KDD Project Report

# Using Error-Correcting Codes for Efficient Text Classification with a Large Number of Categories

Rayid Ghani
Center for Automated Learning and Discovery,
School of Computer Science,
Carnegie Mellon University
Pittsburgh, PA 15213

## Abstract

We investigate the use of Error-Correcting Output Codes (ECOC) for efficient text classification with a large number of categories and propose several extensions which improve the performance of ECOC. ECOC has been shown to perform well for classification tasks, including text classification, but it still remains an under-explored area in ensemble learning algorithms. We explore the use of error-correcting codes that are short (minimizing computational cost) but result in highly accurate classifiers for several real-world text classification problems. Our results also show that ECOC is particularly effective for high-precision classification. In addition, we develop modifications and improvements to make ECOC more accurate, such as intelligently assigning codewords to categories according to their confusability, and learning the decoding (combining the decisions of the individual classifiers) in order to adapt to different datasets. To reduce the need for labeled training data, we develop a framework for ECOC where unlabeled data can be used to improve classification accuracy. This research will impact any area where efficient classification of documents is useful such as web portals, information filtering and routing, especially in open-domain applications where the number of categories is usually very large, and new documents and categories are being constantly added, and the system needs to be very efficient.

## 1. Introduction

The enormous growth of on-line information has led to a comparable growth in the need for methods that help users organize such information. One area in particular that has seen much recent research activity is the use of automated learning techniques to categorize text documents. Such methods are useful for addressing problems such as information filtering and routing, clustering of related documents and classification of documents into pre-defined topics, keyword tagging, word sense disambiguation, sentence parsing,. A primary application of text categorization systems is to assign subject categories to documents to support information retrieval, or to aid human indexers in assigning such categories. Text categorization components are also seeing increasing use in natural language processing systems for data extraction. Categorization may be used to filter out documents or parts of documents that are unlikely to contain extractable data, without incurring the cost of more expensive natural language processing.

Automated text categorization on the Web allows savings of human resources, more frequent updates, dealing with large amounts of data, discovery and categorization of new sites without human intervention, re-categorization of known sites when their content changes or when the taxonomy changes. In the case of Search Engines, in response to a query, a search engine might report the most relevant categories that contain significant URLs, combining available information retrieval and categorization capabilities.

The use of machine learning techniques for text classification is difficult due to certain characteristics of this domain – the very large number of input features, noise, and the large variance in the percentage of features that are actually relevant are just some of them. A relatively moderate sized text corpus can easily have a vocabulary of tens of thousands of distinct words. It is also challenging because natural language is ambiguous and the same sentence can have multiple meanings. Furthermore, it is considered good writing style to not repeatedly use the same word for a particular concept, but instead use synonyms. This means that the categorizer has to deal with a lot of words having similar meanings. Natural language text also contains homonyms; words that are spelled the same but have different meanings. The *bank* of a river, *bank* as a financial institution, to *bank* (turn) an aircraft, to *bank* a fire and so on. The classifier must in spite of all these being the same "word", be able to distinguish between different meanings, since meaning is what the classification is based on. As a result, algorithms to classify text documents need a lot of time to run and a large number of labeled training documents.

Text categorization systems attempt to reproduce human categorization judgments. One common approach to building a text categorization system is to manually assign some set of documents to categories, and then use inductive learning to automatically assign categories to future documents. Such an approach can save considerable human effort in building a text categorization system, particularly when replacing or aiding human indexers who have already produced a large database of categorized documents. In many applications of such systems such as those involving news, current affairs and web-based ones, new documents keep coming in at regular intervals and new categories are added, while old categories and documents are modified and deleted.

It is important for these applications that the system can be retrained fairly quickly and that the training scales up well with the number of categories. Naïve Bayes, which is one of the faster algorithms for multiclass text classification, scales up linearly with the number of categories. Other algorithms performing well on benchmark datasets for text such as SVMs are built for binary classification problems and the most common approach for multiclass learning with SVMs has been to decompose an n-class problem into n binary problems and build one binary classifier for each class. Such an approach also scales up linearly with the number of classes and is not efficient when dealing with hundreds or thousands of categories. Another method that decomposes multiclass problems into binary problems is the Pairwise approach where each pair of classes is distinguished using a separate classifier/model. This approach turns out to be very inefficient and expensive since an n-class problem is decomposed into (n!/2)/(n-2)! binary problems and requires (n!/2)/(n-2)! classifiers.

We use Error-Correcting Output Codes (ECOC), originally developed by Dietterich & Bakiri (1991), for efficient classification of text documents with a large number of categories. The application of ECOC to text classification is not new but our research differs from the traditional use of ECOC for improved performance at the cost of efficiency in that it specifically focuses on minimizing the computational cost while maximizing classification accuracy. ECOC has been used for several classification tasks including very limited applications to text, but previously the focus has been on improving the classification accuracy at the cost of increasing training time and computational cost. For example, Berger (1999) used ECOC for several text classification tasks and reported improved performance over Naïve Bayes (around 10% improvement) while increasing the computational cost by several factors. This increase in cost was mainly due to their use of randomly generated codes which were extremely long. We explore the use of ECOC to not only increase performance but at the same time increasing the efficiency of the system.

Our research is different from previous work done in text classification in general and with ECOC in particular in that it focuses specifically on text classification with a large number of categories (on the order of hundreds of categories) and that we are exploring ways of increasing the performance of ECOC with short-length codes. The approaches we use are:

- Specifically using short algebraic codes with good error-correcting properties that maximize accuracy while minimizing code length.
- Assign codewords to categories depending on their confusability
- Learning the decoding function rather than use the standard hamming distance.
- Develop a framework for ECOC that can enable us to use a small amount of labeled training data and augment it with large amounts of unlabeled data

Our approach is tested on several real-world text classification tasks and we find that using short, error-correcting codes results in efficient, highly accurate and high-precision text classifiers and that by learning the decoding function, we can adapt to various datasets. Our combination of ECOC and Co-Training to use labeled and unlabeled data performs well and outperforms other algorithms designed to combine labeled and unlabeled data.

## 2. Related Work

A wide range of statistical and machine learning techniques have been applied to text categorization, including multivariate regression models [Fuhr et al 1991, Schutze 1995], nearest neighbor classifiers [Yang 1994], probabilistic Bayesian models [Koller & Sahami 1997, McCallum et al. 98], decision trees [Lewis & Ringuette 1997], neural networks [Schutze 1995, Weigend et al. 1999], symbolic learning [Apte et al. 1994, Cohen & Singer 1996], ensemble learning [Schapire & Singer 2000, Ghani 2000, Berger 1999] and support vector machines [Dumais et al. 1998, Joachims 1998].

Error-correcting codes have been shown to increase the accuracy of decision trees and neural networks on several “Non-text” data sets available from the Irvine Repository (Murphy & Aha, 1994) using artificial neural networks and decision trees by Kong & Dietterich (1995). They tried several different random assignments of codewords to categories but did not see any significant performance differences. Schapire (1997) showed how AdaBoost can be combined with ECOC to yield a method that was superior to ECOC on several UCI datasets. Guruswami and Sahai (1999) propose another method combining boosting and ECOC which weights the individual weak learners differently than Schapire (1997) and show that their method outperforms Schapire’s AdaBoost.OC. Ricci and Aha (1997) applied a method that combines ECOC with feature selection for local learners that selects a different subset of features to learn from in each bit.

Berger (1999) applies the ECOC approach to several text classification problems that do not contain a large number of categories. To our knowledge, this is the only study of ECOC applied to text classification besides our previous work. They also give some theoretical evidence for the use of random rather than error-correcting codes. In previous work we have shown that “short” random codes do not perform well in practice and have given empirical evidence in favor of using error-correcting codes rather than random codes.

There has recently been a surge of work combining labeled and unlabeled data for text learning tasks such as using EM (Nigam et al. 2000) and Co-Training type algorithms (Blum & Mitchell 1998, Nigam & Ghani 2000). These studies have resulted in encouraging results showing that unlabeled data can indeed be of tremendous value but none of these studies have focused on the problem of a large number of categories, especially the work on co-training where the datasets used were mostly binary problems.

## 3. Overview of ECOC

In all of the work previously described, no attempt has been made to specifically maximize the performance of ECOC using short codes and to develop a framework where unlabeled data can be used for classifying documents in a taxonomy with a large number of categories. The focus of this research is on developing improved algorithms based on ECOC for efficient text categorization for a large number of categories. Our approach is based on extensions to ECOC that range from using short and effective codes to more elaborate algorithms that can be used to augment a limited number of labeled training examples with a large number of unlabeled data in order to improve the performance of our classifier.

The error-correcting codes and their application to classification problems can be better understood by an analogy given by Dietterich and 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.

ECOC 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 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. Each class is assigned a unique binary string of length L; we will refer to these strings as codewords (Dietterich & Bakiri, 1995). Then we train L 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).

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 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 row-separation is that we want codewords or classes to be maximally far apart from each other; the column separation is necessary because the functions being learned 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 them.

It is not clear how much column separation is actually necessary. 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 and 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.

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

```
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
```

## 4. Experimental Setup

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 (e.g. Hill, 1986; Peterson & Weldon, 1972; Pless, 1989). The codes used for these experiments are available online at http://www.cs.cmu.edu/~rayid/ecoc.

### 4.1 Datasets

The datasets used in the experiments are described in this section.

#### 4.1.1 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.[1] 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.

#### 4.1.2 HOOVERS DATASET

This corpus of company web pages was assembled using the Hoovers Online Web resource (www.hoovers.com) by obtaining a list of the names and home-page URLs for 4285 companies on the web and using a custom crawler to collect up to the first 50 Web pages on each site (in breadth first order), examining just over 108,000 Web pages. There are two sets of categories available from Hoover Online one consists of 255 classes (which we call Hoovers-255) and the other of 28 categories (Hoovers-28) which label each company with the industry sector it belongs to (e.g. Oil & Gas, Sports Manufacturers, Computer Software). These categories label companies, not particular web pages. For this reason, we constructed one synthetic page per company by concatenating all the pages (up to 50) crawled for that company. Each website is classified into one category only for each classification scheme. The most populous (majority) class contains 2% of the documents. Since there is no natural feature split available in this dataset, we randomly divide the vocabulary in two equal parts and apply Co-Training to the two feature sets. We have previously shown in Nigam & Ghani (2000) that this random partitioning works reasonably well in the absence of a natural feature split.

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

### 4.1.3 Jobs Dataset

We also use a dataset obtained from WhizBang! Labs consisting of Job titles and Descriptions organized in a two level hierarchy with 15 first level categories and 65 leaf categories. We use both classification schemes (Jobs-15 with 15 classes and Jobs-65 with 65 classes). In all, there are 132000 examples and each example consists of a Job Title and a corresponding Job Description. We consider the Job title and Job Description as the two feature sets for Co-Training.

## 4.2 Naïve Bayes

Naive Bayes is a simple but effective text classification algorithm for learning from labeled data alone [13, 14]. The parameterization given by naive Bayes defines an underlying generative model assumed by the classifier. In this model, first a class is selected according to class prior probabilities. Then, the generator creates each word in a document by drawing from a multinomial distribution over words specific to the class. Thus, this model assumes each word in a document is generated independently of the others given the class.

Naive Bayes forms maximum a posteriori estimates for the class-conditional probabilities for each word in the vocabulary V from labeled training data D. This is done by counting the frequency that word $w_t$ occurs in all word occurrences for documents $d_i$ in class $c_j$, supplemented with Laplace smoothing to avoid probabilities of zero:

$$P(w_t \mid c_j) = \frac{1+\sum_{i=1}^{|D|} N(w_t \mid d_i)P(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} N(w_s \mid d_i)P(c_j \mid d_i)} \quad (1)$$

where $N(w_t, d_i)$ is the count of the number of times word $w_t$ occurs in document $d_i$, and where $P(c_j,d_i) \in \{0,1\}$ as given by the class label. The prior probabilities of each class are calculated using Maximum Likelihood Estimation, counting over documents.

At classification time we use these estimated parameters by applying Bayes' rule to calculate the probability of each class label and taking the most probable class as the prediction. This makes use of the naive Bayes independence assumption, which states that words occur independently of each other, given the class of the document:

$$P(c_j \mid d_i) \propto P(c_j)P(d_i \mid c_j)$$

$$= P(c_j)\prod_{k=1}^{|d_i|} P(w_{d\,i,k} \mid c_j) \quad (3)$$

## 5. Results

This section describes the experiments performed using ECOC. We compare our results with the performance of naive Bayes, one of the most commonly used algorithms for text classification. We also 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 (in terms of hamming distance) with ties broken randomly.

### 5.1 Do Error-Correcting Codes Improve Classification Accuracy?

Figure 2 shows the performance of the ECOC approach vs. a single naive Bayes classifier for the Industry Sector, Jobs-15,Jobs-65, Hoovers28 and Hoovers-255 datasets. The vocabulary size for these experiments was optimized by selecting the words that have the highest mutual information gain given the class. ECOC results in higher accuracy than Naïve Bayes for all the datasets except Hoovers-255. It is noteworthy that this increase in accuracy using ECOC also comes with increased efficiency. Naïve Bayes constructs one "model" for each class and so for the 105 class Industry Sector dataset, it constructs 105 models. ECOC with 63 bits only constructs 63 models and thus reduces the computational cost of both the training and testing by almost a factor of 2. We specifically short codes to keep the computational cost down while increasing classification performance. The lower performance of ECOC on Hoovers-255 dataset can be attributed to the fact that we used a 63-bit code and increased the computational efficiency by 4 times while losing some accuracy.

*Figure 2.* Classification accuracies using ECOC on various datasets. Industry Sector and Hoovers-255 with 63-bit codes, Hoovers-28 and Jobs-65 with 31-bit codes and Jobs-15 with 15-bit code.

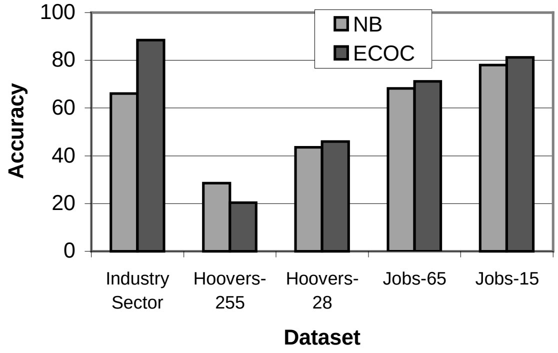
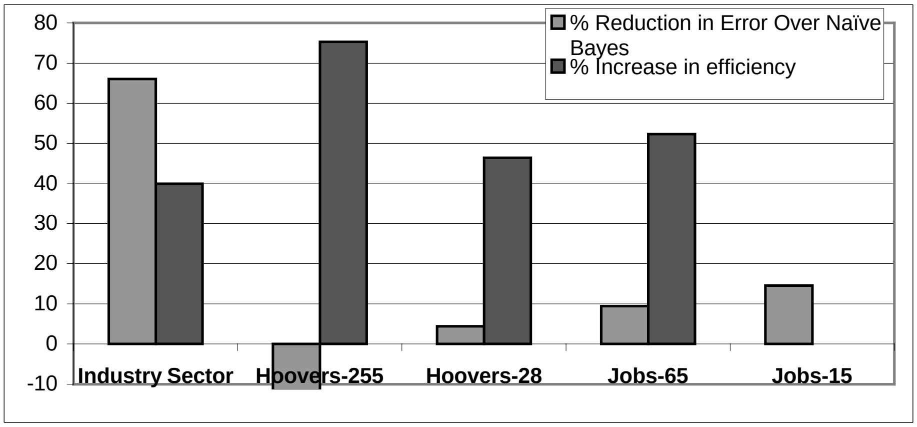


Figure 2 only gives us information about the performance of ECOC in terms of classification accuracy and does not tell us anything about the efficiency of our approach. To visualize the gain in accuracy and efficiency at the same time, Figure 3 compares ECOC with Naïve Bayes on all the datasets in terms of percent reduction in error and percent increase in efficiency. As we can observe, there is marked improvement in efficiency for all the datasets except Jobs-15 where we use a 15-bit code for a 15-class problem and thus there is no increase in efficiency.

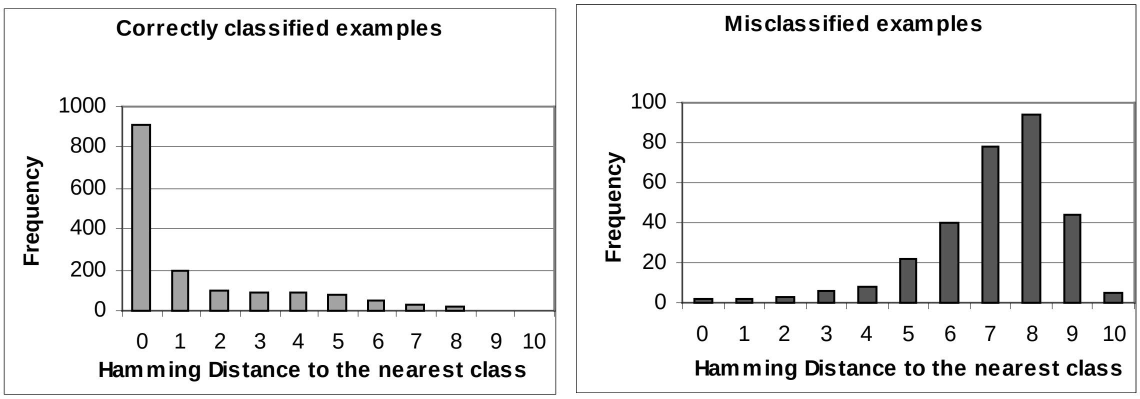
*Figure 3.* Percent reduction in error and Percent increase in efficiency using ECOC as compared to Naïve Bayes. For Jobs-15, the increase in efficiency is zero since we're using a 15-bit code for a 15-class problem.



### 5.2 Can ECOC improve the precision of Naïve Bayes classifier?

In many applications of text classification systems, it is important to get high-precision results. For example, in search engines, a typical user only looks at the top 10-50 hits returned and so instead of maximizing the overall accuracy of the system, it is more important to maximize the precision of the classifier when it only labels a small proportion of examples that its very confident about. In ECOC, we can judge how confident our system is of a prediction by looking at how far the predicted codeword is from the codeword of the nearest class. The closer to the codeword to the nearest class, the more confident the prediction. Figure 4 shows the distributions of the hamming distance to the codeword of the nearest class. It is obvious from the two graphs that the distribution for the correctly classified examples is significantly different from that for the misclassified examples. Most of the correctly classified examples correctly matched the nearest class with hamming distance zero while most of the examples that were misclassified had large hamming distance to the nearest class. Using HD to the nearest class as a measure of confidence for ECOC, we calculated the precision-recall tradeoff.

Figure 4. the distributions of the hamming distance to the codeword of the nearest class



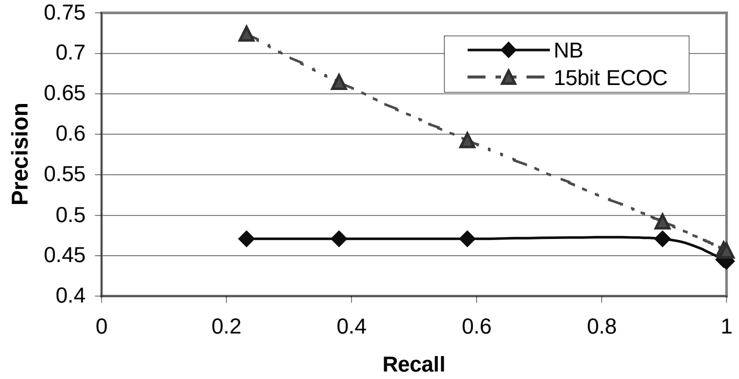


Figure 5. Precision Recall Curve for Hoovers-28 dataset with 15-bit ECOC

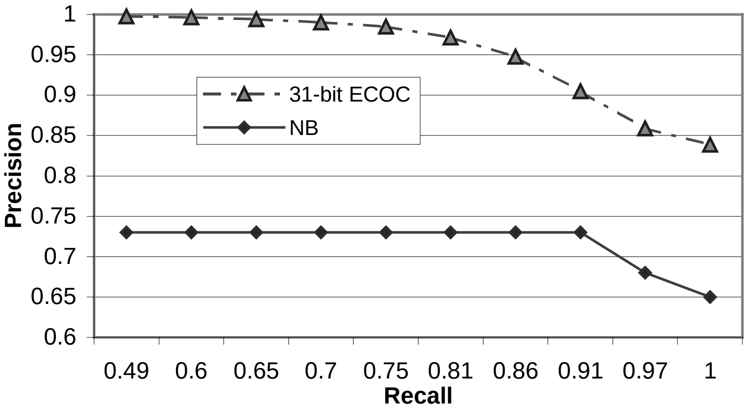


Figure 6. Precision Recall Curve for Industry Sector dataset with 31-bit ECOC

Figures 5 and 6 show this tradeoff for Hoovers-28 and Industry Sector datasets. For both datasets, ECOC results in higher precision classification compared to Naïve Bayes. Although Naïve Bayes works well for classification, it is well-known not to give accurate probabilistic estimates for text classification tasks since the number of features is usually very large and the multiplicative method of combining the evidence drives scores to either 1 or 0. Not surprisingly, Naïve Bayes does not give high-precision results and the curve is flat for recalls lower than around 90%. ECOC, on the other hand, performs extremely well and for the Industry Sector dataset, gives almost 100% precision at 45% recall level. A significant point to observe in Figure 4 is that although ECOC only improves the classification accuracy on the Hoovers-28 by 3%, ECOC performs much better at improving the precision. NB gives a maximum of 47% precision while ECOC gets over 70% precision and is computationally more efficient by a factor of 2. These results suggest that ECOC is a very useful method for combining multiple NB classifiers to give high-precision results with the added advantage that it is computationally more efficient than using standard NB models.

### 5.3 How Does The Length Of The Codes Affect Performance?

Table 2 shows the classification accuracies 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 2. This trend was also observed 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.

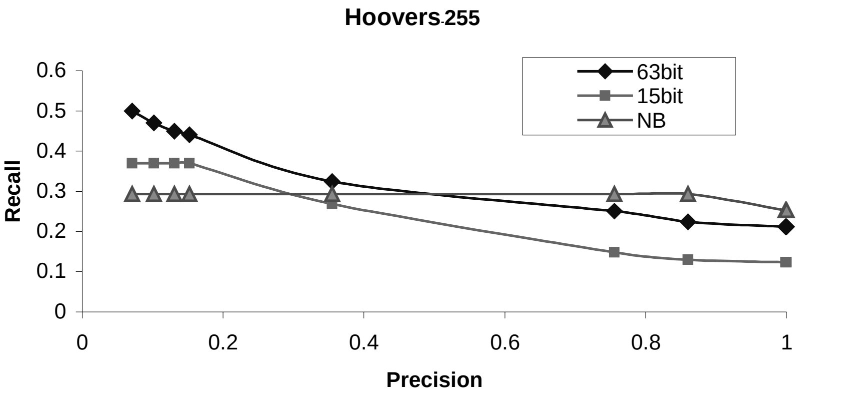
If the minimum hamming distance of a code C is m, then that code is an $\lfloor (m-1)/2 \rfloor$ error-correcting code. So if our individual bit classifiers make $\lfloor (m-1)/2 \rfloor$ or fewer errors, the resulting codeword is closer to the correct one than it is to any other codeword and can be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

*Table 2.* 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 | NAÏVE BAYES | 15-BIT ECOC | 31-BIT ECOC | 63-BIT ECOC |
|---|---|---|---|---|
| HOOVERS-255 | 28.8 | 12.5 | 17.2 | 20.4 |
| INDUSTRY SECTOR | 66.1 | 77.4 | 83.6 | 88.1 |

*Figure 7.* Precision-Recall at different code lengths for Hoovers-255 dataset.



We could assume from looking at Table 2 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.

### 5.4 How Does The Number Of Training Examples Affect Accuracy?

Figure 8 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 3 shows the average classification accuracies for the individual binary classifiers at different sample sizes. As we increase the number of training examples, naive Bayes ( learning binary problems) shows an improved performance which in turn improves the accuracy of the combined classification using ECOC. This improvement can be observed from Figure 9 which shows percent reduction in error by the ECOC over the multiclass naïve Bayes at different training sizes.

*Figure 8*. Performance of ECOC on Industry Sector dataset while varying the number of training examples.

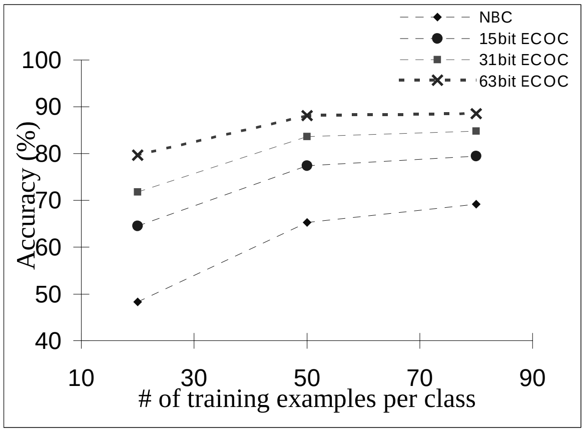
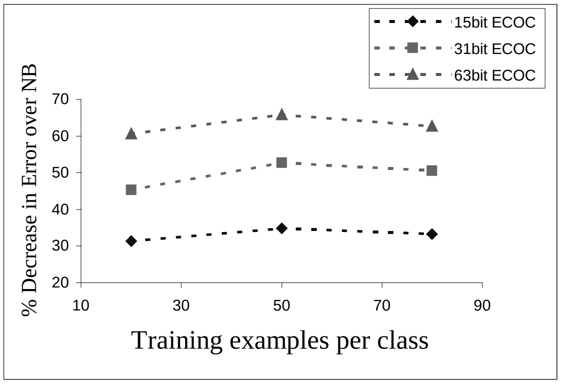


*Table 3*. Average classification accuracies of the individual bit classifiers on 30 random splits of the 105-class Industry Sector dataset with a vocabulary size of 10000. The standard deviation is quite small (<1).

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

*Figure 9*. Percent decrease in error over NB on Industry Sector dataset 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 NB 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 ECOC suggests that since increasing the training examples improves the performance of naïve Bayes, it also improves the performance of the individual binary classifiers since those are just naïve Bayes classifiers (See Table 3). The improvement in accuracy for ECOC over naïve Bayes is mainly because of the error-correcting ability which is mostly dependent on the code length and hamming distance. Though the multiclass naïve Bayes is learning to classify a 105-class problem and the individual l-bit classifiers are bianry 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.

## 6. Choosing the Codewords

From the earlier 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 data set.

### 6.1 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.

### 6.2 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, as the length of the codes gets very large (approaches infinity), the probability that the codes will have separation less than an arbitrary number approaches zero.

We argue that although these results hold in theory, we will not want to use such long codes in practice because of high computational cost. 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.

### 6.3 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 4.* 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 expected since these are the bits that should be easier to learn. The interesting point about these “good” 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. Since these “skewed” binary problems were easier to learn, this suggests that to get better results, we could create “new” binary problems with the same property. The degenerate case of that would be the one-per-class approach (Dietterich & 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 2 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 zeros than ones (or vice versa) would help the individual accuracies of each bit but reduce the minimum hamming distance for the code. Guruswami and Sahai (1999) describe some attempts to combine the two approaches (one-per-class and error-correcting codes) which outperform each approach alone.

## 7. Semi-Theoretical Model for ECOC

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 5*. 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 |

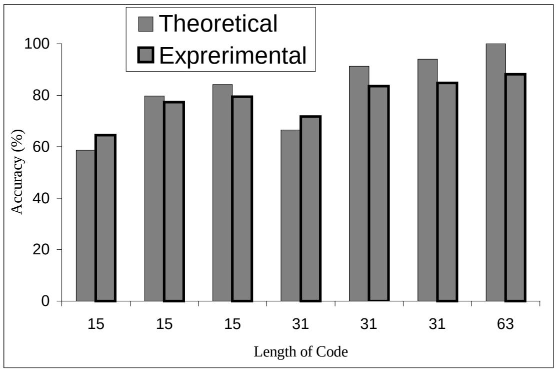


*Figure 10*. Comparison of theoretical accuracies with experimental results using ECOC for various code lengths and number of training examples.

## 8. Extensions to ECOC

Apart from applying ECOC to text classification tasks with a large number of categories and investigating its performance as we vary several conditions, we also investigate several extensions to ECOC. Though the domain will be text classification, we hope that these results will generalize to any classification task with a large number of categories.

### 8.1 Assignment of codewords to categories

To our knowledge, all of the previous work in ECOC has involved randomly assigning codewords to categories. We propose a method of assignment that adapts to particular datasets and could lead to better performance. We learn the original multiclass problem using a standard classifier that performs well for text classification tasks e.g kNN or Naive Bayes (learning one model per class) and generate a confusion matrix from the training set. This matrix tells is to what extent the classifier confuses each pair of classes. The codewords in an error-correcting code are not equidistant from each other and the two codewords that are farthest apart can then be assigned to the two classes that are most confusable (difficult to distinguish between) in the dataset. The next farthest pair of codewords gets assigned to the next most confusable classes and so on. Since at times, as we go down the list, one of the classes may already have a codeword assigned to it from before in that case, we will not reassign the class a different codeword since the higher one corresponds to a higher confusability.

We believe that this approach of assigning codewords to categories will allow a generic error-correcting code, which can applied to any problem, to adapt to a specific problem increase performance. Intuitively, since the classifiers are likely to make many errors between the two most confused classes, if the two codewords that are farthest apart (e.g. the one with all zeros and one with all ones) are assigned to these two classes, then it allows the system to recover from these errors.

One potential problem we see with this approach is that the classes that are most confusable will most probably end up in different classes (in the new binary problem). This will make the binary problem harder to learn and could potentially lead to poorer results. On the other hand it would help the decoding step by correcting more errors than before, resulting in a tradeoff between accuracy of the individual classifiers and the error-correcting properties of the code. We have noticed this tradeoff when artificially constructing domain-specific codes that correspond to the position of the classes in the class hierarchy and observed that the error-correcting property was at times more important than the difficulty of the individual binary problems (Ghani 2000).

We apply this method of assigning codewords to categories to a 10-class problem with the 10 classes chosen from the Industry Sector dataset and use a 10-bit code and as shown in Table 6, our approach performs significantly worse than assigning codewords randomly. Looking closely at the performance of the individual binary classifiers, we notice that some of them had very low accuracies and that assigning codewords intelligently made the individual binary problems much harder to learn and in turn led to a performance degradation.

Table 6. Intelligently assigning codewords to categories according to their confusability – 10bit code used on a 10-class problem (10 classes chosen from Industry Sector Dataset)

| ECOC with random codeword assignment | ECOC with “intelligent” codeword assignment |
|---|---|
| 75.4% | 56.3% |

### 8.2 Decoding Step

Like most ensemble learning algorithms, ECOC also does most of the work in decomposing the problem and the decisions of the individual classifiers are usually combined in very simple ways. One of the most common ways of decoding codes is to use hamming distance and map the “received” binary string to the nearest codeword in the code. This approach assumes that all bits are equally important and ignores how easy or hard each bit is to learn. Guruswami and Sahai (1999) showed that weighting the bits during the classification step by the accuracies on the training set gives a better classification accuracy, and they called it maximum likelihood decoding. This is again a fixed method of decoding which relies on the estimates obtained from the training set.

We propose to actually pose the combining/classification step as a learning problem and use a learning algorithm to learn the decoding. It is possible that for some datasets it would learn to give equal weights to all the bits and on others the weights would depend on the difficulty of the individual binary problems and the performance of the binary classifiers. In our experiments, we use a 3 layer Neural Network to learn the decoding. We used a 10 bit code, so the input and output layers of the NN had 10 units each (one for each bit). A training instance for the NN consisted of the (binary) codeword predicted by the learned binary classifiers as the input and the codeword of the actual categories of training example as the desired output. After the learning phase, we apply the NN to the test cases and map the codeword output by the NN to the class with the nearest codeword (in terms of Hamming Distance). Results on 10 classes chosen fro mthe Industry Sector dataset are shown in Table 7. Using the NN to learn the decoding before using Hamming Distance does not result in significant improvement over using Hamming Distance by itself. These results are only representative of the particular dataset and further experiments are needed to evaluate the merits of this potentially promising approach.

Table 7. Learning the Decoding Function – 10-bit code used on a 10-class problem (10 classes chosen from Industry Sector Dataset)

| ECOC with Hamming Distance Decoding | ECOC with using a NN to learn the decoding |
|---|---|
| 75.4% | 76.1% |

One intuition behind learning the decoding is that if one of the binary problems (bit) in the code is very hard to learn and the learner consistently classifies it wrong and gets less than 50% accuracy, the decoding step can learn that and when given the classification from that particular bit, can flip the classification to correct it and thus increase the accuracy considerably. The reason we don't observe that in our last experiments is because none of the bits were less than 50% accurate and so the NN couldn't correct the classification. We hypothesize that by intelligently assigning codewords to categories (described in the previous section) we make the binary problems harder to learn and then using a NN to learn the decoding should improve performance instead of the decrease in performance observed in the previous section.

Table 8. Learning the Decoding Function – 10-bit code used on a 10-class problem (10 classes chosen from Industry Sector Dataset)

| | | CodeWord Assignment | |
|---|---|---|---|
| | | Random | Intelligent |
| Decoding | Hamming Distance | 75.4 | 54.6 |
| | Neural Network | 76.1 | **83.2** |

We test this hypothesis on the same dataset as earlier and find that combining the two approaches does indeed improve performance considerably (See Table 8). Upon looking at the individual accuracies of the binary classifiers before and after the "learning decoding" step, we notice that there is a marked improvement in accuracies corresponding to the "bit-flipping" phenomena that we expected. This suggests that we can construct codes that have good error-correcting properties and even though the individual binary problems may not be learnable, we can use the decoding step to correct for those errors by just inverting the individual classifications.

## 9. Combining labeled and unlabeled data with ECOC

In the previous sections, we proposed ways to increase the accuracy and efficiency of the ECOC given a fixed amount of labeled training data. A different way of looking at efficiency is to focus on the amount of labeled training data that is required by the learning algorithm. A major difficulty with supervised learning techniques for text classification is that they often require a large number of labeled examples to learn accurately. Collecting labeled examples is often very costly since the labeling process has to be done manually. We would ideally prefer systems that can provide accurate classifications after labeling only a few examples, rather than thousands.

One way to reduce the amount of labeled data required is to develop algorithms that can learn effectively from a small number of labeled examples augmented with a large number of unlabeled examples. In general, unlabeled examples are much less expensive and easier to obtain than labeled examples. This is particularly true for text classification tasks involving online data sources, such as web pages, email, and news stories, where huge amounts of unlabeled text are readily available. Collecting this text can often be done automatically, so it is feasible to quickly gather a large set of unlabeled examples. If unlabeled data can be integrated into supervised learning, then the process of building text classification systems will be significantly faster and less expensive than before.

There has been recent work in supervised learning algorithms that combine information from labeled and unlabeled data. Such approaches include using Expectation-Maximization to estimate maximum a posteriori parameters of a generative model for text classification [Nigam et al. 1999], using a generative model built from unlabeled data to perform discriminative classification [Jaakkola & Haussler 1999], and using transductive inference for support vector machines to optimize performance on a specific test set [Joachims 1999]. Each of these results, and others, has shown that using unlabeled data can significantly decrease classification error, especially when labeled training data are sparse.

A related body of research uses labeled and unlabeled data in problem domains where the features naturally divide into two disjoint sets. For example, Blum and Mitchell [1998] present an algorithm for classifying web pages that builds two classifiers: one over the words that appear on the page, and another over the words appearing in hyperlinks pointing to that page.

Datasets whose features naturally partition into two sets, and algorithms that use this division, fall into the *co-training* setting (Blum & Mitchell 1998) Blum and Mitchell (1998) show that PAC-like guarantees on learning with labeled and unlabeled data hold under the assumptions that (1) each set of features is sufficient for classification, and (2) the two feature sets of each instance are conditionally independent given the class.

Published studies with Co-training type algorithms (Blum & Mitchell 1999, Nigam & Ghani 2000) have focused on small, often binary, problems and it is not clear whether their conclusions would generalize to real-world classification tasks with a large number of categories. On the other hand, Error-Correcting Output Codes (ECOC) are well suited for classification tasks with a large number of categories. However, most of the earlier work has focused neither on text classification problems (except our earlier work (Ghani 2000) and Berger (1999)), nor on algoithms which specifically deal with a large number of categories.

In this paper, we develop a framework to incorporate unlabeled data in the ECOC setup to decompose multiclass problems into multiple binary problems and then use Co-Training to learn the individual binary classification problems. We show that our approach is especially useful for classification problems involving a large number of categories and outperforms several other algorithms that are designed to combine labeled and unlabeled data for text classification.

### 9.1 The Co-Training Setting

The co-training setting applies when a dataset has a natural division of its features. For example, web pages can be described by either the text on the web page, or the text on hyperlinks pointing to the web page. Traditional algorithms that learn over these domains ignore this division and pool all features together. An algorithm that uses the co-training setting may learn separate classifiers over each of the feature sets, and combine their predictions to decrease classification error. Co-training algorithms using labeled and unlabeled data explicitly leverage this split during learning.

Blum and Mitchell [1998] formalize the co-training setting and provide theoretical learning guarantees subject to certain assumptions. In the formalization, each instance is described by two sets of features. Under certain assumptions Blum and Mitchell [1998] prove that co-training algorithms can learn from unlabeled data starting from only a weak predictor. The first assumption is that the instance distribution is *compatible* with the target function; that is, for most examples, the target functions over each feature set predict the same label. For example, in the web page domain, the class of the instance should be identifiable using either the hyperlink text or the page text alone. The second assumption is that the features in one set of an instance are *conditionally independent* of the features in the second set, given the class of the instance. This assumes that the words on a web page are *not* related to the words on its incoming hyperlinks, except through the class of the web page, a somewhat unrealistic assumption in practice.

They argue that a weak initial hypothesis over one feature set can be used to label instances. These instances seem randomly distributed to the other classifier (by the conditional independence assumption), but have classification noise from the weak hypothesis. Thus, an algorithm that can learn in the presence of classification noise will succeed at learning from these labeled instances.

### 9.2 Combining ECOC and Co-Training

We propose a new algorithm that aims at combining the advantages that ECOC offers for supervised classification with a large number of categories and that of Co-Training for combining labeled and unlabeled data. Since ECOC works by decomposing a multiclass problem into multiple binary problems, we can incorporate unlabeled data into this framework by learning each of these binary problems using Co-training.

The algorithm we propose is as follows:

- Training Phase

  1. Given a problem with m classes, create an m x n binary matrix M.
  2. Each class is assigned one row of M.
  3. Train n *Co-trained* classifiers to learn the n binary functions (one for each column since each column divides the dataset into two groups).

- Test Phase

  1.Apply each of the n single-bit *Co-trained* 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

Of course, an n-class problem can be decomposed naively into n binary problems and co-training can then learn each binary problem, but our approach is more efficient since by using ECOC we reduce the number of models that our classifier constructs. We also believe that our approach will perform better than the naïve approach under the conditions that

(1) ECOC can outperform Naïve Bayes on a multiclass problem (which actually learns one model for every class)

(2) Co-Training can improve a single Naïve Bayes classifier on a binary problem by using unlabeled data

The complication that arises in fulfilling condition 2 is that unlike normal binary classification problems where Co-Training has been shown to work well, the use of Co-Training in our case involves binary problems which themselves consist of multiple classes. Since the two classes in each bit are created artificially by ECOC and consist of many “Real” classes, there is no guarantee that Co-Training can learn these arbitrary binary functions.

Each of the two classes in every ECOC bit consists of multiple classes in the original dataset. Let’s take a sample classification task consisting of classes C1 through C10 where one of the ECOC bits partitions the data such that classes C1 through C5 are in one class (B0) and C6 through C10 are in the other class (B1). The actual classes C1 through C10 contain different number of training examples and it is possible that the distribution is very skewed. If we pick our initial labeled examples randomly from the two classes B0 and B1, there is no guarantee that we will have at least one example from all of the original classes C1 through C10. If Co-training does not contain at least one labeled example from one of the original classes, it is likely that it will never be confident about labeling any unlabeled example from that class. Under the conditions that

(1) the initial labeled examples cover every “original” class,

(2) the target function for the binary partition is learnable by the underlying classifier,

(3) the feature split is redundant and independent so that the co-training algorithm can utilize unlabeled data,

theoretically, our combination of ECOC and Co-Training should result in improved performance by using unlabeled data.

## 9.3 Descriptions of Algorithms Used

We use Naïve Bayes as the base classifier in our experiments. to learn each of the binary problems in ECOC and also as the classifier within Co-Training. We also use Expectation-Maximization (EM) algorithm to compare with our proposed approach. A short description of Naive Bayes and EM as used in our experiments is given below.

### 9.3.1 Expectation-Maximization

If we extend the supervised learning setting to include unlabeled data, the naive Bayes equations presented above are no longer adequate to find maximum a posteriori parameter estimates. The Expectation-Maximization (EM) technique can be used to find locally maximum parameter estimates.

EM is an iterative statistical technique for maximum likelihood estimation in problems with incomplete data [Dempster et al. 77]. Given a model of data generation, and data with some missing values, EM will locally maximize the likelihood of the parameters and give estimates for the missing values. The naive Bayes generative model allows for the application of EM for parameter estimation. In our scenario, the class labels of the unlabeled data are treated as the missing values.

In implementation, EM is an iterative two-step process. Initial parameter estimates are set using standard naive Bayes from just the labeled documents. Then we iterate the E- and M-steps. The E-step calculates probabilistically-weighted class labels, Pr(cj|di), for every unlabeled document. The M-step estimates new classifier parameters using all the documents, by Equation 1, where Pr(cj|di) is now continuous, as given by the E-step. We iterate the E- and M-steps until the classifier converges.

It has been shown this technique can significantly increase text classification accuracy when given limited amounts of labeled data and large amounts of unlabeled data [Nigam et al 99]. However, on datasets where the assumption correlating the classes with a single multinomial component is badly violated, basic EM performance suffers.

*Table 9.* Average classification accuracies with five-fold cross-validation for Jobs-65 and Hoovers-255 Datasets

| Dataset | Naïve Bayes<br>(No UnLabeled Data) | | ECOC<br>(No UnLabeled Data) | | EM | Co-Training | ECOC + Co-Training |
|---|---|---|---|---|---|---|---|
| | **10% Labeled** | **100% Labeled** | **10% Labeled** | **100% Labeled** | **10% Labeled** | **10% Labeled** | **10% Labeled** |
| **Jobs-65** | 50.1 | 68.2 | 59.3 | 71.2 | 58.2 | 54.1 | 64.5 |
| **Hoovers-255** | 15.2 | 32.0 | 24.8 | 36.5 | 9.1 | 10.2 | 27.6 |

## 9.4 Experimental Results

All the codes used in the following experiments are BCH codes (31-bit codes for the Jobs dataset and 63-bit codes for the Hoovers Dataset) and are similar to those used in Ghani (2000)[2].

### 9.4.1 Does Combining ECOC and Co-Training Work?

Table 2 shows the results of the experiments comparing our proposed algorithm with EM and Co-Training. The baseline results with Naïve Bayes and ECOC using no unlabeled data are also given, as well as those when all the labels are known. The latter serve as an upper bound for the performance of our algorithm.

[2] The codes used in these experiments can be downloaded from http://www.cs.cmu.edu/~rayid/ecoc/codes

From results reported in recent papers [Blum & Mitchell 1998, Nigam & Ghani 2000], it is not clear whether co-training will perform well by itself and give us any leverage out of unlabeled data on a dataset consisting of a large number of classes. We can see that both Co-Training and EM did not improve the classification accuracy by using unlabeled data on the Hoovers-255 dataset; rather they had a negative effect and resulted in decreased accuracy. The accuracy reported for EM and Co-Training was decreasing at every iteration and since the experiments were stopped at different times, they are not comparable to each other.

On the other hand, our proposed combination of ECOC and Co-Training does indeed take advantage of the unlabeled data much better than EM and Co-Training and outperforms both of those algorithms on both datasets. It is also worth noting that ECOC outperforms Naïve Bayes for both datasets and this is more pronounced when the number of labeled examples is small.

Figure 2 shows the performance of our algorithm in terms of precision-recall tradeoff. Precision and Recall are both standard evaluation measures in text classification and Information Retrieval literature. As we can see from the figure, both Naïve Bayes and EM are not very good at giving high-precision results. This is not surprising since the resulting classifier after learning with EM is a Naïve Bayes classifier which gives very skewed scores to test examples and is not good at providing accurate probabilistic estimates. Interestingly, when Naïve Bayes is used within ECOC, it results in high-precision classification at reasonable levels of recall. This result is very encouraging and of enormous value in applications which require high-precision results such as search engines and hypertext classification systems.

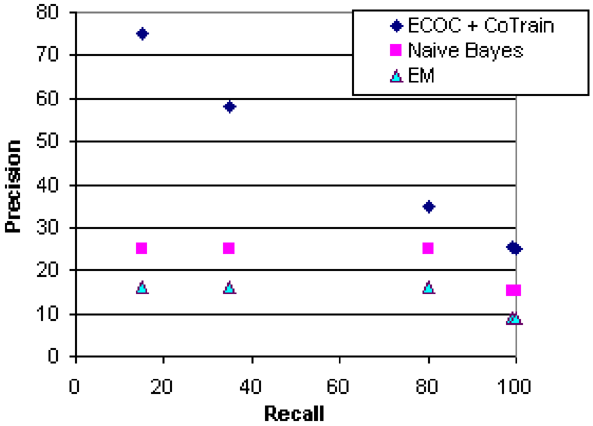


Figure 11. Precision-Recall Tradeoff for Hoovers Dataset

### 9.4.2 CAN ALGORITHMS OTHER THAN CO-TRAINING BE USED TO LEARN THE BINARY PROBLEMS CREATED BY ECOC?

The framework presented in this paper to incorporate unlabeled data into ECOC, it is not necessary to use Co-Training to learn the individual binary functions. Theoretically, any learning algorithm that can learn binary functions from labeled and unlabeled examples can be used. In this section, instead of Co-Training, we employ an algorithm named Co-EM which is a hybrid of EM and Co-Training to learn the binary problems.

#### CO-EM

Co-EM [Nigam & Ghani 2000] is an iterative algorithm that uses the feature split in a similar fashion as Co-Training does. Given a feature split with two feature sets A and B, it trains two classifiers (one for each feature set). It proceeds by initializing the A-feature-set naive Bayes classifier from the labeled data only. Then, A probabilistically labels all the unlabeled data. The B-feature-set classifier then trains using the labeled data and the unlabeled data with A's labels. B then relabels the data for use by A, and this process iterates until the classifiers converge. A and B predictions are combined together as co-training embedded classifiers are. In practice, co-EM converges as quickly as EM does, and experimentally we run co-EM for 10 iterations. The co-EM algorithm can be thought of as a closer match to the theoretical argument of Blum and Mitchell [1998] than the co-training algorithm. The essence of their argument is that an initial A classifier can be used to generate a large sample of noisily-labeled data to train a B classifier. The co-EM algorithm does exactly this using one learner to assign labels to all the unlabeled data, from which the second classifier learns. In contrast, the co-training algorithm learns from only a single example at a time.

RESULTS

Using Co-EM instead of Co-Training within ECOC performs better for the Jobs Dataset (66.1% accuracy) and worse for the Hoovers one (22.1% accuracy). The key difference between the two algorithms is that Co-EM re-labels all the unlabeled examples at every iteration while Co-Training never re-labels an example after adding it to the labeled set. The better performance of Co-EM on the Jobs Dataset may be due to the fact that the relabeling prevents the algorithm from getting stuck in local minima and makes it less sensitive to the choice of initial examples.

### 9.5 Discussion

We noted while running our experiments that our approach was very sensitive to the initial documents that are provided as labeled examples. This leads us to believe that some form of active learning combined with this method to pick the initial documents should perform better than picking random documents. Also, Co-training always adds to its labeled set the unlabeled examples about which it is most confident. This selection criterion can be modified and improved by making it more directly focused on the classification task at hand. For example, instead of always adding the most confident examples, one could balance this confidence (which minimizes the risk of adding a misclassified example) with a measure of how much will be learned from that example. McCallum & Nigam (1999) use a prototypicality measure in an active learning setting that approximately measures the benefit of labeling a particular example. This should allow co-training algorithms to work with fewer labeled examples and perform better.

As mentioned in Section 4, there is no guarantee that Co-Training can learn these arbitrary binary functions where the two classes are created artificially. If Co-training does not have at least one labeled example from one of the original classes, it is likely that it will never be confident about labeling any unlabeled example from that class. We ran some experiments using training examples that did not cover all “original” classes and as expected, the results were much worse than the ones reported in the previous section where a certain number of examples were chosen initially from every class.

There are several other ways in which ECOC and Co-Training can be combined, e.g. training two ECOC classifiers on the two feature sets separately and combining them using Co-Training. It will be interesting to pursue the other approaches in future work.

One potential drawback of any approach using Co-Training type algorithms is the need for redundant and independent feature sets. In the experiments reported in this paper, we split our feature sets in a random fashion (for the Hoovers dataset). In previous work (Nigam & Ghani 2000), we have shown that random partitions of the feature set can result in reasonable performance and some preliminary work has also been done for developing algorithms that can partition a standard feature set into two redundantly sufficient feature sets. This would extend the applicability of our proposed approach to regular datasets.

## 10. Summary

We proposed novel improvements to Error-Correcting Output Coding method for the task of efficiently text classification with a large number of categories. These improvements not only result in algorithms that are more efficient than current ones but also result in better classification accuracy. We showed that using short error-correcting codes in ECOC results in efficient, accurate and high-precision classifiers which significantly outperform naïve Bayes. We also showed that intelligently assigning codewords to categories according to their confusability, and learning the decoding (combining the decisions of the individual classifiers) improves the performance of ECOC. Our experiments with combining labeled and unlabeled data lead us to believe that the combination of ECOC and Co-Training algorithms is indeed useful for learning with labeled and unlabeled data. We have shown that our approach outperforms both Co-Training and EM algorithms, which have previously been shown to work well on several text classification tasks. Our approach not only performs well in terms of accuracy but also provides a smooth precision-recall tradeoff which is useful in applications requiring high-precision results. Furthermore, we have shown that the framework presented in this paper is general enough that any algorithm that can learn binary functions from labeled and unlabeled data can be used successfully within ECOC. This research gives us more insight into both ECOC and the problem of text classification when applied to a domain with a large number of categories. It makes on-the-fly personal categorizers possible where each individual can quickly train classifiers giving only a handful of labeled examples. It will also impact areas such as search engines, web portals, filtering systems, personalization tools, and information retrieval systems. Though our proposed work will be specifically evaluated on text collections but it is our hope that the results will generalize to any classification problem and will have an impact on large-scale classification problems.

## References

Apte, C., Damerau, F., & Weiss. S. (1994). Towards language independent automated learning of text categorization problems. *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (pp. 23-30). Dublin, Ireland: Springer-Verlag.

Berger, A. (1999). Error-correcting output coding for text classification. *IJCAI'99: Workshop on Machine Learning for Information Filtering*.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with Co-Training. *of the 1998 Conference on Computational Learning Theory*, July 1998.

Bose, R. & Ray-Chaudhri, D. (1960). On a class of error-correcting binary group codes. *Information and Control, 3*, 68-69.

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

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. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 509-516). AAAI Press / The MIT Press.

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

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

Fuhr, N., Hartmanna, S., Lustig, G., Schwantner, M., and Tzeras, K. Air/X – A rule-based multi-stage indexing system for lage subject fields. *Proceedings of RIAO'*91, 606-623, 1991.

Ghani, R (2000). Using Error-correcting coides for text classification. *Proceedings of the 17th International Conference on Machine Learning.*

Ghani, R., Slattery, S., and Yang, Y. (2001). Hypertext Categorization using Hyperlink Patterns and Meta Data *Proceedings of the 18th International Conference on Machine Learning (ICML 2001).*

Guruswami, V., & Sahai, A. (1999) MulticlassLearning, Boosting, and Error-Correcting Codes. *Proceedings of the 12th Annual Conference on Computational Learning Theory* (pp. 145-155). ACM.

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

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

Koller, D. and Sahami, M. 1997. Hierarchically classifying documents using very few words.*Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97*), 170-178, 1997.

Kong, E., & Dietterich, T. (1995). Error-correcting output coding corrects bias and variance. *Proceedings of the 12th International Conference on Machine Learning*. (pp. 313-321). Morgan Kaufmann.

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. *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. *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 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. *Proceedings of the Fifteenth International Conference on Machine Learning*.

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

Moulinier, I., Raskinis, G., and Ganascia, J. (1996). Text categorization: a symbolic approach. *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. 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.

Nigam, K., McCallum A., Thrun, S., and Mitchell, T. (2000) Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3). pp. 103-134.

Nigam, K. and Ghani, R. (2000) Analyzing the Effectiveness and Applicability of Co-training. *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM-2000)*

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. *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*.

Schütze, H., Hull, D. and Pedersen, J.O. A comparison of classifiers and document representations for the routing problem. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, 229-237, 1995.

Wiener, E., Pederson, J., and Weigend, A. (1995). A neural network approach to topic spotting. *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*.

Yang, Y., & Chute, C. (1994). An example-based mapping method for text categorization and information retrieval. *ACM Transactions on Information Systems*, 3, 252-295.