Improving Multiclass Text Classification with the Support Vector Machine*

Jason D. M. Rennie
jrennie@ai.mit.edu

Ryan Rifkin
rif@mit.edu

Revised, April 2002

Abstract

We compare Naive Bayes and Support Vector Machines on the task of multiclass text classification. Using a variety of approaches to combine the underlying binary classifiers, we find that SVMs substantially outperform Naive Bayes. We present full multiclass results on two well-known text data sets, including the lowest error to date on both data sets. We develop a new indicator of binary performance to show that the SVM's lower multiclass error is a result of its improved binary performance. Furthermore, we demonstrate and explore the surprising result that one-vs-all classification performs favorably compared to other approaches even though it has no error-correcting properties.

Acknowledgements

This report describes research done within the Center for Biological & Computational Learning in the Department of Brain & Cognitive Sciences and in the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology.

This research was sponsored by grants from: Office of Naval Research (DARPA) under contract No. N00014-00-1-0907, National Science Foundation (ITR) under contract No. IIS-0085836, National Science Foundation (KDI) under contract No. DMS-9872936, and National Science Foundation under contract No. IIS-9800032

Additional support was provided by: Central Research Institute of Electric Power Industry, Center for e-Business (MIT), Eastman Kodak Company, DaimlerChrysler AG, Compaq, Honda R&D Co., Ltd., Komatsu Ltd., Merrill-Lynch, NEC Fund, Nippon Telegraph & Telephone, Siemens Corporate Research, Inc., Toyota Motor Corporation and The Whitaker Foundation.

*Published October 2001 by the Massachusetts Institute of Technology as AI Memo 2001-026 and CBCL Memo 210.

1 Introduction and Related Work

Multiclass text classification involves the assigning of one of $m > 2$ labels to test documents. There are many ways to approach this problem. One, called Error-Correcting Output Coding (ECOC), is to learn a number of different binary classifiers and use the outputs of those classifiers to determine the label for a new example. An advantage of this method is that the problem of binary classification is well studied. ECOC merely bridges the gap between binary classification and multiclass classification. The Support Vector Machine (SVM) has proven to be an effective binary text classifier. Through experiments on two data sets, we show that the SVM can also be an effective multiclass text classifier when used with ECOC.

In 1997 Joachims published results on a set of binary text classification experiments using the Support Vector Machine. The SVM yielded lower error than many other classification techniques. Yang and Liu followed two years later with experiments of their own on the same data set [1999]. They used improved versions of Naive Bayes (NB) and k-nearest neighbors (kNN) but still found that the SVM performed at least as well as all other classifiers they tried. Both papers used the SVM for binary text classification, leaving the multiclass problem (assigning a single label to each example) open for future research.

Berger and Ghani individually chose to attack the multiclass problem using Error-correcting Output Codes (ECOC) with NB as the binary classifier[1999] [2000]. Ghani found the greatest reduction in error on Industry Sector, a data set with 105 classes. In parallel, Allwein *et al.* wrote an article on using ECOC with loss functions for multiclass classification in non-text domains [2000]. They presented a "unifying framework" for multiclass classification, encouraging the use of loss functions, especially when the classifier is optimized for a particular one.

We bridge these bodies of work by applying ECOC to multiclass text classification with the SVM as the binary learner. We achieve the lowest-known error rate on both the Industry Sector and 20 Newsgroups data sets. Surprisingly, we find that one-vs-all performs comparably with matrices that have high row- and column-separation. Binary performance plays a key role in these results. We see this through the introduction of a new measure, ROC breakeven. We show that trends in binary performance are directly reflected in multiclass error. In particular, improved binary performance allows for low error with the one-vs-all matrix even though it has no error-correcting properties. Allwein *et al.* give theoretical arguments that the loss function used should match that used to optimize the binary classifier [2000]. We found that the most important aspect of the loss function is to contribute confidence information, and that in practice, other loss functions perform equally well.

2 Error-Correcting Output Coding

Error-correcting output coding (ECOC) is an approach for solving multiclass categorization problems originally introduced by Dietterich and Bakiri [1991]. It reduces the multiclass problem to a group of binary classification tasks and combines the binary classification results to predict multiclass labels.

R is the *code matrix*. It defines the data splits which the binary classifier is to learn. $R_{i\cdot}$ is the i^{th} row of the matrix and defines the *code* for class i. $R_{\cdot j}$ is the j^{th} column of the matrix and defines a split for the classifier to learn. $R \in \{-1,+1\}^m \times \{-1,+1\}^l$ where m is the number of classes and l is the number of partitionings (or length of each code). In a particular column, $R_{\cdot j}$, -1 and $+1$ represent the assignment of the classes to one of two partitions. We use three different matrices, 1) the one-vs-all (OVA) matrix, where the diagonal is filled with $+1$'s and is otherwise filled with -1 entries, 2) the Dense matrix [Allwein *et al.*, 2000], where entries are independently determined by flipping a fair coin, assigning $+1$ for heads and -1 for tails, and 3) BCH codes, a matrix construction technique that yields high column- and row-separation [Ghani, 2000]. We use the 63-20 and 63-105 BCH codes[1] that Ghani has made available on-line at http://www.cs.cmu.edu/∼rayid/ecoc.

Let $(f_1, \ldots, f_l)$ be the classifiers trained on the partitionings indicated in the code matrix. Furthermore, let $g : \Re \to \Re$ be the chosen loss function. Then, the multiclass classification of a new example, x is

$$\hat{H}(x) = \text{argmin}_{c \in \{1,\ldots,m\}} \sum_{i=1}^{l} g(f_i(x) R_{ci}). \tag{1}$$

Allwein *et al.* give a full description of the code matrix classification framework and give loss functions for various models [2000]. As suggested by Allwein *et al.*, we use "hinge" loss, $g(z) = (1-z)_+$, for the SVM. Naive Bayes does not optimize a loss function, but we use the "hinge" loss since it gives the lowest error in practice.

3 Classification Algorithms

Here we briefly describe the two classification algorithms used for experiments in this paper.

3.1 Naive Bayes

Naive Bayes is a simple Bayesian text classification algorithm. It assumes that each term in a document is drawn independently from a multinomial distribution and classifies according to the Bayes optimal decision rule. Each class has its own set of multinomial parameters, θ^c. We estimate parameters via Maximum

[1] We use sub-matrices of the 63-column code for 15- and 31-column matrices.

a Posteriori (MAP) using the training data. A new document, d, is assigned the label

$$\hat{H}(d) = \text{argmax}_c\, p(d|\hat{\theta}^c)p(c). \tag{2}$$

Using D^c to denote the training data for class c, we use parameter estimates

$$\hat{\theta}^c = \text{argmax}_\theta\, p(D^c|\theta)p(\theta). \tag{3}$$

A Dirichlet parameter prior, $p(\theta)$, with hyper-parameters $\alpha_i = 2\ \forall i$ gives an estimate of $\hat{\theta}^c_k = \frac{N^c_k+1}{N^c+V}$ for word k, where N^c_k is the number of times word k occurred in class c, N^c is the total number of word occurrences in class c and V is the size of the vocabulary. Using a flat class prior, $p(c) = \frac{1}{m}$, our decision rule is

$$\hat{H}(d) = \text{argmax}_c \prod_k \left(\frac{N^c_k + 1}{N^c + V}\right)^{f_k} \tag{4}$$

See Rennie for more [2001]. Chakrabarti *et al.* use a flat parameter prior and expectation to derive the same parameter estimates [Chakrabarti *et al.*, 1997]. McCallum and Nigam give an explanation of the distinction between the traditional Naive Bayes classifier and the one commonly used for text classification [McCallum and Nigam, 1998].

3.2 The Support Vector Machine

The Support Vector Machine is a classifier, originally proposed by Vapnik, that finds a maximal margin separating hyperplane between two classes of data [1995]. An SVM is trained via the following optimization problem:

$$\text{argmin}_w \frac{1}{2}\|w\|^2 + C\sum_i \xi_i, \tag{5}$$

with constraints

$$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) \geq 1 - \xi_i\ \forall i. \tag{6}$$

For more information, see Burges' tutorial and Cristianini and Shawe-Taylor's book [1998] [2000]. There are non-linear extensions to the SVM, but Yang and Liu found the linear kernel to outperform non-linear kernels in text classification [1999]. In informal experiments, we also found that linear performs at least as well as non-linear kernels. Hence, we only present linear SVM results. We use the SMART ltc[2] transform; the SvmFu package is used for running experiments [Rifkin, 2000].

[2] The list of SMART weightings can be found at http://www.ai.mit.edu/people/jrennie/ecoc-svm/smart.html.

3.3 Relative efficiency of NB and the SVM

Naive Bayes and the linear SVM are both highly efficient and are suitable for large-scale text systems. As they are linear classifiers, both require a simple dot product to classify a document. The implication by Dumais *et al.* that the linear SVM is faster to train than NB is incorrect [1998]. Training NB is faster since no optimization is required; only a single pass over the training set is needed to gather word counts. The SVM must read in the training set and then perform a quadratic optimization. This can be done quickly when the number of training examples is small (e.g. < 10000 documents), but can be a bottleneck on larger training sets. We realize speed improvements with chunking and by caching kernel values between the training of binary classifiers.

4 Data Sets

For our experiments, we use two well-known data sets, 20 Newsgroups and Industry Sector [McCallum and Nigam, 1998; Slonim and Tishby, 1999; Berger, 1999; Ghani, 2000]. We use rainbow to pre-process the text documents into feature-vector form [McCallum, 1996]. Documents that are empty after pre-processing (23 for 20 Newsgroups, 16 for Industry Sector) are not used. The code and pre-processed data we used for our experiments can be found at http://www.ai.mit.edu/people/jrennie/ecoc-svm/.

20 Newsgroups is a data set collected and originally used for text classification by Lang [1995].[3] It contains 19,997 documents evenly distributed across 20 classes. We remove all headers and UU-encoded blocks, and skip stoplist words and words that occur only once.[4] The vocabulary size is 62,061. We randomly select 80% of documents per class for training and the remaining 20% for testing. This is the same pre-processing and splitting as McCallum and Nigam used in their 20 Newsgroups experiments [McCallum and Nigam, 1998].

The Industry Sector data is a collection of corporate web pages organized into categories based on what a company produces or does.[5] There are 9649 documents and 105 categories. The largest category has 102 documents, the smallest has 27. We remove headers, and skip stoplist words and words that occur only once.[6] Our vocabulary size is 55197. We randomly select 50% of documents per class for training and the remaining 50% for testing.

We conduct our experiments on 10 test/train splits and report average performance. To gauge performance for different amounts of training documents, we create nested training sets of 800, 250, 100 and 30 documents/class for 20 Newsgroups and (up to) 52, 20, 10 and 3 documents/class for Industry Sector.

[3] The 20 Newsgroups data set can be obtained from http://www.ai.mit.edu/people/jrennie/20Newsgroups/.

[4] Our 20 Newsgroups pre-processing corresponds to rainbow options "–istext-avoid-uuencode –skip-header -O 2."

[5] We obtained the Industry Sector data set from http://www-2.cs.cmu.edu/∼TextLearning/datasets.html.

[6] Our Industry Sector pre-processing corresponds to rainbow options "–skip-header -O 2."

When a class does not have the specified number of documents, we use all of the training documents for that class.

Text classification experiments often include a feature selection step which may improve classification. McCallum and Nigam performed feature selection experiments on the 20 Newsgroups data set and an alternate version of the Industry Sector data set [1998]. Feature selection did not improve accuracy for 20 Newsgroups and improved accuracy only slightly on Industry Sector. In our own experiments, we found feature selection to only increase error. We use the entire vocabulary in our experiments.

5 Results

The SVM consistently outperforms Naive Bayes, but the difference in performance varies by matrix and amount of training data used. On the Industry Sector data set, the disparity for the OVA matrix between the SVM and Naive Bayes is especially marked. Our analysis (§ 5.4) shows that this is a direct result of differences in binary performance. A new measure of binary classification, ROC breakeven (§ 5.3), allows us to see this close connection. Other factors, such as the loss function and the independence of binary classifiers show less influence. Our results include the best-known multiclass error on both data sets.

5.1 Multiclass classification

The SVM always outperforms Naive Bayes on the multiclass classification task. Table 1 shows multiclass error for a variety of conditions. The SVM achieves 10-20% lower error than Naive Bayes for most of the 20 Newsgroups experiments. The differences are larger for Industry Sector. When all training documents are used, the SVM achieves 80% and 48% less error for the OVA and 63-column BCH matrices, respectively. On both data sets, the lowest error is found using the SVM with a 63-column BCH matrix. BCH consistently outperforms Dense; the under-performance of the 15-column BCH matrix is a result of using a sub-matrix of the 63-column BCH.[7]

With the SVM, the OVA matrix performs well even though it has no error-correcting properties and experiments on non-text data sets have shown it to perform poorly [Allwein *et al.*, 2000]. OVA performs about as well as BCH even though its row- and column-separation is small. OVA has a row- and column-separation of 2; The 63-column BCH matrix has a row-separation of 31 and a column-separation of 49.[8] Some have suggested that OVA is inferior to other methods [Ghani, 2000; Guruswami and Sahal, 1999]. These results show

[7] The BCH matrix is optimized to maximize row- and column-separation. A sub-matrix of a BCH matrix will not necessarily exhibit the same row- and column-separation properties as one that is optimized for that size. However, except where noted, we find little degradation in using sub-matrices.

[8] We define row-separation to be the average over all rows of the smallest Hamming distance between the row and all other rows: $\frac{1}{n}\sum_{i=1}^{n}\min_{j\neq i}\mathrm{Hamming}(r_i, r_j)$. Column-separation is defined analogously.

20 Newsgroups	800		250		100		30	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA	0.131	0.146	0.167	0.199	0.214	0.277	0.311	0.445
Dense 15	0.142	0.176	0.193	0.222	0.251	0.282	0.366	0.431
BCH 15	0.145	0.169	0.196	0.225	0.262	0.311	0.415	0.520
Dense 31	0.135	0.168	0.180	0.214	0.233	0.276	0.348	0.428
BCH 31	0.131	0.153	0.173	0.198	0.224	0.259	0.333	0.438
Dense 63	0.129	0.154	0.171	0.198	0.222	0.256	0.326	0.407
BCH 63	0.125	0.145	0.164	0.188	0.213	0.245	0.312	0.390

Industry Sector	52		20		10		3	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA	0.072	0.357	0.176	0.568	0.341	0.725	0.650	0.885
Dense 15	0.119	0.191	0.283	0.363	0.461	0.542	0.738	0.805
BCH 15	0.106	0.182	0.261	0.352	0.438	0.518	0.717	0.771
Dense 31	0.083	0.145	0.216	0.301	0.394	0.482	0.701	0.769
BCH 31	0.076	0.140	0.198	0.292	0.371	0.462	0.676	0.743
Dense 63	0.072	0.135	0.189	0.279	0.363	0.453	0.674	0.745
BCH 63	0.067	0.128	0.176	0.272	0.343	0.443	0.653	0.734

Table 1: Above are results of multiclass classification experiments on the 20 Newsgroups (top) and Industry Sector (bottom) data sets. The top row of each table indicates the number of documents/class used for training. The second row indicates the binary classifier. The far left column indicates the multiclass technique. Entries in the table are classification error.

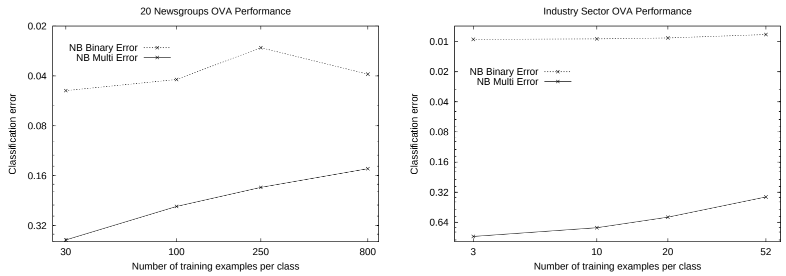

Figure 1: Multiclass error improves as the number of training examples increases, but binary error improves marginally for Industry Sector and degrades for 20 Newsgroups. Shown is the performance of OVA with Naive Bayes as the binary classifier. Guessing achieves a binary error of 0.05 for 20 Newsgroups and approximately 0.01 for Industry Sector. Binary error is loosely tied to binary classifier strength. Note the log scale on both axes.

that OVA should not be ignored as a multiclass classification technique. In subsequent sections, we show that the success of OVA is due to strong binary SVM classifiers and the use of confidence information. Naive Bayes performs poorly on Industry Sector with the OVA matrix because of its lackluster binary performance.

Error differences between the SVM and Naive Bayes are less on 20 Newsgroups than Industry Sector. We believe this to be partially caused by the fact that nearly 6% (we counted 1169) of the non-empty 20 Newsgroups documents are exact duplicates, many cross-posts to multiple newsgroups. This limits classification accuracy and makes training more difficult. We ignored documents with duplicate feature vectors when training the SVM since handling duplicate vectors in SVM training is inconvenient.

5.2 Comparison with other work

Our Naive Bayes results match those of neither Ghani nor Berger [2000] [1999]. We did not try to reproduce the results of Berger because his preprocessing code was hand-written and somewhat complex. We made an effort to match those of Ghani, but were not successful. When we used the same pre-processing (stripping HTML markup), we saw higher error, 0.187 for the 63-column BCH matrix, compared to the 0.119 he reports. We worked with Ghani to try to resolve this discrepancy, but were unsuccessful. We found that including HTML gives lower error than skipping HTML.

5.3 ROC Breakeven

The performance of an ECOC classifier is affected by a number of factors: (1) binary classifier performance, (2) independence of the binary classifiers, and

		Guess	
		+1	−1
True	+1	tp	fn
Label	−1	fp	tn

Table 2: The performance of a binary classifier can be described with a 2x2 confusion matrix, as shown. Two letters describe each entry. "t" stands for true. "f" is false. "p" is positive. "n" is positive. The false alarm rate is fp/(tn+fp). The miss rate is fn/(tp+fn). ROC breakeven is the average of the false alarm and miss rates when the difference between them is minimized.

(3) the loss function. Of these, we find binary performance to be the most influential in observed differences between SVM and Naive Bayes performance. We use error to measure multiclass performance. However, we avoid binary error as a measure of binary performance. Figure 1 shows why. Additional training examples improves multiclass error for Naive Bayes, but binary error is sporadic, showing little change for Industry Sector and rising, then falling for 20 Newsgroups. The OVA matrix partitions examples very unevenly, assigning most examples to a single class. Hence, error mainly judges the classifiers performance on examples of a single class.

A better measure of binary performance is one that evenly measures the two classes. We propose ROC breakeven as such a measure. Table 2 shows terms used to describe the output of a classifier. We define the ROC breakeven as the average of the miss and false alarm rates at the point where the difference between false alarm rate and the miss rate is minimum. This corresponds to the convex combination of two points on the ROC curve and is hence always achievable, unlike precision-recall breakeven [Joachims, 1997]. We achieve different rates by modifying the bias term of the classifier. ROC breakeven selects the bias such that the classifier performs as well on examples of class $+1$ as examples of class -1. This allows us to better judge the strength of a binary classifier when the example distribution is uneven. When the example distribution is even (such as with BCH and Dense matrices), ROC breakeven is nearly identical to binary error.

5.4 Multiclass error is largely a function of binary performance

Figure 2 compares multiclass error and ROC breakeven using the 63-column BCH matrix. In the 20 Newsgroups plot, we see the consistent gap between ROC breakeven scores reflected in a similar gap in multiclass errors. The Industry Sector results follow a different trend: ROC breakeven between the SVM and Naive Bayes diverges as the number of training examples increases. We see this directly affecting multiclass error: the difference between multiclass errors widen as more examples are used.

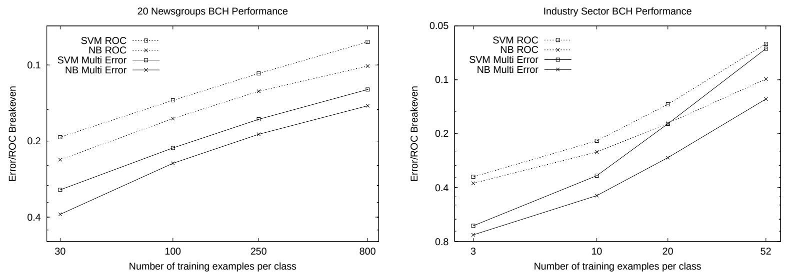

Figure 2: Shown is a comparison between ROC breakeven and multiclass error of ECOC using a BCH-63 matrix and the SVM and Naive Bayes as the binary classifier. We see that ROC breakeven largely dictates multiclass error. Trends in the ROC breakeven curves are reflected in the multiclass error curves. Note the log scale on both axes.

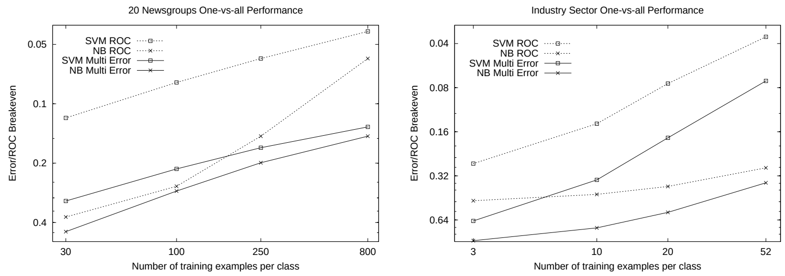

Figure 3: Shown is ROC breakeven and multiclass error for ECOC with the OVA matrix. Changes in ROC breakeven are directly reflected in multiclass error. Multiclass error changes gradually for 20 Newsgroups, but trends in ROC breakeven are evident in the multiclass error. Note the log scale on both axes.

20 Newsgroups	800		250		100		30	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA/Error	0.015	0.039	0.021	0.027	0.03	0.042	0.044	0.049
OVA/ROC	0.043	0.059	0.059	0.146	0.078	0.262	0.118	0.375
BCH/Error	0.079	0.101	0.105	0.121	0.135	0.151	0.194	0.224
BCH/ROC	0.081	0.101	0.108	0.127	0.138	0.163	0.193	0.237

Industry Sector	52		20		10		3	
	SVM	NB	SVM	NB	SVM	NB	SVM	NB
OVA/Error	0.003	0.008	0.005	0.009	0.007	0.009	0.009	0.010
OVA/ROC	0.036	0.282	0.075	0.378	0.141	0.428	0.264	0.473
BCH/Error	0.062	0.100	0.137	0.176	0.218	0.253	0.347	0.376
BCH/ROC	0.063	0.099	0.137	0.175	0.219	0.253	0.348	0.378

Table 3: Shown are binary errors and ROC breakeven points for the binary classifiers trained according to the matrix columns. Results for the Dense matrix are omitted since they are nearly identical to the BCH results. Table entries are averaged over all matrix columns and 10 train/test splits. Error is a poor judge of classifier strength for the OVA matrix. Error increases with more examples on 20 Newsgroups. Note that error and ROC breakeven numbers are very similar for the BCH matrix.

A similar relation between ROC breakeven and multiclass error can be seen on the OVA results, as is depicted in figure 3. Large differences in ROC breakeven translate into small differences in multiclass error on 20 Newsgroups, but the connection is still visible. The SVM shows the greatest improvements over Naive Bayes on the Industry Sector data set. Using all of the training examples, the SVM achieves an ROC breakeven of 0.036, compared to Naive Bayes's lackluster 0.282. This translates into the large difference in multiclass error. The SVM's binary performance is a result of its ability to handle uneven example distributions. Naive Bayes cannot make effective use of such a training example distribution [Rennie, 2001].

The full binary error and ROC breakeven results can be found in table 3. As we have seen, ROC breakeven helps greatly to explain differences in multiclass error. Trends in ROC breakeven are clearly reflected in multiclass error. Independence of the binary classifiers also clearly plays a role—identical Naive Bayes and SVM ROC breakevens do not yield identical multiclass errors. But, such effects are secondary when comparing the SVM and Naive Bayes. Next, we show that the loss function plays essentially no role in ECOC text classification.

5.5 The Loss Function

Allwein *et al.* indicate that it is important for the loss function used for ECOC to be the same as the one used to optimize the classifier [2000]. We find this not to be the case. Rather, we see the most important aspect of the loss function being

20 Newsgroups	Hinge	Linear
OVA/SVM	0.131	0.131
OVA/NB	0.146	0.146
BCH 63/SVM	0.125	0.126
BCH 63/NB	0.145	0.144

Industry Sector	Hinge	Linear
OVA/SVM	0.072	0.072
OVA/NB	0.357	0.357
BCH 63/SVM	0.067	0.067
BCH 63/NB	0.128	0.127

Table 4: Shown are multiclass errors on two data sets and a variety of ECOC classifiers. Errors are nearly identical between the hinge and linear loss functions. Although ECOC provides opportunity for non-linear decision rules through the loss function, the use of a non-linear loss function provides no practical benefit.

that it convey confidence information of the binary classifier. The simple linear loss function, $g(x) = -x$, performs as well as the hinge loss in our experiments. See table 4 for a full comparison. Both the linear and hinge losses perform better than other loss functions we tried. The Hamming loss, which conveys no confidence information, performed worst. The principle job of the loss function is to convey binary classification confidence information. In the case of the SVM, we find it unnecessary that the loss function match the loss used to optimize the binary classifier.

6 Conclusion

We have shown that the Support Vector Machine can perform multiclass text classification very effectively when used as part of an ECOC scheme. Its improved ability to perform binary classification gives it much lower error scores than Naive Bayes. In particular, one-vs-all, when used with confidence scores, holds promise as an alternative to more complicated matrix-construction techniques.

Another important area of future work is in matrix design. We have shown that high row- and column-separation are not the only aspects of a code matrix that lead to low overall error. A binary classifier that can learn one-vs-all splits better than BCH splits may yield equal or lower error overall. Different data sets will have different natural partitionings. Discovering those before training may lead to reduced multiclass error. Crammer and Singer have proposed an algorithm for relaxing code matrices, but we believe that more work is needed [2001]. It may also be beneficial to weight individual examples, as is done in multiclass boosting algorithms [Guruswami and Sahal, 1999; Schapire, 1997].

References

[Allwein *et al.*, 2000] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[Berger, 1999] Adam Berger. Error-correcting output coding for text classification. In *Proceedings of IJCAI-99 Workshop on Machine Learning for Information Filtering*, Stockholm, Sweeden, 1999.

[Burges, 1998] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Chakrabarti *et al.*, 1997] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Using taxonomy, discriminants and signatures for navigating in text databases. In *Proceedings of the 23rd VLDB Conference*, 1997.

[Crammer and Singer, 2001] Koby Crammer and Yoram Singer. Improved output coding for classification using continuous relaxation. In *Advances in Neural Information Processing Systes 13 (NIPS*00)*, 2001.

[Cristianini and Shawe-Taylor, 2000] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, 2000.

[Dietterich and Bakiri, 1991] Tom G. Dietterich and Ghulum Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 572–577, Anaheim, CA, 1991. AAAI Press.

[Dumais *et al.*, 1998] Susand Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and reepresentations for text classification. In *Seventh International Conference on Information and Knowledge Management*, 1998.

[Ghani, 2000] Rayid Ghani. Using error-correcting codes for text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[Guruswami and Sahal, 1999] Venkatesan Guruswami and Amit Sahal. Multiclass learning, boosting and error-correcting codes. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

[Joachims, 1997] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical report, University of Dortmund, Computer Science Department, 1997.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[McCallum and Nigam, 1998] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI-98 workshop on Learning for Text Categorization*, 1998.

[McCallum, 1996] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

[Rennie, 2001] Jason D. M. Rennie. Improving multi-class text classification with naive bayes. Master's thesis, Massachusetts Institute of Technology, 2001.

[Rifkin, 2000] Ryan Rifkin. Svmfu. http://five-percent-nation.mit.edu/SvmFu/, 2000.

[Schapire, 1997] Robert E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth International Conference*, 1997.

[Slonim and Tishby, 1999] Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In *Neural Information Processing Systems 12 (NIPS-99)*, 1999.

[Vapnik, 1995] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[Yang and Liu, 1999] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.