Using Error-Correcting Codes For Text Classification

Rayid Ghani RAYID@CS.CMU.EDU
Center for Automated Learning & Discovery, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA

Abstract

This paper explores in detail the use of Error Correcting Output Coding (ECOC) for learning text classifiers. We show that the accuracy of a Naive Bayes Classifier over text classification tasks can be significantly improved by taking advantage of the error-correcting properties of the code. We also explore the use of different kinds of codes, namely Error-Correcting Codes, Random Codes, and Domain and Data-specific codes and give experimental results for each of them. The ECOC method scales well to large data sets with a large number of classes. Experiments on a real-world data set show a reduction in classification error by up to 66% over the traditional Naive Bayes Classifier. We also compare our empirical results to semi-theoretical results and find that the two closely agree.

1. Introduction

Text Classification is the problem of grouping text documents into classes or categories. For the purpose of this paper, we define classification as categorizing documents into one of a fixed number of predefined classes with a single document belonging to only one class. The enormous increase in the amount of available textual data has resulted in various new algorithms for *automatic* text classification. In this paper, we apply the Error-Correcting Output coding (ECOC) method (Diettrich & Bakiri, 1995) to the problem of text classification. Whereas others have shown the utility of this approach in classification tasks such as NETtalk and letter recognition, our work refines the use of ECOC for text classification.

We will take the usual approach for supervised classification learning where a labeled training set is presented to the learning algorithm. The algorithm uses the training set to construct a probabilistic model that is later used to map new (test) documents to a class. Each class is assigned a unique binary string of length n; we will refer to these strings as codewords (Diettrich & Bakiri, 1995). Then we train n classifiers to predict each bit of the string. The predicted class is the one whose codeword is closest to the codeword produced by the classifiers. The distance metric we use in our experiments is the Hamming distance which counts the number of bits that the two codewords differ by. This process of mapping the output string to the nearest codeword is identical to the decoding step for error-correcting codes (Bose & Ray-Chaudhri, 1960; Hocuenghem, 1959).

The error-correcting codes and their application to classification problems can be better understood by an analogy given by Diettrich & Bakiri (1995). We can look at text classification as a type of communications problem where the correct category is being 'transmitted' over a medium or channel. The channel consists of words, the training examples, and the learning algorithm. Due to errors introduced by the finite training sample, poor choice of input features and limitations or invalid assumptions made in the learning process, the class information is distorted. By using an error-correcting code and 'transmitting' each bit separately (via a separate run of the algorithm), the system may be able to recover from the errors. The codes used should then correct as many errors as possible.

We show that using ECOC improves the classification accuracy of the Naive Bayes Classifier (NBC) by up to 66%[1] and give some explanation as to why this improvement occurs. This improvement in accuracy comes with a cost of increased training time that is linear in the length of the code. We also use different types of codes and give experimental results for each of them showing that codes constructed using algebraic coding theory should be the codes of choice for most classification tasks when using the ECOC method.

2. Error-Correcting Codes

Error-Correcting Codes have traditionally been used to correct errors when transmitting data in communication tasks. The idea behind these codes is to add redundancy to the data being transmitted so that even if some errors occur due to the noise in the channel, the data can be correctly received at the other end.

The key difference in the use of Error-Correcting Codes in communication tasks as opposed to their use in Machine Learning (classification) tasks, such as ours, is

[1] This improvement in accuracy was observed on the Industry Sector Dataset described in the Experiments section.

that communication tasks only require the rows of a code to be well-separated (in terms of the hamming distance), whereas classification tasks require the columns to be well-separated as well. The reason behind the rows being well-separated is obvious, since we want codewords or classes to be maximally far apart from each other, but the column separation is necessary because the functions being learned by the learner for each bit should be uncorrelated so that the errors in each bit are independent of each other. If the errors made by the learner in each bit were correlated then an error in one bit would result in errors in multiple bits and the code would not be able to correct those errors. This independence between the errors made in each bit of the codeword forms the basis of the theory behind error-correcting codes.

It is not clear how much column separation is actually necessary and after what point does it become unnecessary. In fact, there may be other ways of making the errors in each bit independent and uncorrelated, instead of column separation, such as using disjoint sets of features to learn each bit of the codeword which are independent of each other but denote the same concept. This would utilize a setting similar to that of the co-training algorithm proposed by Blum & Mitchell (1998). For example, in the case of classifying web pages, the training data could be both the hyperlinks to a page and the page itself and we could use a code that is not well separated in terms of column hamming distance and still have independent errors since the data itself is independent.

- Training Phase

1. Create an m x n binary matrix M.
2. Each class is assigned one row of M.
3. Train the base classifier to learn the n binary functions (one for each column).

- Test Phase

1. Apply each of the n classifiers to the test example.
2. Combine the predictions to form a binary string of length n.
3. Classify to the class with the nearest codeword

Table 1. The ECOC algorithm : m is the number of classes

3. Related Work

A variety of work has been done in the area of text classification and error-correcting output coding separately, but not many attempts have been made to combine the efforts and utilize the advantage ECOC offers in the area of text classification.

3.1 Text Classification

Most of the work in text classification involves statistical approaches such as the TFIDF (Salton, 1991) and Naive Bayes. These algorithms represent documents as vectors of features and train by calculating the frequencies and probabilities of these features within training documents belonging to classes. These features are usually single words or in some cases phrases (n-grams) (Mladenic, 1998). Other approaches such as neural networks (Wiener, Pederson & Weigend, 1995), nearest neighbor classifiers (Yang & Chute, 1994), rule-based algorithms (Apte, Damerau & Weiss, 1994; Moulinier, Raskinis, Ganascia, 1996), decision trees (Lewis & Ringuette, 1994; Moulinier, 1997) and inductive learning techniques (Cohen & Singer, 1996; Lewis, Schapire, Callan, Papka, 1996) have also been used for text classification.

3.2 Error-Correcting Output Coding

Recently there have been a number of text classification algorithms, such as Bagging and AdaBoost, that work by voting between classifiers. Error-Correcting Output coding is also a form of voting with multiple classifiers. Schapire et al (1998) describe it as every classifier voting on the class and the class receiving the most votes is the winner. It works by converting a k-class supervised learning problem into a large number L of two-class supervised learning problems. Any learning algorithm that can handle two-class learning problems, such as the decision-tree algorithm C4.5 (Quinlan, 1993), can then be applied to learn each of these L problems. L can then be thought of as the length of the codewords with one bit in each codeword for each classifier.

Error-correcting codes have been used by Diettrich and Bakiri with decision trees and neural networks on the glass, vowel, soybean, audiologyS, ISOLET, letter and NETtalk data sets available from the Irvine Repository of machine learning databases (Murphy & Aha, 1994). With artificial neural networks, the functions can be implemented by the output units of a single network. With decision trees, separate decision trees are learned, one for each bit position in the output code (Kong & Diettrich, 1995). They have shown the ECOC approach to improve the performance of both learning algorithms and to provide improvements even with very small sample sizes.

Berger (1999) applies the ECOC approach to text classification and reports results on several datasets that are encouraging. They also give some theoretical evidence for the use of random rather than error-correcting codes. Our work differs from Berger (1999) in that it gives a more detailed experimental study using various types of codes and also gives empirical evidence for using error-correcting codes rather than random codes.

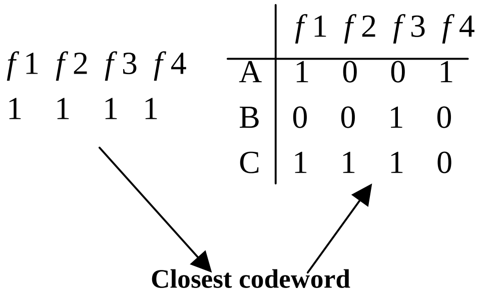

Figure 1. Test Phase of the ECOC Algorithm

4. Problem Domain

In this section, we describe the data sets used in our experiments.

4.1 Industry Sector Dataset

The Industry Sector dataset, based on data made available by Market Guide Inc. (www.marketguide.com), consists of company web pages classified in a hierarchy of industry sectors. The data is publicly available at http://www.cs.cmu.edu/~TextLearning/datasets.html. We do not take the hierarchy into account in our experiments and use a flattened version of the dataset. This dataset contains a total of 9555 documents divided into 105 classes. A small fraction of these documents belongs to multiple classes but in our experiments we remove these documents[2]. In tokenizing the data, we skip all MIME and HTML headers, use a standard stoplist, and do not perform stemming. This procedure is the same as in McCallum et al (1998) who use a slightly modified version of this data set. After removing tokens that occur only once, the corpus contains 1.2 million words with a vocabulary size of 29964.

5. Approach

In all the experiments conducted, the base classifier used to learn each bit of the codeword was the Naive Bayes Classifier. The implementation used was *Rainbow* developed by Andrew McCallum and available from http://www.cs.cmu.edu/~mccallum/bow.

Feature selection, when used, was done by selecting the words with the highest mutual information gain. The codes used in the next section were constructed using the BCH method which uses algebraic coding theory to generate codes well-separated in hamming distance between rows. More information about BCH codes can be found in Error-Correcting Codes literature such as (Hill, 1986; Peterson & Weldon, 1972; Pless, 1989). The codes used for these experiments are available online at http://www.cs.cmu.edu/~rayid/ecoc.

[2] Only 15 documents out of 9555 belong to two classes so they can be removed from the dataset without affecting our results considerably.

6. Experimental Results

This section describes the experiments performed using ECOC and also contains an explanation of the results. We compare our results with the performance of the Naive Bayes Classifier, one of the most commonly used algorithms for text classification. In our experiments we use classification accuracy as the evaluation measure. Other papers have proposed different performance measures like precision-recall graphs or break-even point of recall and precision (Lewis, 1991) but since the goal of a classification algorithm is to achieve a low misclassification rate on a test set (Schapire et al, 1998), accuracy is the best measure of performance for our purposes.

In the following experiments, we use the Naive Bayes Classifier to learn the individual functions for each bit of the code. Each class has a unique binary code of the same length. To classify a test instance, we test it on all the individual bit classifiers and combine the output and then compare it to the codes for the class. The test instance is assigned to the class with the nearest codeword with ties broken randomly.

6.1 Do Error-Correcting Codes Improve Classification Accuracy?

Table 2 shows the performance of the ECOC approach vs. a single Naive Bayes Classifier in the Industry Sector dataset with repeated random 50-50 train-test splits. The vocabulary size for these experiments was 10000 words with feature selection being performed by selecting the words that have the highest mutual information gain given the class. As we can observe from the table, ECOC reduces the classification error of the Naive Bayes Classifier by 66% which is currently the lowest error-rate on this dataset. McCallum et al (1998), using Shrinkage techniques on a slightly modified version of this dataset along with its hierarchical structure, achieved a classification accuracy of 76% as compared to 88% through ECOC.

Table 2. Classification Accuracies using the same code on the Industry Sector dataset with a vocabulary size of 10000.

METHOD	TRIAL 1	TRIAL 2	TRIAL 3	TRIAL 4
NAÏVE BAYES CLASSIFIER	64.5	64.8	65	66.1
63-BIT ECOC	88.1	87.9	88.2	88.6

6.2 How Does The Length Of The Codes Affect The Classification Accuracy?

Table 3 shows the classification accuracy on the Industry Sector dataset for codes of different lengths. We can clearly observe that increasing the length of the codes increases the classification accuracy. However, the increase in accuracy is not directly proportional to the increase in the length of the code. As the codes get larger, the accuracies start leveling off as we can observe from table 3. The flattening of the accuracy curve as the codes get longer is corroborated by the results reported by Berger (1999).

The increase in accuracy with the code length can be explained by the fact that as we increase the length of the codes, the error-correcting properties are also enhanced. A good error-correcting code has large row and column separation. In other words, each individual codeword should be separated from each of the other codewords with a large Hamming distance and the columns (the functions learned by the separate classifiers) should also be different as learning the same functions would cause the individual classifiers to make the same mistakes in multiple bits which hinders the error-correcting code from correcting them.

Table 3. Average classification accuracies on 10 random 50-50 train/test splits of the 105-class Industry Sector dataset with a vocabulary size of 10000.

METHOD	CLASSIFICATION ACCURACY
NAÏVE BAYES CLASSIFIER	64.5
15-BIT ECOC	77.4
31-BIT ECOC	83.6
63-BIT ECOC	88.1

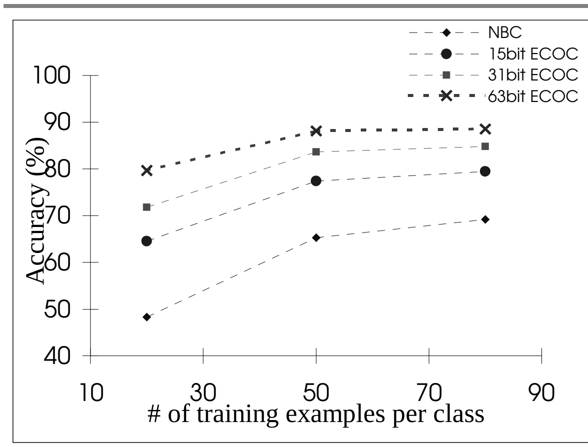

Figure 2. Performance of ECOC while varying the number of training examples.

If the minimum hamming distance of a code C is m, then that code is an $\lfloor (m-1)/2 \rfloor$ error-correcting code. So if our individual bit classifiers make $\lfloor (m-1)/2 \rfloor$ or fewer errors, the resulting codeword is closer to the correct one than it is to any other codeword and can be correctly decoded.

The longer a code is, the more separated the individual codewords can be, thus having a larger minimum hamming distance and improving the error-correcting ability. This can be seen from the table 3 as the increase in the length of the codes reduces the classification errors.

We could assume from looking at table 3 that as we keep on increasing the code length, the classification accuracy would also keep on increasing. That is not the case in practice, as this would only be possible if the errors made by the individual bit classifiers were completely independent which is not the case. The errors are dependent since there is a lot of overlap in the data and feature set being used to learn those binary functions and so after a certain code length is exceeded, the classification accuracy starts leveling off.

6.3 How Does The Number Of Training Examples Affect Accuracy?

Figure 2 shows the results of our experiments while varying the number of training examples. Five different samples for each of 20-80, 50-50, and 80-20 train-test splits were taken and the results averaged to produce the graph. As we can see, ECOC outperforms the Naive Bayes Classifier at all times. The previous results regarding increase in classification accuracy with increase in code length still hold as longer codes give better performance in these experiments too.

Table 4 shows the average classification accuracies for the individual binary classifiers (which are just NBCs) at different sample sizes. As we increase the number of training examples, NBC shows an improved performance that is to be expected. This improvement is also present in the ECOC and can be observed from Figure 3 which shows percent reduction in error by the ECOC over the NBC at different training sizes.

Table 4. Average classification accuracies of the individual bit classifiers on 30 random splits of the 105-class Industry Sector dataset with a vocabulary size of 10000. The standard deviation is quite small (<1).

# OF TRAINING EXAMPLES PER CLASS	AVERAGE CLASSIFICATION ACCURACY OF THE INDIVIDUAL BIT CLASSIFIERS
20	84.7
50	89.9
80	90.8

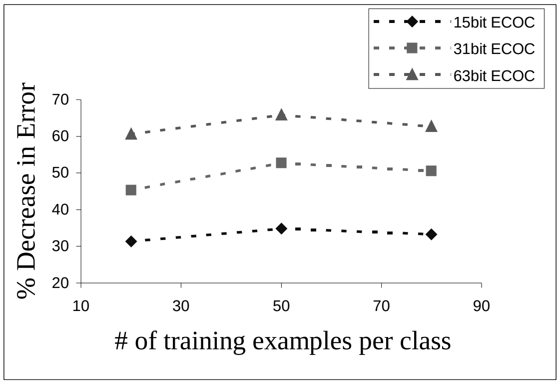

Figure 3. Percent decrease in error over NBC while varying the number of training examples.

The percent reduction in error does not vary much with the number of training examples which suggests that ECOC provides good performance and considerable reduction in error over the NBC regardless of the size of the training set and would be a good algorithm when training data is sparse.

This does not seem intuitive at first, but an analysis of the ECOC suggests that since increasing the training examples improves the performance of NBC, it also improves the performance of the individual bit classifiers since those are just NBCs (See Table 4). So the improvement in accuracy for ECOC over NBC is mainly because of the error-correcting ability which is mostly dependent on the code length and hamming distance, and not as much on the accuracy of the individual bits compared to the accuracy of a single NBC. Though the NBC is learning to classify a 105-class problem and the individual l-bit classifiers are 2-way problems, the experimental results suggest that they both improve their performance at roughly the same rate with increase in training examples and thus keep the percentage reduction in error due to ECOC the same and only dependent on the length of the code. This fact can be used for domains where training data is sparse and NBC would not perform well but using the error correcting properties of the code, the overall performance could be increased.

7. Choosing The Codewords

From the previous results in this paper, we can observe that ECOC provides a very powerful way to increase the accuracy of a collection of learners. The two central features of this approach are the learning algorithm used and the code employed. It is usually the case that the learning algorithm is domain dependent but surprisingly, the code does not have to be. A good code for our purposes is one that has both large column and row separation. There has been some discussion on the methods used for choosing codes in recently published papers using ECOC (Berger, 1999; Mayoraz & Moreira, 1997). The three approaches that we consider are:

1. Constructing codes using standard coding theory methods (BCH, Hadamard, etc)

2. Constructing Random Codes.

3. Constructing Meaningful Codes that capture or represent some feature of the dataset.

7.1 Using Coding Theory

Algebraic coding theory gives us various ways of constructing codes with good error-correcting properties. The codes used in all the earlier experiments reported in this paper were binary BCH codes which may be defined by constructing a matrix whose entries belong to a field of order 2^h and then converting this to a parity-check matrix for a binary code. More details about BCH codes can be found in (Peterson & Weldon, 1972).

The primary reason for using codes such as BCH is that for a fixed length, a certain row-separation is guaranteed. In terms of classification, this translates to having a guarantee that even if *x* of our classifiers give the wrong classification, we will classify to the correct class. Most error-correcting codes only guarantee separation between rows (codewords) but as we mentioned earlier, we also need separation between columns so that the errors made by the classifiers learning each bit are as independent as possible. This column separation is not guaranteed by BCH codes but there do exist codes such Hadamard codes that provide this guarantee. Even in the case of BCH codes, we calculated the minimum, maximum and average column separation of the codes used in our experiments and found the separation quite large.

Another reason for using codes based on coding theory is computational efficiency, since there are efficient ways of decoding rather than doing a linear search through all codewords which is the case in random codes. This may be of concern when using datasets consisting of thousands of classes.

7.2 Using Random Codes

Berger (1999) argues for the use of random codes which are generated by picking each entry in the matrix to be 0 or 1 at random. They give some theoretical results and bounds on the column and row separation of random codes and show that asymptotically in the limit, as the length of the codes gets very large (approaches infinity), the probability that the codes will have separation less than x approaches zero.

We argue that although these results hold in theory, we will not want to use such long codes in practice because of computational bottlenecks. We would prefer short codes that guarantee us a certain column and row separation. To compare the error-correcting properties of random codes with our constructed codes (BCH codes), we generated random codes and calculated the minimum, maximum and average row and column separations or them. We found that for a fixed length of the code, the BCH codes outperformed the random codes significantly. Similarly we also used random codes to classify the Industry Sector dataset and found the results to be worse than BCH codes used in the earlier experiments. On average, the ECOC method with random codes had an error rate 10-15% greater than that with algebraic codes.

7.3 Using Meaningful (Domain and Data-specific) Codes

The two types of codes described above give good classification results even though they split the set of classes into two disjoint sets a priori without taking the data into account at all. This section explores alternatives to this method which are still based on the error-correcting notion, but where the binary functions learned by each of the classifiers are inspired by the data at hand.

It would be quite logical to believe that functions that partition the classes into groups that have similar classes within themselves would be easier to learn than random functions. For example, if we have 4 classes such as Football, Rugby, U.S. Politics, and World Politics, we would expect a function that combines Football and Rugby in one class and U.S. Politics and World Politics in another class to be easier to learn then one which groups Rugby and U.S. Politics together.

To test this idea on the industry Sector dataset and compare results with the codes used earlier in this paper, we constructed a 4-bit code that contained binary problems that grouped together related economic sectors. Since the 4-bit code would not give us the row separation we need for error-correcting, we appended these 4 bits to the 15-bit BCH code resulting in a 19-bit code.

Table 5. Minumum and Maximum Hamming Distance information between rows and columns for 15-bit BCH code and 19-bit hybrid code with 105 codewords and classification accuracy for the Industry Sector dataset.

Code	Min Row HD	Max Row HD	Min Col HD	Max Col HD	Accuracy
15-Bit BCH	5	15	49	64	79.4
19-Bit HYBRID	5	18	15	69	78.7

We would expect the new 19 bit code to perform much better than the 15-bit code because the new code has 4 bits which are constructed using the data at hand which should be much easier to learn for the classifier. Table 4 gives the accuracy for the two codes and we can observe that the 19-bit code performs slightly worse than the 15-bit one. If we look at how well the classifiers are learning each bit, we see that all of them have accuracies around 90% except for two of the four bits that were added which have accuracies of 97%. This is certainly expected since these are the bits that should be easier to learn but the odd thing about these bits is that the binary partitions induced by these bits don't have equal number of documents/classes in both groups. For example in bit 16, class 0 contains 7000 documents and class 1 contains only 700. In the other bits, there are almost equal number of 0s and 1s in a column. So one way to get better results would be to create 2 class problems with most of data belonging to one of the classes and that should improve the individual accuracies. The degenerate case of that would be the one-per-class approach (Diettrich & Bakiri, 1999) where only 1 bit in each column is a 1 and each column is distinguishing one particular class from the rest. In that case, we would expect each of the individual classifiers to perform extremely well but the overall accuracy would be poor since the hamming distance between codewords is just 1 and no error-correction is possible.

It is obvious that adding the 4 bits to the 15-bit BCH code can only increase the row hamming distance. However, it can reduce the column separation of the resulting code. Table 4 shows the hamming distance (HD) information of the two codes and as expected, the row HD is increased slightly but the column HD is significantly reduced. So creating columns with a lot more 0s than 1s (or vice versa) would help the individual accuracies of each bit but reduce the minimum hamming distance for the code. Guruswami & Sahai (1999) describe some attempts to combine the two approaches (one-per-class and error-correcting codes) which outperform each approach alone.

8. Discussion

Given the results from the previous sections, what can we conclude about the behavior and advantages of the ECOC approach for text classification? We saw in Table 4 the average accuracies of the individual classifiers for each bit of the code. Since the variation in the accuracies is very small for each training sample, using these values as the average probability for one bit to be classified correctly and calculating the minimum hamming distance of the code used in each of the experiments, the ECOC approach can be modeled by a Binomial Distribution B(n,p) with n being the length of the code (number of bits) and p being the probability of each bit being classified incorrectly. Then the probability of an instance being classified correctly would just follow the binomial distribution.

Table 6 shows the results of the calculation. H_{min} is the minimum hamming distance the code used and since a code with minimum distance h can correct at least (h-1)/2 errors, E_{max} is the maximum number of errors the code can correct. P is the probability that each bit classifier will classify correctly which is obtained from the experimental results. The theoretical accuracy of the ECOC method can then be calculated by using the binomial distribution. A sample calculation for the first row of Table 5 would be:

Since H_{min}=5, E_{max}= (5-1)/2=2 , even if 2 of the 15 classifiers give the wrong classification, the correct classification can still be obtained. The probability of at most 2 classifiers classifying incorrectly is:

P(0 classifiers classify incorrectly) + P(1 classifier classifies incorrectly) + P(2 classifiers classify incorrectly)=

$$\binom{15}{0}0.846^{15} + \binom{14}{1}0.846^{14}(1-.846)^{1} + \binom{13}{2}0.846^{13}(1-.846)^{2}$$

= 0.589 which suggests that we would expect the accuracy of the classifier to be 58.9%. As we can see from Figure 4, the expected accuracies are quite close to the actual results in our experiments.

Table 6. Calculating the Theoretical accuracy for the ECOC with Hmin being the minimum row hamming distance and Emax being the maximum number of errors the code can correct.

No. Of Bits	H_{MIN}	$E_{MAX} = \lfloor H_{MIN}-1/2 \rfloor$	Average Accuracy for each bit	Theoretical overall accuracy
15	5	2	0.846	58.68
15	5	2	0.895	79.64
15	5	2	0.907	84.23
31	11	5	0.847	66.53
31	11	5	0.899	91.34
31	11	5	0.908	93.97
63	31	15	0.897	99.95

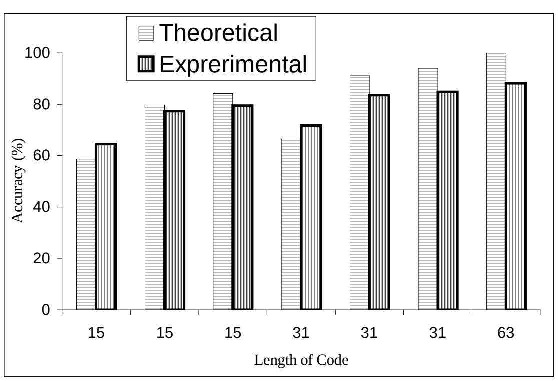

Figure 4. Comparison of theoretical accuracies with experimental results using ECOC for various code lengths and number of training examples.

9. Conclusion And Future Work

This paper has examined the use of Error-Correcting Output Coding for improving text classification. We have demonstrated that using error-correcting codes can reduce text classification error by up to 66%. Because the reduction in error over the Naive Bayes Classifier is the same for sparse training data as for large training data, this method can be used when training data is difficult or expensive to gather. Analyzing the ECOC approach and comparing it with other voting methods (Schapire et al., 1998) like Boosting and Bagging for a Naïve Bayes Classifier is an open problem and should provide valuable insight into the extensibility of this approach. Further work is also needed to experiment with various types of codes and analyze the changes in the performance of ECOC. We are currently in the process of experimenting with automatically constructing codes which would perform better than either random or algebraic ones. One such method would be to start with a random code and then choose the column that has the worst classification accuracy and modify that column randomly such that the minimum hamming distance of the code is not reduced and the accuracy of that bit is increased. This can be repeated until the increase in accuracy is below a preset threshold. Another way of constructing codes would be to cluster the classes and then create binary partitions for he code such that classes in the same cluster end up in the same “class” in the binary partition. This would create binary functions that are much easier to learn that arbitrary ones and would increase overall accuracy. Combining the ECOC approach with other algorithms which utilize disjoint feature sets such as Co-training would provide the independence between bits and allow the use of shorter codes and improve classification accuracy considerably.

Acknowledgements

The author would like to thank Tom Mitchell and Mark Craven for numerous suggestions and discussions about this work. This research was funded in part by the Tonya Internship Fund (University of the South, Sewanee, TN).

References

Apte, C., Damerau, F., and Weiss. S. (1994). Towards language-independent automated learning of text categorization problems. In Proceedings of the Seventeenth Annual ACM/SIGIR conference.

Berger, A. (1999). Error-Correcting Output Coding for Text Classification. IJCAI'99: Workshop on machine learning for information filtering.

Bose, R. & Ray-Chaudhri, D. (1960). On a Class of Error-Correcting Binary Group Codes. Information and Control,3;68--69, 1960.

Cohen, W. & Singer. Y. (1996). Context-sensitive learning methods for text categorization. In Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 307—315.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. (1998). Learning to Extract Symbolic Knowledge from the World Wide Web. In Proceedings of the Fifteenth National Conference on Artificial Intelligence.

Diettrich, T. & Bakiri, G. (1995). Solving Multiclass Learning Problems via Error-Correcting Output Codes. Journal of Artificial Intelligence Research,2;263--286, 1995.

Freund, Y., Iyer, R., Schapire, R., and Singer, Y. (1998) An efficient boosting algorithm for combining preferences. In Proceedings of the Fifteenth International Conference on Machine Learning

Guruswami, V., & Sahai, A. (1999) MulticlassLearning, Boosting, and Error-Correcting Codes. In Proceedings of the 12th Annual Conference on Computational Learning Theory.

Hill, R. (1986). A First Course in Coding Theory. Oxford University Press.

Hocuenghem, A. (1959). Codes corecteurs d'erreurs. Chiffres,2;147—156.

Kong, E., & Dietterich, T. (1995). Error-Correcting Output Coding Corrects Bias and Variance. In Proceedings of the 12th International Conference on Machine Learning.

Kong, E., & Dietterich, T. (1997). Probability estimation using error-correcting output coding. In IASTED International Conference: Artificial Intelligence and Soft Computing, Banff, Canada.

Lewis, D., & Ringuette, M. (1994). Comparison of two learning algorithms for text categorization. In Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval.

Lewis, D., Schapire, R., Callan, J., and Papka, R. (1996). Training algorithms for linear text classifiers. In the Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 298—306.

Mayoraz, E., & Moreira, M., (1997). On the Decomposition of Polychotomies into Dichotomies. In Proceedings of the Fourteenth International Conference on Machine Learning.

McCallum, A., Rosenfeld, R., Mitchell, T. and Ng, A. (1998). Improving Text Classification by Shrinkage in a Hierarchy of Classes. In Proceedings of the Fifteenth International Conference on Machine Learning.

Mitchell, T. (1997). Machine Learning. McGraw Hill Publishing.

Mladenic, D. (1998). Turning Yahoo into an Automatic Web-Page Classifier. In Proceedings of the Thirteenth European Conference on Artificial Intelligence (pp. 473-474).

Moulinier, I., Raskinis, G., and Ganascia, J. (1996). Text categorization: a symbolic approach. In Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval.

Moulinier, I. (1997). Is learning bias an issue on the text categorization problem. In Technical Report LAFORIA-LIP6, Universite Paris VI.

Murphy, P., & Aha, D. (1994). UCI repository of machine learning databases [machine-readable data repository]. Technical Report, University of California, Irvine.

Peterson, W., & Weldon, E., Jr. (1972). Error-Correcting Codes. MIT Press, Cambridge, MA.

Pless, V. (1989). Introduction to the Theory of Error-Correcting Codes. John Wiley and Sons.

Quinlan, R. (1993). C4.5: Program for Empirical Learning. Morgan Kaufmann, San Mateo, CA.

Quinlan, R. (1987). Decision trees as probabilistic classifiers. In Proceedings of the Fourth International Conference on Machine Learning.

Salton, G. (1991). Developments in Automatic Text Retrieval. Science,253;974--979.

Schapire, R., & Singer, Y. (1998). BoosTexter: A system for multiclass multi-label text categorization Unpublished.

Schapire, R., Freund, Y., Bartlett, P., and Lee, W. (1998). Boosting the Margin: A new Explanation for the Effectiveness of Voting Methods. The Annals of Statistics.

Wiener, E., Pederson, J., and Weigend, A. (1995). A Neural Network Approach to Topic Spotting. In Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95).

Yang, Y., & Chute, C. (1994). An example-based mapping method for text categorization and information retrieval. ACM Transactions on Information Systems, pages 253—277.