (x_1 - \overline{x}_1)(x_2 - \overline{x}_2) \qquad (3.7)$$

where n is the number of data points in T_3 and $\overline{x}_1, \overline{x}_2$ are the means of the marginal distributions of X_1 and X_2. From basic probability theory it follows that

$$P(X_1 = 1 \wedge X_2 = 1) = cov(X_1, X_2) + P(X_1 = 1) * P(X_2 = 1) \tag{3.8}$$

This procedure requires training the classifiers v_1 and v_2, but we do not need any further expensive steps for $k > 2$ by making two assumptions:

1. For any two subsets T_i, T_j in any possible k-split partitioning, the covariance is the same as $cov(X_1, X_2)$ computed above. So the covariance estimator for $k = 2$ can be reused without additional computations. Below we therefore refer to the covariance estimator simply as cov without any subscripts or arguments.

2. In a k-split partitioning we consider only the dependencies between v_i and v_{i+1} and postulate that all other pairs v_i and v_j can be considered as independent.

Assumption 1 is justified as long as all subsets T_i in a k-split partitioning are reasonably representative samples for the original data T_0.

We can justify Assumption 2 by using a tree dependence model, which is a well known approximation method in probabilistic IR ([22]): We define a *Dependence Graph* $G = (V, E)$ where V consists of the Bernoulli Variables X_i, and which contains for all X_i, X_j $(i \neq j)$ an undirected edge $e(X_i, X_j)$ with weight $w(e(X_i, X_j)) = cov(X_i, X_j)$. We approximate the Dependence Graph by a Maximum Spanning Tree $G' = (V, E')$ which maximizes the sum of the edge weights. The nodes in G' with no edges in between are considered as independent. So we obtain:

$$\begin{gathered} P(X_1 = x_1, \ldots, X_k = x_k) = \\ P(X_{root} = 1) \prod_{(i,j) \in E'} \frac{P(X_i = x_i, X_j = x_j)}{P(X_i = x_j)} \end{gathered} \tag{3.9}$$

where X_{root} is the root node of the tree G' and $x_i \in \{0, 1\}$. Because $w(e(X_i, X_j)) = cov$ (where cov is a constant according to Assumption 1) we can w.l.o.g. choose X_1 as the root node and the edges (X_i, X_{i+1}) as tree edges. This corresponds to Assumption 2.

Now we have:

$$\begin{aligned} P(X_1 = 1, \ldots, X_k = 1) &= P(X_1 = 1) \prod_{i=1}^{k-1} P(X_{i+1} | X_i) \\ &= P(X_1 = 1) \prod_{i=1}^{k-1} \frac{P(X_i = 1, X_{i+1} = 1)}{P(X_i = 1)} \end{aligned} \tag{3.10}$$

By considering equation 3.8 and Assumption 1 we obtain:

$$P(X_1 = 1, \ldots, X_k = 1) = \\ P(X_1 = 1) \prod_{i=1}^{k-1} \frac{P(X_i = 1)P(X_{i+1} = 1) + cov}{P(X_i = 1)} \tag{3.11}$$

Analogously we obtain $P(X_1 = 0 \wedge ...X_k = 0)$.

Estimators for $P(X_i = 1)$ and $P(X_i = 0)$ (i.e., for accuracy and error of the single classifiers v_i) can be determined by either the CV or the KL estimator explained in Section 3.4.1.

Finally we can substitute these results in the following formulas for the loss estimator

$$loss(Meta(v_1, ..., v_k)) = 1 - P(X_1 = ... = X_k) \\ = 1 - (P(X_1 = 1, \ldots, X_k = 1) + P(X_1 = 0, \ldots, X_k = 0)) \tag{3.12}$$

and the error and accuracy estimator

$$\begin{aligned} & error(Meta(v_1, ..., v_k)) \\ & \quad = P(X_1 = 0...X_k = 0 | X_1 = ... = X_k) \\ & \quad = \frac{P(X_1 = 0...X_k = 0)}{P(X_1 = 1...X_k = 1) + P(X_1 = 0...X_k = 0)} \end{aligned} \tag{3.13}$$

$$accuracy(Meta(v_1, ..., v_k)) = 1 - error(Meta(v_1, ..., v_k)) \tag{3.14}$$

The point of these analytic derivations is that we can start with a limited set of empirically determined quality measures (based on leave-one-out and cross-validation techniques) and can assess candidates for a k-split partitioning meta classifier in an efficiently computable, solely analytic manner without further re-training. This capability is crucial for an efficient traversal of the space of possible k-split partitionings, as discussed in the next section.

In principle, the same estimation procedure can be carried over to a meta classifier based on k-fold random resampling. The only potential caveat is that the independence assumption for more than two partitions could be more critical in the case of overlapping partitions. We will discuss the estimators' accuracy in our experiments in Section 4.

3.5 Parameter Search Heuristics

Our goal is to minimize the error subject to the constraint that the loss is bounded by some application specific threshold.

For the basic classifiers, SVM, Centroid, and Naive Bayes, we simply perform a binary search over their control parameters, which are either hyperplane distance or probability threshold. This way we can easily find their best parameter settings. Varying these parameters does not require retraining the classifier with the above range of methods. Our estimation techniques presented in Section 3.4 allow us to predict loss and accuracy from the leave-one-out predictions for the non-restrictive baseline cases (with all thresholds set to 0).

For the k-split meta classifier, the parameter search space is much larger. We need to decide

- into how many partitions we split T_0,
- how we divide the positive training samples among the resulting partitions $T_1, \ldots, T_k$, and
- how we divide the negative training samples (note that we may consider treating positive and negative samples differently for they may vary significantly in cardinality).

Obviously, there is some danger of combinatorial explosion here (e.g., exponentially many options in the cardinality of T_0); so we restrict ourselves to a fairly simple greedy heuristics .We only consider splitting T_0 into k equally sized subsets (plus/minus one if exactly equal sizes were impossible) and we always preserve the original ratio of positive to negative samples. For dividing the training samples themselves, we only consider random partitioning, which is both efficient and guarantees that the resulting training subsets reasonably preserve the statistical properties (e.g., feature distributions) of the original data. More sophisticated alternatives (e.g., dividing based on information-theoretic models) are subject to future research.

The meta classifiers that we tentatively construct consider a 1-split, a 2-split, a 3-split, and so on, and this leads to a fairly simple search procedure:

```
Input: lower bound threshold for loss
Output: best choice of k,
         training subsets T1, ..., Tk
Algorithm:
k=1; T1 = T0; best := 1;
estimate loss(1) for
  the meta classifier with k=1;
while (loss(k) < threshold) {
    randomly partition T0 into k subsets
  estimate loss(k+1);
```

```
estimate error(k+1);
if (loss(k+1) < threshold) {
  if (error(k+1) < best) best = k+1;};
k = k+1; };
```

This procedure exploits that loss is almost certainly monotonically increasing with increasing k, not only with the unanimous-decision variant that we are using but also with most other variants of our meta classifier framework. For unanimous decision our experiments even showed that accuracy is monotonically increasing with increasing k; so in this particular case our procedure returns the maximum k such that the estimated loss is still acceptable.

Chapter 4

Experimental Results

4.1 Setup

We performed a series of experiments with real-life data from the newsgroups collection at [1] and the Internet Movie Database (imdb) at [2]. For our experiments we selected subsets of 250 and 500 documents for training (i.e., a small and a medium-sized training set), and tested our various methods with the remaining held-out data. We performed two kinds of experiments:

- *baseline experiments* investigated the classification error as a function of the document loss, and compared the measured results to the predictions by our various estimators, and
- *use-case experiments* investigated the error and loss as functions of the application goal for the maximum tolerable loss, and studied to what extent we could indeed automatically tune the methods to a given loss threshold.

All experiments were based on software written in Java, except for the base classifiers SVM, where we used SVM light, and Centroid, which we implemented in C++. We compared the following methods and meta methods:

- Restrictive variants of the base classifiers SVM and Centroid.
- The k-split meta method, based on (non-restrictive variants of) either SVM or Centroid, where k was varied from 1 to 16.
- Two variants of the k-fold random resampling meta method, one with replacement of drawn samples (variant *A*) and one without replacement for the same training partition (variant *B*). Each one of these was based on either SVM or Centroid. k was varied from 1 to 16, and the number m of samples per training partition was set to $1.5 * (\#training\ docs)/k$ (e.g., 375 for a total training set of 500 documents and $k = 2$ partitions).

4.2 Results

Our systematic experiments capture the behavior of classifiers and meta classifiers for pairs of topics such as ”Drama vs. Horror” for imdb data or ”rec.autos vs. rec.motorcycles” for the newsgroups data. For each data sets we identified all topics with sufficiently many documents (> 900 for newsgroups, > 550 for imdb) for training and testing. These were 17 topics for newsgroups and 5 for the genres of imdb documents. and randomly choose 100 topic pairs from newsgroups and 10 from imdb. We computed micro-averaged results for these topic pairs.

4.2.1 Baseline Experiments

Figure 4.1 illustrates the loss-error tradeoff for the two base methods SVM and Centroid, the k-split meta method and the two variants of random resampling. A typical observation is that willing to lose up to 20 percent of the documents by abstaining on low-confidence decisions could reduce the classification error from about 10 percent down to less than 5 percent. This behavior was consistent across all methods, with little variation of the quantitative results.

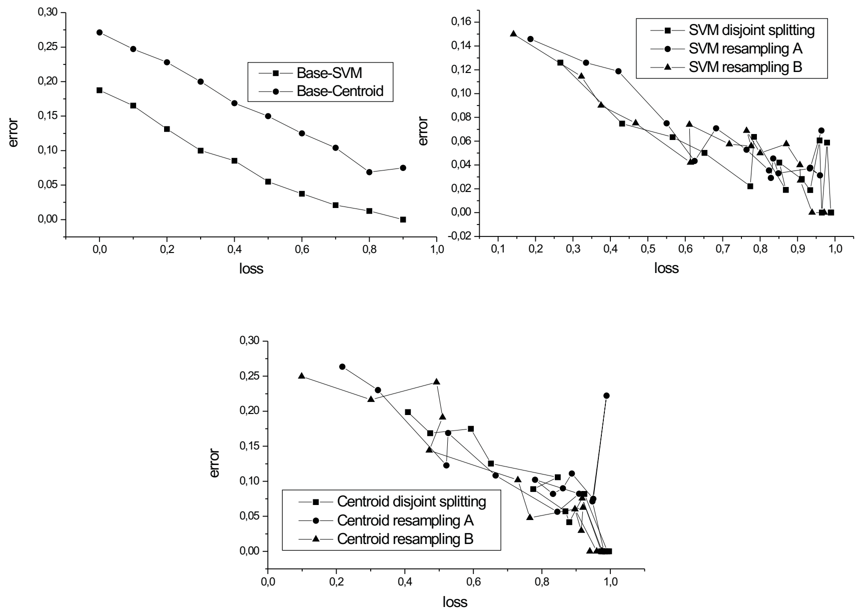

Figure 4.1: Loss-Error Tradeoff for ”Drama vs. Horror”

Figure 4.2 shows detailed comparisons of the k-split vs. k-fold random resampling meta methods for the case *drama* vs. *horror* from the imdb data , as a function of the number of partitions, k, and the number of training documents.

Similar results for other examples of class pairs are shown in Figures B.1 through B.3 in the appendix. The microaveraged values over the whole newsgroup and the imdb dataset show that the major trends were the same (see Figures A.1 and A.2 in the appendix).

	Disjoint k-Split						**Resampling A**		**Resampling B**		**Method**
#Partitions	Loss	estLossCV	estLossKL	Error	estErrorCV	estErrorKL	Loss	Error	Loss	Error	**#TrainDocs**
2	0,266	0,247	0,462	0,126	0,11	0,043	0,205	0,157	0,113	0,156	
4	0,566	0,581	0,688	0,063	0,046	0,009	0,473	0,088	0,399	0,102	SVM
8	0,869	0,91	0,922	0,019	0,008	0,001	0,809	0,046	0,726	0,05	250
16	0,99	0,999	0,999	0	0	0,002	0,989	0	0,959	0,06	
2	0,179	0,215	0,311	0,082	0,099	0,041	0,141	0,096	0,115	0,096	
4	0,418	0,54	0,514	0,061	0,028	0,011	0,291	0,075	0,312	0,056	SVM
8	0,744	0,871	0,813	0,023	0,002	0,001	0,609	0,045	0,541	0,026	500
16	0,947	0,995	0,996	0	0	0	0,85	0,039	0,862	0,043	
2	0,409	0,29	0,635	0,199	0,311	0,133	0,141	0,255	0,09	0,234	
4	0,593	0,778	0,849	0,175	0,1	0,05	0,686	0,092	0,411	0,168	Centroid
8	0,869	0,948	0,976	0,057	0,03	0,01	0,84	0,063	0,79	0,077	250
16	0,996	0,999	0,1	0	0,003	0,004	0,959	0	0,983	0,071	
2	0,494	0,192	0,655	0,18	0,368	0,18	0,1	0,225	0,056	0,24	
4	0,647	0,867	0,865	0,075	0,059	0,08	0,309	0,243	0,321	0,182	Centroid
8	0,809	0,737	0,976	0,0462	0,012	0,015	0,676	0,1	0,65	0,118	500
16	0,95	0,998	0,999	0	0,004	0,001	0,898	0	0,856	0,041	

Figure 4.2: Different Restrictive Splitting and Resampling Methods for ”Drama vs. Horror”

Most importantly, the figures show that both analytical estimators for document loss (CV based on cross-validation over the k partitions, and KL based on the Kullback-Leibler divergence for estimating how quality degrades with smaller training sets) turned out to be fairly accurate and slightly conservative. The estimator for error, on the other hand, turned out to be optimistically biased and moderately accurate at best. Note, however, that our parameter search procedure (see Section 3.5) does not depend on the absolute accuracy of the error estimator but only on its relative ordering of the quality of different k values. In contrast, the estimator for loss is critical as it determines whether a certain k value is acceptable or not. The experiments show that the loss estimator is indeed sufficiently accurate.

When comparing k-split vs. random resampling, we see that resampling (both variants) usually led to lower loss but often to significantly higher error for the same number of partitions. But the affordable loss can be effectively controlled by our tuning procedure; so for the same tolerable loss, k-split may simply use a slightly smaller number of partition and would still usually be at least as good as resampling in terms of error. Moreover, it is much easier to construct accurate and computationally inexpensive estimators for the k-split meta method than for the resampling approach. Therefore, we will disregard resampling in our discussion of use-case results in the next subsection.

Both loss and error results were, not surprisingly, significantly improved when increasing the total number of training documents from 250 to 500, but the relative behavior of the methods remained unchanged. Recall from Section 1 that we are primarily interested in situations with few training data. Finally, the simple Centroid method as a basis for the k-split and resampling meta methods performed amazingly well relative to the theoretically much superior SVM classifiers. However, the Centroid-based meta methods did exhibit a non-negligible penalty in terms of classification error.

4.2.2 Use-Case Experiments

In a second line of experiments we studied how well our procedure for goal-oriented tuning was indeed able to determine practically viable parameter settings. Again we will focus the discussion on the results for *drama* vs. *horror* given in Figure 4.3 (see Figures B.4 through B.6 in the appendix for other examples; see Figures A.3 and A.4 in the appendix for the micro-averaged results).

Here we gave ourselves a goal for the acceptable loss threshold, ran our automatic tuning procedure for the various methods, and finally evaluated the tuned methods on the held-out validation data. We compare only the k-split meta method against the base methods SVM and Centroid (for random resampling did not perform better than k-split and poses more difficulties for accurate estimators).

Loss Threshold (k)	**Disjoint k-Split**				**BaseMethod**			**Method #TrainDocs**
	Loss	estLossCV	Error	estErrorCV	Loss	Error	estError	
0,1 (1)	0	0	0,188	–	0,1	0,165	0,131	SVM 250
0,3 (2)	0,266	0,247	0,126	0,11	0,3	0,1	0,094	
0,5 (3)	0,431	0,465	0,075	0,08	0,5	0,055	0,06	
0,7 (4)	0,566	0,581	0,063	0,046	0,7	0,021	0,04	
0,9 (7)	0,784	0,85	0,064	0,015	0,9	0	0,06	
0,1 (1)	0	0	0,126	–	0,1	0,098	0,118	SVM 500
0,3 (2)	0,179	0,215	0,082	0,099	0,3	0,055	0,079	
0,5 (3)	0,306	0,368	0,055	0,054	0,5	0,018	0,058	
0,7 (5)	0,509	0,62	0,036	0,019	0,7	0,02	0,03	
0,9 (9)	0,679	0,871	0,046	0,003	0,9	0	0,03	
0,1 (1)	0	0	0,271	–	0,1	0,247	0,3	Centroid 250
0,3 (2)	0,409	0,29	0,199	0,311	0,3	0,2	0,3	
0,5 (3)	0,474	0,471	0,169	0,212	0,5	0,15	0,255	
0,7 (3)	0,474	0,470	0,169	0,212	0,7	0,104	0,179	
0,9 (7)	0,775	0,881	0,089	0,057	0,9	0,075	0,088	
0,1 (1)	0	0	0,256	–	0,1	0,239	0,134	Centroid 500
0,3 (3)	0,206	0,244	0,207	0,253	0,3	0,176	0,113	
0,5 (3)	0,206	0,244	0,207	0,253	0,5	0,118	0,094	
0,7 (3)	0,206	0,244	0,207	0,253	0,7	0,069	0,078	
0,9 (4)	0,647	0,867	0,075	0,059	0,9	0,118	0,118	

Figure 4.3: Tuning Results for "Drama vs. Horror"

The measured results show that k-split always met the
application-provided loss goal and achieved a fairly low error rate. The first column of the table in Figure 4.3 shows the optimal values of k that our tuning procedure determined for a given loss threshold.

We see that these cannot be easily found by simple rules of thumb, and, moreoever, vary quite a bit depending on the number of training documents, the choice of the base classifier, and the datasets themselves.

The base methods can exactly control the loss, and thus inherently performed much better in terms of the actual loss. In terms of error, the base Centroid method was also superior to the Centroid-based k-split meta method, but lost against the SVM-based k-split meta method (for the same loss goal). For example, for a loss threshold of 0.3 and with 250 training documents, base Centroid achieved exactly this loss and an error of 20 percent, whereas the auto-tuned k-split meta method with underlying SVM achieved a loss of 26.6 percent (because of the conservatively biased loss estimator) and an error of 12.6 percent. In situations where the low error rate is critical (e.g., when automatically selecting new, initially unlabeled, training documents for a focused crawler), this gain may be important.

The SVM-based k-split meta method was still outperformed by the base SVM method (albeit sometimes only by a small margin), but the key point here is that base SVM requires expensive cross-validation or even leave-one-out validation for its estimator in order to calibrate its thresholding to the application-given loss goal. Figure 4.4 compares the training times for the various meta methods and base methods that are necessary to construct the loss and error estimators. Base SVM without any partitioning would often be considered prohibitively expensive for interactive applications (i.e., when the training and estimation procedures themselves are part of user-perceived response times).

The bottom line is that both k-split SVM and base Centroid are practically viable methods for which our estimators and tuning procedure work fairly well.

partitions	k-Split SVM	k-Split Centroid
1	19,64	0,67
2	8,44	0,44
4	4,12	0,32
8	2,0	0,2
16	1,12	0,19

L-fold	SVM	Centroid
2	8,22	0,45
3	25,65	1,05
4	52,88	1,64
5	69,55	2,3

L-1-O SVM	L-1-0 Centroid
670064,64	2,64

Figure 4.4: Training and Estimation Times for k-Split, L-fold Cross Validation, and Leave-one-out (in Seconds)

4.3 Summary and Lessons Learned

Summary We have gained new insights into the fundamental tradeoffs between *classification accuracy*, *loss in ternary classification*, *amount of necessary training data*, and *training efficiency*.

The experiments clearly show that our analytical estimators are useful for driving the tuning procedure. In terms of absolute quality and meeting the error and loss goals of the application, the k-split meta methods are very competitive. The restrictive variant of the basic SVM classifier still is usually the best classifier, but its training and tuning time is an order of magnitude higher than the corresponding cost of the k-split meta method.

Lessons Learned Our findings suggest the following guidelines for selecting the most appropriate method and tuning it towards a given application setting:

- When only few training documents are available and the time needed for training is uncritical, then the *restrictive variant of SVM* is the method of choice. Its accuracy is usually the best among all competitors, and it can be easily tuned, as shown in this paper, for a specified loss tolerance. A typical situation where these properties are important is for classifying query results in a personalized search engine. Such a system would be trained with few documents that reflect a user's interest profile, but this is performed offline, i.e., not within the response time of a query, so that training efficiency is not critical. At query time, it may be desirable for an advanced user to tolerate a certain loss and see only query results that clearly fall into the given scope of interest.

- When only few training documents are available and the efficiency of training is critical, then the *restrictive Centroid variant* is the method of choice. Its accuracy is worse than for the other methods, but may still be acceptable to the application. Moreover, it can be easily tuned with regard to the accuracy-loss tradeoff. Last but not least, its training time is drastically shorter than that of an SVM classifier on the same training data. A situation where these arguments in favor of restrictive Centroid apply is within an expert user's focused crawler with online re-training [25].Here a classifier may start with very few, manually selected, training documents that capture a user's interest profile. To incrementally improve the focused crawler, semisupervised learning techniques can be used to automatically select additional training data among the automatically classified documents and to dynamically re-train the classifier. As this additional training data comes with a non-negligible error probability, it may be important to accept only the highest-confidence documents and tolerate a certain loss. As the re-training is performed within an interactive session with an expert user, its efficiency is highly critical.

- When a medium or large number of training documents are available and training efficiency is a critical issue, then the *k-split meta method with SVM as base method* is most appropriate. Its training time is substantially shorter than for restrictive SVM, and its accuracy is competitive to SVM, albeit not quite as good, and significantly better than that of restrictive Centroid. With regard to the accuracy-loss tradeoff, the k-split meta method can be automatically tuned using the estimators developed in this paper. A situation where this case arises is in generating, maintaining, and organizing Web information portals or digital libraries on special topics (e.g., for hiking and climbing, for music styles, for cultural heritage projects, etc.). Here a classifier can be used to aid a human portal administrator or digital-library curator to organize information more easily, quickly, and accurately. In such an environment, there should be a sufficient number of initial training documents, but the administrator may occasionally select additional high-quality documents for re-training, e.g., when she decides that a topic with (too) many documents should be subdivided into more specialized topics. If such reorganizations are to be carried out interactively, allowing the administrator to intellectually validate the resulting classification quality, then efficient re-training is a must and flexible control over accuracy versus loss is important.

Chapter 5

Conclusion

In this work we have started out to investigate the engineering and, in particular, tuning issues of using automatic classifiers for text document categorization. We have developed a constructive and practically efficient methodology for tuning a repertoire of classifiers and meta methods to the application's specific goals in terms of classification error and document loss. A key element in our approach has been to devise analytic estimators that can predict the error and loss for a given parameter setting sufficiently accurately. Although our techniques are anchored upon empirical leave-one-out or cross-validation estimators on the underlying training data to some extent, we have taken great care to avoid computationally expensive steps that would involve repeated retraining. While the approach applies to a wider repertoire of classifiers, we have especially worked on the k-split meta classifier based on SVM or Centroid, and we have experimentally shown that we can tune these methods to meet the application goals with a classification error that is competitive to that of conservative SVM variants at drastically reduced training cost.

Our ongoing and future work includes a number of relatively obvious directions like 1) generalizations towards weighted quorum-consensus meta methods, and 2) application studies, especially in the context of focused crawling and personalized data exploration both of which can benefit from interactive re-training and good quality estimators.

The work presented here is embedded in the BINGO! project [25], a toolsuite for building information portals and specialized search engines. Our long-term objective is to better understand the engineering of how to incorporate, adapt, and tune machine learning methods into more intelligent next-generation systems for information organization and search.

Bibliography

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

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

[3] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

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

[5] H. Blok, D. Hiemstra, S. Choenni, F. Jong, H. Blanken, and P. Apers. Predicting the cost-quality trade-off for information retrieval queries: Facilitating database design and query optimization. *CIKM*, 2001.

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

[7] 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.

[8] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

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

[10] S. Chakrabarti. *Mining the Web*. Morgan Kaufmann, 2003.

[11] P. Chan. An extensible meta-learning approach for scalable and accurate inductive learning. *PhD thesis, Department of Computer Science, Columbia University, New York*, 1996.

[12] S. Cronen-Townsend, Y. Zhou, and W.B. Croft. Predicting query performance. *SIGIR*, 2002.

[13] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, 2000.

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

[15] Y. Freund. An adaptive version of the boost by majority algorithm. *Workshop on Computational Learning Theory*, 1999.

[16] T Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for

text categorization. *ICML*, 1997.

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

[18] T. Joachims. Estimating the generalization performance of an SVM efficiently. *ECML*, 2000.

[19] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *FOCS*, 1989.

[20] C.D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[21] T. Mitchell. *Machine Learning*. McGraw Hill, 1996.

[22] C. Van Rijsbergen. A theoretical basis for the use of co-occurence data in information retrieval. *Journal of Documentation, 33:2, pp. 106-119*, 1977.

[23] A.I. Schein, A. Popescul, L.H. Ungar, and D.M. Pennock. Methods and metrics for cold-start recommendations. *ACM Conference on Research and Development in Information Retreival*, 2002.

[24] S. Siersdorfer and S. Sizov. Construction of feature spaces and meta methods for classification of Web documents (in german). *Conference on Database Systems for Business, Technology and Web (BTW)*, 2003.

[25] S. Sizov, M. Biwer, J. Graupmann, S. Siersdorfer, M. Theobald, G. Weikum, and P. Zimmer. The BINGO! system for information portal generation and expert Web search. *Conference on Innovative Systems Research (CIDR)*, 2003.

[26] H. Wang, W Fan, P.S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. *SIGKDD*, 2003.

[27] D.H. Wolpert. Stacked generalization. *Neural Networks, Vol. 5, pp. 241-259*, 1992.

[28] H. Yu, K.C. Chang, and J. Han. Heterogeneous learner for Web page classification. *ICDM*, 2002.

Appendix A

Aggregated evaluation results

#Partitions	Disjoint k-Split avg(error)	Disjoint k-Split avg(loss)	Resampling A avg(error)	Resampling A avg(loss)	Resampling B avg(error)	Resampling B avg(loss)	Method #TrainDocs
2	,03	,052	,036	,038	,037	,024	
4	,018	,131	,023	,101	,024	,09	SVM
8	,008	,26	,013	,2	,013	,188	250
16	,003	,456	,005	,368	,005	,351	
2	,024	,04	,027	,03	,029	,02	
4	,016	,097	,019	,076	,02	,066	SVM
8	,008	,187	,011	,15	,013	,141	500
16	,004	,331	,006	,262	,006	,256	
2	,057	,05	,062	,04	,067	,02	
4	,038	,135	,045	,111	,049	,084	Centroid
8	,023	,314	,029	,248	,03	,206	250
16	,014	,59	,017	,49	,017	,445	
2	,057	,032	,059	,03	,062	,016	
4	,041	,09	,048	,073	,048	,058	Centroid
8	,025	,194	,031	,153	,033	,132	500
16	,014	,397	,019	,296	,019	,276	

Figure A.1: Micro-Averaged Results for Different Restrictive Splitting and Resampling Methods for the Newsgroups Data Set

#Partitions	Disjoint k-Split avg(error)	Disjoint k-Split avg(loss)	Resampling A avg(error)	Resampling A avg(loss)	Resampling B avg(error)	Resampling B avg(loss)	Method #TrainDocs
2	,113	,193	,139	,133	,155	,078	
4	,068	,399	,103	,315	,092	,288	SVM
8	,026	,629	,046	,533	,048	,51	250
16	,005	,791	,017	,739	,016	,729	
2	,176	,171	,142	,105	,166	,067	
4	,14	,359	,119	,246	,117	,228	SVM
8	,066	,6	,068	,409	,072	,413	500
16	,018	,789	,016	,629	,017	,612	
2	,105	,149	,119	,141	,13	,063	
4	,067	,376	,088	,291	,085	,25	Centroid
8	,027	,623	,045	,545	,051	,47	250
16	,008	,895	,026	,816	,024	,771	
2	,125	,098	,126	,071	,151	,032	
4	,085	,264	,091	,215	,095	,176	Centroid
8	,027	,467	,05	,395	,067	,345	500
16	,012	,66	,018	,631	,016	,583	

Figure A.2: Micro-Averaged Results for Different Restrictive Splitting and Resampling Methods for the imdb Data Set

Loss Threshold	Disjoint k-Split Avg(error)	BaseMethod Avg(error)	Method #TrainDocs
0.1	,033	,021	
0.3	,017	,009	SVM
0.5	,011	,005	250
0.7	,007	,003	
0.9	,004	,002	
0.1	,024	,014	
0.3	,013	,005	SVM
0.5	,009	,003	500
0.7	,006	,002	
0.9	,005	,001	
0.1	,059	,074	
0.3	,0403	,045	Centroid
0.5	,027	,018	250
0.7	,017	,01	
0.9	,014	,006	
0.1	,052	,068	
0.3	,032	,039	Centroid
0.5	,021	,015	500
0.7	,016	,008	
0.9	,015	,005	

Figure A.3: Micro-Averaged Tuning Results for the Newsgroup Data Set

Loss Threshold	**Disjoint k-Split** Avg(error)	**BaseMethod** Avg(error)	**Method #TrainDocs**
0.1	,176	,15	
0.3	,137	,114	SVM
0.5	,095	,076	250
0.7	,069	,046	
0.9	,028	,042	
0.1	,176	,154	
0.3	,149	,108	SVM
0.5	,109	,075	500
0.7	,086	,042	
0.9	,047	,025	
0.1	,136	,121	
0.3	,122	,088	Centroid
0.5	,08	,073	250
0.7	,074	,053	
0.9	,038	,042	
0.1	,153	,128	
0.3	,128	,092	Centroid
0.5	,097	,065	500
0.7	,07	,039	
0.9	,045	,033	

Figure A.4: Micro-Averaged Tuning Results for the imdb Data Set

Appendix B

Results for different Class Pairs

#Partitions	Disjoint k-Split Loss	estLossCV	estLossKL	Error	estErrorCV	estErrorKL	Resampling A Loss	Resampling A Error	Resampling B Loss	Resampling B Error	Method #TrainDocs
2	0,1	0,077	0,132	0,056	0,036	0,018	0,056	0,064	0,047	0,069	
4	0,224	0,228	0,271	0,031	0,011	0,003	0,186	0,045	0,157	0,049	SVM
8	0,502	0,575	0,616	0,022	0,001	0	0,381	0,031	0,355	0,029	250
16	0,785	0,937	0,974	0	0	0	0,689	0,013	0,614	0,029	
2	0,082	0,04	0,104	0,036	0,029	0,031	0,067	0,057	0,038	0,06	
4	0,179	0,109	0,207	0,022	0,007	0,011	0,14	0,018	0,114	0,026	SVM
8	0,353	0,364	0,473	0,014	0	0,001	0,264	0,015	0,256	0,013	500
16	0,612	0,819	0,903	0,006	0	0	0,474	0,004	0,466	0,002	
2	0,078	0,069	0,202	0,067	0,043	0,024	0,093	0,072	0,034	0,088	
4	0,211	0,253	0,36	0,035	0,008	0,005	0,171	0,041	0,131	0,053	Centroid
8	0,466	0,607	0,678	0,013	0	0	0,472	0,019	0,331	0,027	250
16	0,84	0,955	0,97	0	0	0	0,806	0,005	0,687	0,003	
2	0,049	0,057	0,128	0,065	0,046	0,038	0,056	0,071	0,028	0,072	
4	0,146	0,187	0,234	0,036	0,025	0,014	0,113	0,056	0,093	0,056	Centroid
8	0,35	0,411	0,456	0,009	0,001	0,001	0,284	0,02	0,202	0,031	500
16	0,662	0,867	0,828	0,003	0	0	0,479	0,006	0,469	0,006	

Figure B.1: Different Restrictive Splitting and Resampling Methods for ”comp.sys.mac.hardware vs. comp.graphics”

#Partitions	Disjoint k-Split						Resampling A		Resampling B		Method
	Loss	estLossCV	estLossKL	Error	estErrorCV	estErrorKL	Loss	Error	Loss	Error	#TrainDocs
2	0,089	0,092	0,187	0,037	0,024	0,047	0,074	0,043	0,047	0,052	
4	0,227	0,24	0,348	0,018	0,01	0,005	0,2	0,018	0,165	0,034	SVM
8	0,46	0,581	0,686	0,007	0,001	0	0,404	0,01	0,36	0,017	250
16	0,76	0,921	0,989	0	0	0	0,625	0	0,661	0,003	
2	0,054	0,052	0,094	0,032	0,016	0,021	0,064	0,036	0,03	0,045	
4	0,161	0,18	0,196	0,024	0,002	0,005	0,157	0,014	0,126	0,015	SVM
8	0,324	0,435	0,467	0,007	0,001	0	0,298	0,009	0,253	0,013	500
16	0,586	0,844	0,924	0	0	0	0,519	0,005	0,453	0,002	
2	0,081	0,093	0,212	0,079	0,048	0,03	0,097	0,062	0,045	0,086	
4	0,251	0,584	0,371	0,055	0,024	0,007	0,217	0,051	0,133	0,07	Centroid
8	0,466	0,587	0,673	0,021	0,001	0	0,404	0,012	0,468	0,013	250
16	0,823	0,956	0,972	0	0	0	0,708	0	0,731	0	
2	0,051	0,076	0,135	0,057	0,047	0,028	0,066	0,056	0,033	0,06	
4	0,144	0,211	0,248	0,036	0,015	0,008	0,197	0,037	0,138	0,036	Centroid
8	0,327	0,484	0,491	0,01	0	0	0,249	0,024	0,268	0,02	500
16	0,598	0,872	0,896	0	0	0	0,56	0,003	0,549	0,007	

Figure B.2: Different Restrictive Splitting and Resampling Methods for ”rec.autos vs. rec.motorcycles”

#Partitions	Disjoint k-Split						Resampling A		Resampling B		Method
	Loss	estLossCV	estLossKL	Error	estErrorCV	estErrorKL	Loss	Error	Loss	Error	#TrainDocs
2	0,104	0,088	0,184	0,046	0,05	0,012	0,067	0,056	0,035	0,053	
4	0,285	0,389	0,361	0,027	0,008	0,001	0,198	0,04	0,135	0,039	SVM
8	0,6	0,774	0,738	0,008	0	0	0,403	0,025	0,444	0,025	250
16	0,882	0,989	0,996	0	0	0	0,782	0	0,768	0,004	
2	0,084	0,055	0,119	0,035	0,037	0,026	0,063	0,05	0,039	0,039	
4	0,177	0,23	0,237	0,024	0,006	0,007	0,168	0,032	0,118	0,034	SVM
8	0,379	0,546	0,511	0,023	0	0	0,28	0,025	0,284	0,022	500
16	0,709	0,926	0,926	0,006	0	0	0,566	0,021	0,502	0,014	
2	0,176	0,231	0,32	0,087	0,097	0,036	0,107	0,127	0,031	0,109	
4	0,467	0,589	0,519	0,06	0,022	0,008	0,306	0,056	0,282	0,054	Centroid
8	0,733	0,888	0,808	0,015	0,001	0,001	0,516	0,052	0,583	0,034	250
16	0,927	0,994	0,992	0	0	0	0,835	0,006	0,833	0,018	
2	0,082	0,12	0,274	0,132	0,136	0,056	0,057	0,144	0,046	0,142	
4	0,271	0,444	0,456	0,091	0,047	0,018	0,22	0,078	0,139	0,108	Centroid
8	0,486	0,698	0,729	0,049	0,004	0,002	0,455	0,056	0,271	0,098	500
16	0,789	0,961	0,973	0,017	0	0	0,682	0,0281	0,534	0,038	

Figure B.3: Different Restrictive Splitting and Resampling Methods for ”Western vs. Comedy”

Loss Threshold (k)	Disjoint k-Split				BaseMethod			Method #TrainDocs
	Loss	estLossCV	Error	estErrorCV	Loss	Error	estError	
0,1 (2)	0,1	0,077	0,056	0,036	0,1	0,051	0,024	
0,3 (4)	0,224	0,228	0,031	0,011	0,3	0,023	0,006	SVM
0,5 (7)	0,387	0,451	0,016	0,001	0,5	0,014	0,004	250
0,7 (9)	0,53	0,63	0,017	0	0,7	0,007	0	
0,9 (15)	0,741	0,89	0,008	0	0,9	0,01	0	
0,1 (2)	0,082	0,04	0,036	0,029	0,1	0,036	0,017	
0,3 (5)	0,212	0,24	0,024	0,005	0,3	0,016	0,004	SVM
0,5 (8)	0,353	0,428	0,014	0	0,5	0,007	0,004	500
0,7 (12)	0,468	0,645	0,004	0	0,7	0	0,007	
0,9 (16)	0,612	0,819	0,006	30	0,9	0	0	
0,1 (2)	0,078	0,069	0,067	0,043	0,1	0,059	0,074	
0,3 (4)	0,211	0,253	0,035	0,008	0,3	0,02	0,074	Centroid
0,5 (6)	0,403	0,456	0,017	0,001	0,5	0,006	0,074	250
0,7 (9)	0,546	0,67	0,009	0	0,7	0,003	0,007	
0,9 (12)	0,797	0,861	0	0	0,9	0	0	
0,1 (2)	0,049	0,057	0,065	0,046	0,1	0,044	0,0198	
0,3 (5)	0,201	0,23	0,031	0,021	0,3	0,011	0,006	Centroid
0,5 (8)	0,35	0,48	0,009	0,001	0,5	0,002	0,004	500
0,7 (12)	0,472	0,67	0,004	0	0,7	0	0	
0,9 (16)	0,662	0,867	0,003	0	0,9	0	0	

Figure B.4: Tuning Results for ”comp.sys.mac.hardware vs. comp.graphics”

Loss Threshold (k)	Disjoint k-Split				BaseMethod			Method #TrainDocs
	Loss	estLossCV	Error	estErrorCV	Loss	Error	estError	
0,1 (2)	0,089	0,092	0,037	0,024	0,1	0,031	0,016	
0,3 (4)	0,227	0,24	0,018	0,01	0,3	0,011	0,003	SVM
0,5 (6)	0,403	0,426	0,012	0,003	0,5	0,004	0	250
0,7 (9)	0,572	0,671	0,002	0	0,7	0	0	
0,9 (13)	0,653	0,842	0	0	0,9	0	0	
0,1 (2)	0,054	0,052	0,032	0,016	0,1	0,019	0,007	
0,3 (5)	0,234	0,268	0,01	0,002	0,3	0,006	0,001	SVM
0,5 (8)	0,324	0,435	0,007	0	0,5	0,002	0	500
0,7 (12)	0,466	0,68	0,008	0	0,7	0,004	0	
0,9 (16)	0,586	0,844	0	0	0,9	0	0	
0,1 (2)	0,081	0,093	0,079	0,048	0,1	0,063	0,09	
0,3 (4)	0,251	0,274	0,055	0,024	0,3	0,034	0,09	Centroid
0,5 (6)	0,447	0,464	0,013	0,006	0,5	0,012	0,09	250
0,7 (8)	0,466	0,587	0,021	0,001	0,7	0	0,017	
0,9 (12)	0,719	0,856	0	0	0,9	0	0,01	
0,1 (2)	0,051	0,076	0,057	0,047	0,1	0,037	0,022	
0,3 (5)	0,206	0,28	0,024	0,008	0,3	0,021	0,006	Centroid
0,5 (8)	0,327	0,484	0,01	0	0,5	0,009	0,004	500
0,7 (12)	0,486	0,691	0,002	0	0,7	0	0,003	
0,9 (16)	0,598	0,872	0	0	0,9	0	0	

Figure B.5: Tuning Results for ”rec.autos vs. rec.motorcycles”

Loss Threshold (k)	Disjoint k-Split				BaseMethod			Method #TrainDocs
	Loss	estLossCV	Error	estErrorCV	Loss	Error	estError	
0,1 (2)	0,104	0,088	0,046	0,05	0,1	0,037	0,038	
0,3 (3)	0,175	0,232	0,035	0,021	0,3	0,023	0,014	SVM
0,5 (5)	0,376	0,458	0,026	0,014	0,5	0,006	0,004	250
0,7 (7)	0,566	0,631	0,016	0,002	0,7	0,007	0	
0,9 (10)	0,613	0,896	0,012	0	0,9	0	0	
0,1 (2)	0,084	0,055	0,035	0,037	0,1	0,026	0,028	
0,3 (4)	0,177	0,23	0,024	0,006	0,3	0,028	0,01	SVM
0,5 (7)	0,334	0,444	0,027	0,001	0,5	0,011	0,004	500
0,7 (10)	0,493	0,678	0,014	0	0,7	0,006	0,003	
0,9 (15)	0,661	0,87	0,011	0	0,9	0	0	
0,1 (1)	0	0	0,127	–	0,1	0,092	0,146	
0,3 (2)	0,176	0,231	0,087	0,097	0,3	0,059	0,146	Centroid
0,5 (3)	0,255	0,372	0,075	0,037	0,5	0,036	0,146	250
0,7 (5)	0,486	0,532	0,043	0,05	0,7	0,017	0,037	
0,9 (8)	0,733	0,888	0,015	0,001	0,9	0,02	0,03	
0,1(1)	0	0	0,168	–	0,1	0,129	0,04	
0,3(3)	0,152	0,229	0,112	0,066	0,3	0,077	0,031	Centroid
0,5(5)	0,282	0,366	0,08	0,051	0,5	0,064	0,029	500
0,7(7)	0,439	0,623	0,068	0,01	0,7	0,036	0,018	
0,9(12)	0,616	0,871	0,033	0	0,9	0,018	0	

Figure B.6: Tuning Results for ”Western vs. Comedy”