Improving Multiclass Text Classification with Error-Correcting Output Coding and Sub-class Partitions*

Baoli Li and Carl Vogel

School of Computer Science and Statistics
Trinity College Dublin, Ireland
{baoli.li,vogel}@tcd.ie

Abstract. Error-Correcting Output Coding (ECOC) is a general framework for multiclass text classification with a set of binary classifiers. It can not only help a binary classifier solve multi-class classification problems, but also boost the performance of a multi-class classifier. When building each individual binary classifier in ECOC, multiple classes are randomly grouped into two disjoint groups: positive and negative. However, when training such a binary classifier, sub-class distribution within positive and negative classes is neglected. Utilizing this information is expected to improve a binary classifier. We thus design a simple binary classification strategy via multi-class categorization (2vM) to make use of sub-class partition information, which can lead to better performance over the traditional binary classification. The proposed binary classification strategy is then applied to enhance ECOC. Experiments on document categorization and question classification show its effectiveness.

Keywords: Text Classification, Error Correcting Output Coding, Binary Classification.

1 Introduction

Text classification aims at assigning one or more predefined categories to a textual segment. As an implicit reasoning mechanism, text classification is widely used in Natural Language Processing and Information Retrieval, such as document categorization, spam filtering, sentiment analysis, question classification, textual entailment recognition, named entity recognition, and so forth. In the past years, many algorithms, which include Naïve Bayes and Support Vector Machines, have been proposed and successfully used in dealing with different text classification tasks [1].

According to whether one textual segment can belong to more than one class, text classification problems are divided into two main categories: single-label multi-class categorization (in which any item may have only a single label; often called just *multi-class* categorization) and multi-label multi-class categorization (in which items may be

* This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Trinity College Dublin.

A. Farzindar and V. Keselj (Eds.): Canadian AI 2010, LNAI 6085, pp. 4–15, 2010.

cross-classified with multiple labels; AKA *multi-label* categorization). Herein we focus on multi-class categorization in which each segment only belongs to one class.

Error Correcting Output Coding (ECOC) provides a general framework to transform a multi-class problem into a set of binary classification problems [2, 3]. It can help a binary classifier solve multiclass classification problems, and, moreover, boost the performance of a multiclass classifier for dealing with this kind of problem. In ECOC, each class is assigned a unique codeword – a binary string of length *L*. With each bit *i* of these codewords, the original multi-class dataset will be split into two mixed classes: one contains all samples of the classes that have value 1 at bit *i* of their codewords, and the other has all the remaining samples. *L* binary classifiers (one for each bit) are learned for classifying a new sample and producing a codeword for it. The predicted class is the one whose codeword is closest to that produced by the classifiers according to a distance metric[1]. Figure 1 shows an example of ECOC classification (5 classes and *L*=10).

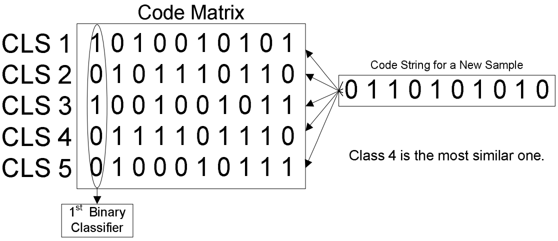
Code Matrix
CLS 1 1 0 1 0 0 1 0 1 0 1
CLS 2 0 1 0 1 1 1 0 1 1 0
CLS 3 1 0 0 1 0 0 1 0 1 1
CLS 4 0 1 1 1 1 0 1 1 1 0
CLS 5 0 1 0 0 0 1 0 1 1 1
Code String for a New Sample
0 1 1 0 1 0 1 0 1 0
Class 4 is the most similar one.
1st Binary Classifier

Fig. 1. An example of ECOC classification

Reconsidering the generation of each binary classifier in the ECOC framework, a problem is evident: when we build a binary classifier corresponding to a column of the codeword matrix, several heterogeneous classes may be combined together as a positive or negative class, and a relatively homogeneous class may be separated into different classes (one positive and one negative). Effectively, the sub-class partition information within positive and negative classes is ignored. *A priori* we do not expect an optimal classifier to emerge. ECOC, itself, has a strong correcting capacity, but why not use a better classifier if one exists? It is possible to improve binary classifiers on each bit and thereby boost ECOC.

In this study, we propose a simple strategy to improve binary text classification via multi-class categorization (dubbed 2vM) for applications where sub-class partitions of positive and/or negative classes are available. As multi-class categorization may implicitly capture the interactions between sub-classes, we expect that detailed sub-classes will help differentiating the positive and negative classes with high accuracy. With carefully designed experiments, we empirically verified this hypothesis. After that, we integrate this strategy into the ECOC framework with hill-climbing local search. Experiments on document categorization and question classification demonstrate the effectiveness of the enhanced ECOC framework.

[1] For example, hamming distance counts the number of bit differences in codes.

The rest of this paper is organized as follows: in section 2, we investigate empirically whether we can improve binary text classification strategy when we use known sub-class partition information within positive and/or negative classes. Then, in section 3, our 2vM strategy with hill-climbing search is integrated into an ECOC framework. Experiments on document categorization and question classification, in section 4, demonstrate that the enhanced ECOC framework can lead to better performance. Related work and conclusions are given in section 5 and section 6, respectively.

2 Binary Classification via Multi-class Categorization

We present a binary classification strategy that utilizes sub-class partition information within positive and/or negative classes.

2.1 Method

Our proposed binary classification strategy targets solving a special kind of binary classification problem, where positive and/or negative classes may consist of several sub-classes. Suppose that the positive and negative classes in a binary classification problem contain $|P|$ and $|N|$ sub-classes, respectively, where $P=\{p_1, p_2, \ldots, p_{|P|}\}$ and $N=\{n_1, n_2, \ldots, n_{|N|}\}$. Our strategy then works as follows:

a). Build a multi-class classifier C_m, which considers $|P|+|N|$ sub-classes.
b). Classify a new item α with the learned classifier C_m outputting prediction c.
c). If c belongs to P, then label α as positive; otherwise, label α as negative.

If the multi-class classifier C_m supports probability outputs, the probability sums of sub-classes within P and N will be used for final decision. This binary classification strategy is expected to work with any multi-class categorization algorithm.

2.2 Experiments

2.2.1 Dataset

To evaluate the effectiveness of the proposed strategy, we experiment with the 20 Newsgroups dataset. This dataset is nearly evenly partitioned across 20 different newsgroups, each corresponding to a different topic. Among the different versions of this dataset, we use the *bydate* version[2] which is sorted by date and divided into a training set (60%) and a test set (40%), without cross-posts (duplicates or multi-labeled documents) nor newsgroup-identifying headers. The total number of documents in the "bydate" version is 18,846, with 11,314 for training and 7,532 for testing. We choose this dataset for experiments because it is almost balanced and without multi-label cases. We hope to remove the effects caused by these two factors.

2.2.2 Experimental Design

To obtain binary datasets, we randomly choose one or more original classes, combine those into a positive class and take the rest to form its complementary negative class.

[2] `http://www.ai.mit.edu/~jrennie/20Newsgroups/-` last verified January 2010.

With the 20 newsgroups dataset, we have in total $C_{20}^{1}+C_{20}^{2}+\dots+C_{20}^{10}$ possible separations. They can be classified into ten types: 1vs19 (1 class as positive and the rest as negative), 2vs18, 3vs17, …, and 10vs10. The number of possible separations is huge (616,665). To make our experiments tractable, we randomly choose 100 separations from different types. Obviously, we can only have 20 different separations of type 1vs19. Totally we experiment with 920 different separations. Separations from the same type roughly have the same class distribution.

We compare our proposed strategy with the traditional binary classification strategy that doesn't consider sub-class partition information even though it is available. We label the traditional strategy BIN and call our proposal considering sub-class information 2vM.

We experiment with two widely used text categorization algorithms: Naïve Bayes with a multinomial model (NBM) [4] and Support Vector Machines (SVM). For SVM, we use a robust SVM implementation, LIBSVM[3], with a linear kernel and default values for other parameters. LIBSVM employs 1-against-1 strategy to extend the original binary SVM classifier to deal with multi-class categorization.

In preprocessing, we remove stop words without stemming. Words with document frequency below 2 are ignored. This leaves in total 49,790 words as features. For SVM, we use TFIDF weighting schema as (log(*TF*)+1)*log(*N*/*DF*), where *TF* is the frequency of a feature in a document, *DF* is the document frequency of a feature in a dataset, and *N* is the total number of documents. Micro-averaging and macro-averaging F-1 measures are used to evaluate performance, and paired t-test (two tailed) is used for significance analysis.

2.2.3 Results and Discussions

Figures 2 and 3 show the performance of the two binary text classification strategies with Naïve Bayes algorithm and linear SVM algorithm, respectively. The values are the averages of all separations of the same type. Figure 1 also shows the performance of a variant of 2vM (label as 2vMp), which uses the probability output for decision. 2vMp and 2vM have very close performance, although the former is statistically significantly better than the latter (P-values are 1.345E-142 and 0 for Mic-F1 and Mac-F1, respectively).

Both figures demonstrate the effectiveness of our simple strategy, binary text classification with sub-class information. The difference of Mic-F1 between BIN and 2vM grows larger as the dataset becomes more balanced, while the difference of Mac-F1 keeps stable. With the extremely imbalanced separation (1vs19), the Mic-F1 of the 2vM strategy is just a little higher than that of the traditional BIN strategy (0.9789 vs 0.9684 with Naïve Bayes, and 0.9842 vs 0.9810 with SVM). As the dataset changes from imbalanced to balanced, Mic-F1 sharply worsens, while Mac-F1 steadily improves.

For Mic-F1, this is not unexpected, as we can easily get a higher accuracy with an extremely imbalanced dataset by simply outputting the major one of the two classes. If the two classes within a dataset have more equal size the problem will become harder because the uncertainty of such a dataset becomes higher. As Mac-F1 score is more influenced by the performance on rare classes, the overall average scores are poor on imbalanced datasets because classifiers often perform poorly on rare classes.

[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm` – last verified January 2010.

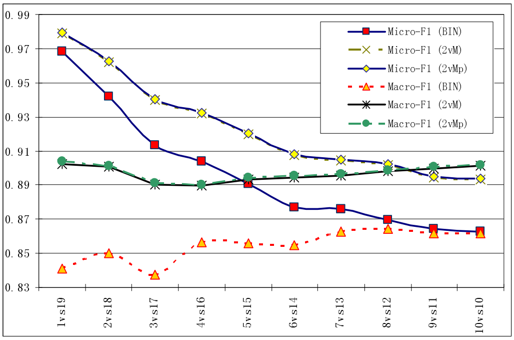

Fig. 2. F-1 measures of the two strategies on the 20 newsgroups dataset (Naïve Bayes)

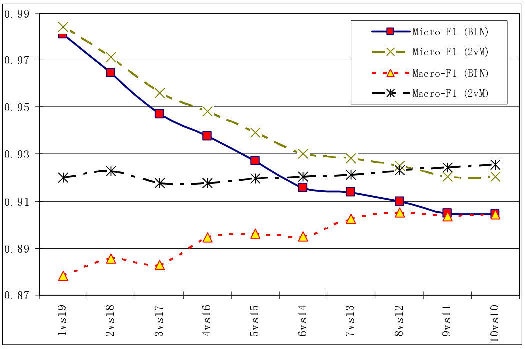

Fig. 3. F-1 measures of the two strategies on the 20 newsgroups dataset (linear SVM)

Imbalance of datasets also underlies the deviation between Mic-F1 and Mac-F1 of BIN and 2vM, which is larger for imbalanced separations than for balanced separations. As the dataset approaches balance, Mic-F1 and Mac-F1 grow very close.

The above analyses are based on the overall tendency: the values are averages of 100 runs. The performance difference between 2vM and BIN is statistically significant (e.g., with NBM, P-values of t-tests for Mic-F1 and Mac-F1 are 4.3E-269 and 8.2E-116, respectively), but this doesn't mean that on every possible separation 2vM beats BIN. With 2vM, better results are more likely: an oracle absent, it is sensible to choose our 2vM strategy.

3 ECOC with 2vM

With 2vM, we can get a better binary classifier in most cases, but, as pointed out in the previous section, there are cases in which BIN does beat 2vM. Therefore, our intuitive solution for improving ECOC is to consider each bit individually and choose the better one for each bit, either 2vM or BIN. To determine which strategy to use on each bit, we do *n*-fold cross validation on the original training data. However, this solution doesn't consistently work well in our experiments. The error correcting mechanism of ECOC is a double-edged sword: a set of binary classifiers can jointly correct errors on some bits; at the same time, they can also produce errors if they are not properly complementary even though each may perform well, individually.

Because it does not work to locally change binary classifiers for each bit due to the correcting mechanism of ECOC, we have to explore some global solution. Our problem can be formulated as a search problem as follows:

1. The search space is determined by L (the codeword length) and the binary choice at each bit: either 2vM or BIN. Thus, 2^L possible combinations exist.
2. Our goal is to find an optimal combination.
3. We choose to use accuracy as the evaluation metric.

As the length of codeword increases, the search space becomes huge, and exhaustively considering each possible combination is impossible. Therefore, we resort to a local optimal solution. A greedy hill-climbing algorithm is a simple but effective solution, as in many AI problems. The algorithm works as follows:

1. Start with all bits using BIN strategy or 2vM strategy, depending on which combination obtains better results with *n*-fold cross validation---Let's suppose all bits use BIN---this is regarded as the Base solution with accuracy *Acc*; we are trying to find a set of bits *C*, which includes the bits that should use another strategy; We use another set, *I,* to record a set of candidate bits. Initially, C=I={ }.
2. Individually consider each bit *b*:
   a) Change bit *b* with the alternative, e.g. 2vM, and evaluate the new combination (bit *b* with 2vM strategy, and other bits with BIN strategy) with *n*-fold cross validation.
   b) If the change in step 2.a) leads to improvement over the Base solution (all bits using BIN strategy), store it into set I, i.e. $I = I \cup \{b\}$.
3. Iteratively do the following until I={ }:
   a) Based on *C*, incrementally change a bit *t* in *I* with the other alternative, e.g. 2vM, evaluate the new combination ($C \cup \{t\}$) with *n*-fold cross validation, find the bit *o* that achieves the best performance *Acc_t*;
   b) If *Acc_t* <= *Acc*, break
   c) *Acc* = *Acc_t*
   d) Remove *o* from *I* and add *o* to *C*, i.e. $I = I\text{-}\{o\}$, $C = C \cup \{o\}$
4. Return an indication of which bits use 2vM and which bits use BIN.

In the above algorithm, we iterate choice of bits whose strategy is swapped, until no improvement is achieved on the cross-validation datasets.

Compared to the original ECOC with traditional binary classification, our proposed solution requires extra computation in the training stage: 1) we need to train a multiclass classifier for using the 2vM binary classification strategy; 2) *n*-fold cross validation on the original training data and hill-climbing search over a potentially huge space are used to find the best possible combination of using either BIN or 2vM strategy at each bit. With hill-climbing local search and calculating prediction value at each bit for each sample before local search, the extra training cost required by our solution can be greatly reduced.

During the running or testing stage, our proposed ECOC variant simply uses either BIN or 2vM strategy (which has been fixed after the training stage) to predict the value of each bit for a new sample. It works just like the original ECOC, and does not bring extra computation cost.

4 Experiments in Application Scenarios

We experiment with the proposed 2vM enhanced ECOC algorithm on two text classification tasks: document categorization and question classification. Longer textual segments are considered in document categorization, while shorter segments need to be processed in question classification.

4.1 Document Categorization

Document categorization aims at assigning one or more predefined classes to a document. As we mentioned earlier, in this research we focus on single-label multi-class problem, where one document belongs to only one class.

4.1.1 Datasets and Settings

We use two document categorization datasets: R52[4] and 20 Newsgroup. The 20 Newsgroup dataset has been introduced in section 2.2.1. R52 is a single-label dataset derived from Reuters-21578 with 90 classes by Ana Cardoso-Cachopo [5]. Documents with multiple labels in the original Reuters-21578 (90 classes) dataset are discarded and finally the R52 dataset contains 52 categories, 6,532 documents for training, and 2,568 documents for test. The dataset is imbalanced and some categories only have a few documents, e.g. classes *cpu* and *potato*. We use the “all-terms” version without stemming.

Naive Bayes with multinomial model algorithm is used as base classifiers. We experimented with two different kinds of codes: random codes and BCH codes. BCH codes are obtained from http://www.cs.cmu.edu/~rayid/ecoc/ecoc-codes.tar.gz. We also tried codes with different length, e.g. 15, 31, and 63.

In text classification with ECOC, how to assign codewords to classes is still an open question. Different assignments may result in different performance. Random assignment is widely used in practice. Therefore, it does not make sense to run just one experiment with a special assignment, deriving conclusions that may not generalize. Our experiments tried 100 different assignments for each setting. Accuracy averaged over 100 runs differentiates algorithms, with t-tests used to assess significance.

To determine a good combination of BIN and 2vM, we apply 3-fold cross validation on the original training data (i.e. n=3).

With the Naïve Bayes algorithm, we can get the probability output of each binary classifier at a bit. For each bit, we keep the probability that its corresponding bit in the

[4] Available at `http://web.ist.utl.pt/~acardoso/datasets/` – last verified January 2010.

codeword is one. Then we get a probability vector of length L (the length of codeword). Following [2], we compute as the distance metric the L^1 distance between this probability vector and a codeword, which can be regarded a 0/1 vector of length L. The L^1 distance is simply defined as the sum of absolute difference values between two corresponding elements in two vectors.

Table 1. Averaged accuracies on the R52 dataset

Code Type / **Length**	Random Codes			BCH Codes		
	All_BIN	**All_2vM**	**Mixed**	**All_BIN**	**All_2vM**	**Mixed**
15	**0.82458**	**0.83746**	**0.83894**	**0.84604**	**0.84941**	**0.85458**
31	**0.86558**	**0.84936**	**0.87234**	**0.86182**	**0.84939**	**0.87253**
63	**0.88448**	**0.84940**	**0.88799**	**0.89539**	**0.84937**	**0.89757**

4.1.2 Results and Analysis

Table 1 compares the averaged accuracies for three algorithms on the R52 dataset:

1. *All_BIN*: every bit uses the traditional binary classification algorithm;
2. *All_2vM*: every bit uses the proposed 2vM binary classification algorithm;
3. *Mixed*: the proposed hill-climbing solution;

From table 1, we can see that:

1) The longer the length of codes, the better performance we can get.

2) It is not always true that using BCH codes can get better results than using random codes. For example, BCH codes of length 31 didn't exhibit advantages over random codes when using the traditional binary strategy as the base classifier in ECOC, although BCH codes perform better elsewhere. This may explain why contradictory conclusions appear in the literature about BCH and random codes.

3) When using the 2vM binary strategy as the base classifier in ECOC, we obtain quite stable results. Actually, all the values are around the performance of a multiclass classifier (0.8493). If we use a hard decision function, *All_2vM* will behave as the multi-class classifier used in 2vM. There may be some small deviation due to the randomness when choosing classes in tie situations. In our experiments, we used a soft decision function with probability outputs, but the deviation is still very small.

4) *Mixed* shows advantages over *All_BIN*. The difference between *All_BIN* and *Mixed* is statistically significant (all P-values << 0.001), and it will decrease as the code length becomes larger. The proposed hill-climbing solution can derive better results, because it tries to find a good combination of BIN and 2vM. In other words, it can reach a reasonable configuration that at some bits, BIN binary classifiers will be used, where at other bits, we choose 2vM binary classifier.

5) With longer enough codes, ECOC can achieve better performance than a multiclass classifier that uses the same algorithm as the base classifier in ECOC.

Table 2. Averaged accuracies on the 20 Newsgroup dataset

Multi-Class Naïve Bayes	All_BIN	Mixed	Mixed (optimal)
0.78903	0.80998	0.81461	0.81840

Because BCH codes of length 63 achieve the highest performance, we use these codes in the following experiments. Table 2 gives the results on the 20 newsgroup dataset. The first column of table 2 shows the accuracy of the classifier using the multi-class Naïve Bayes algorithm, a baseline that we want to improve upon with ECOC. The last column of table 2 gives an optimal value with the proposed hill-climbing strategy derived by using the test data as a validation set. We can regard it as a reasonable upper bound for our proposed strategy, which determines the combination of BIN and 2vM by *n*-fold cross validation on part of the original training data. With this dataset, *Mixed* is also significantly better than *All_BIN* (P-value = 2.34E-12).

With ECOC, we can get better results than the traditional multi-class classifier. Our proposed solution to find a good combination of binary classifiers to use at each bit could further boost the results.

4.2 Question Classification

Question classification for determining the type of a question is an important step in question answering systems: the type of a question may narrow the answer search.

4.2.1 Dataset and Setting

We experiment with the dataset created by Li and Roth [6] at UIUC. 5,500 labeled questions are used for training and 500 questions from TREC-10 for testing. The total number of question types is 50. The dataset including the question taxonomy can be downloaded at http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC.

Hacioglu and Ward [7] used this dataset and tried a solution combing Support Vector Machines and Error Correcting Codes. To compare our results with those reported by them, we use SVM as the algorithm for the base classifiers in this experiment.

Due to the higher computational training cost of SVM, we run 50 times for each setting rather than 100 times as we did in the document categorization experiments.

SVM originally does not provide probability output. Although some researchers have come up with strategies to extend SVM and generate probability output, the quality of such derived probabilities may not be good enough. Therefore, we use a hard decision function rather than a soft decision function in this experiment. Accordingly, hamming distance is taken as the distance metric in this experiment. We can assume that each hard binary classifier outputs either 0 or 1 as a probability estimate, and then we can use the L^1 distance discussed in section 4.1.1.

We use all the words and symbols in the questions as features, without stemming or feature selection. BCH codes of length 63 are used. We employ t-tests for significance analysis.

Table 3. Averaged accuracies (over 50 runs) on the question classification dataset

Multi-Class SVM	All_BIN	Mixed	Mixed (optimal)
0.794	0.81608	0.82113	0.82656

4.2.2 Results and Analysis

Table 3 gives the results on this question classification problem. We used LIBSVM in our experiments as in section 2, and it can handle multi-class classification problems via 1-against-1 strategy. With this algorithm, we obtained accuracy of 0.794 on the question classification dataset. With ECOC and the traditional binary classification strategy, we got accuracy of 0.8161 (column 2), which increases by 2.78%. Our proposed hill-climbing strategy (column 3) to find a good combination of BIN and 2vM can improve the baseline by 3.42%. Mixed is significantly better than All_BIN (P-value < 0.001).

Hacioglu and Ward [7] reported the highest accuracy of 0.82 with feature selection and named entity recognition, whereas our solution can reach this level without any additional processing. They did not indicate whether the reported results are averaged over many runs. As discussed in section 4.1.1, different codeword assignments to classes lead to different results: it does not make sense to show the results of one run or a few runs.

4.3 General Discussion

A performance improvement of 0.2% to 1% is small, but usually hard won. The differences on 20 newsgroup dataset and QC dataset are ~0.5% (with 63 bits BCH codes); 0.351% and 0.218% improvements occur on R52 datasets (63 bits random codes and BCH codes, respectively). Improvement approaching or exceeding 1% obtains in other cases. The gain on R52 is relatively larger than on the other two datasets. We think this is due to the fact that the relatively high inter-class heterogeneity in R52 (which lacks the internal structure inherent in the others) reduces the dependence between the constructed binary classifiers. Further detailed scrutiny is necessary to find the true cause. It is an instance of the general problem of being able to predict how system performance will vary with the profile of the dataset. Our results on R52 show decreasing performance difference with increasing code length. The reason is that hill-climbing search reaches a local optimality quickly (on average, with 63 bits random codes on R52, only 9.52 bits select 2vM) and only small part of a huge space (longer codes, larger space) is explored before it stops. In the future, we plan to explore other local search algorithms.

5 Related Work

Error-correcting output coding was originally introduced by Dietterich and Bakiri [8] for solving multiclass categorization problems. They [2] demonstrate with extensive experiments that ECOC improves both decision trees and neural networks.

Berger explored use of ECOC to improve Naïve Bayes and Decision Tree for text categorization [9]. In that research, Berger provided some theoretical evidence for the use of random codes rather than error-correcting codes. Ghani [3] explored the use of different kinds of codes, namely Error-Correcting Codes, Random Codes, Domain and Data-specific codes. His experiments showed that using error-correcting codes can help to obtain better performance than random codes.

Rennie and Rifkin [10] compared the performance of ECOC on the task of multi-class text classification based on Naive Bayes and Support Vector Machines algorithms. They found that ECOC with Support Vector Machines performs better than ECOC with Naïve Bayes. The accuracy difference between their work and our results follows mainly from the difference in the proportion of the dataset allocated for training vs. testing (us: 60%-40%; Rennie: 80%-20%).

Tan et al. [11] used improved Centroid binary classifiers to boost ECOC with Centroid algorithm for multi-class text categorization problems. The model refinement strategy is expected to improve the original Centroid algorithm, but as we have shown empirically, simply replacing all the binary classifiers with possibly better classifiers does not guarantee overall improvement.

Crammer and Singer [12] relax discrete codes to continuous codes, while Pujol et al. [13] and Zhou et al. [14] consider how to design problem-dependent ECOC code matrix. Luo and Xiong [15] try to improve ECOC by a kernel-based decoding strategy. They propose a new scheme of defining an optimal decoding function via supervised learning.

6 Conclusion and Future Work

For years researchers have sought improved ECOC performance in three directions: 1) different base classification algorithms; 2) different kinds of codes and strategies for code assignments; 3) more effective decoding strategy (i.e. determining final predictions based on the outputs of binary classifiers). Following the first direction, we seek to improve ECOC with a better binary classification strategy, motivated by two observations: a) in ECOC, sub-class partition information of positive and negative classes is available but ignored even though it has value for binary classification; b) no one algorithm can win on every dataset and situation (in ECOC, binary classifiers corresponding to each bit are trained from datasets generated by different binary divisions of the original multi-class datasets). The immediate practical consequences of (b) are daunting because of the additional computation, but our innovation in response is a systematic scheme, using a local search method to find an optimal global configuration that stipulates methods for individual bits, rather than determining each bit locally, and yet, without a uniform method for all bits, as is current practice. Moreover, (a) constitutes a very important distinction which we feel has not yet been adequately capitalized upon in the literature.

In this paper, we empirically demonstrate the effectiveness of a simple strategy for improving binary text classification via multi-class categorization when sub-class information is available. We then apply this strategy to enhance the general multi-class classification framework Error Correcting Output Coding with hill-climbing search. Experiments on document categorization and question classification exhibit the effectiveness of the enhanced ECOC framework.

In the future, we plan to conduct more experiments on more datasets. Our approach is expected to be useful for non-text applications, but it needs to be verified with extensive experiments. We also plan to explore other local search algorithms to further improve the proposed strategy.

References

1. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
2. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. J. of Artificial Intelligence Research 2, 263–286 (1995)
3. Ghani, R.: Using Error-Correcting Codes for Text Classification. In: The Seventeenth International Conference on Machine Learning, ICML 2000 (2000)
4. Mccallum, A., Nigam, K.: A Comparison of Event Models for Naive Bayes Text Classification. In: The AAAI/ICML 1998 Workshop on Learning for Text Categorization (1998)
5. Cardoso-Cachopo, A.: Improving Methods for Single-label Text Categorization. PhD Thesis, Instituto Superior Técnico, Portugal (2007)
6. Li, X., Roth, D.: Learning question classifiers. In: The 19th International Conference on Computational Linguistics (COLING 2002), pp. 556–562 (2002)
7. Hacioglu, K., Ward, W.: Question Classification with Support Vector Machines and Error Correcting Codes. In: Proceedings of HLT-NAACL 2003 (2003) (short papers)
8. Dietterich, T.G., Bakiri, G.: Error-correcting output codes: A general method for improving multiclass inductive learning programs. In: The Ninth National Conference on Artificial Intelligence (AAAI 1991), pp. 572–577 (1991)
9. Berger, A.: Error-correcting output coding for text classification. In: IJCAI 1999 Workshop on Machine Learning for Information Filtering (1999)
10. Rennie, J., Rifkin, R.: Improving Multiclass Text Classification with the Support Vector Machine. Massachusetts Institute of Technology, AI Memo, AIM-2001-026 (2001)
11. Tan, S., Wu, G., Cheng, X.: Enhancing the Performance of Centroid Classifier by ECOC and Model Refinement. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 458–472. Springer, Heidelberg (2009)
12. Crammer, K., Singer, Y.: Improved Output Coding for Classification Using Continuous Relaxtion. In: Neural Information Processing Systems (NIPS 2000), pp. 437–443 (2000)
13. Pujol, O., Radeva, P., Vitria, J.: Discriminant ECOC: A Heuristic Method for Application Dependent Design of Error Correcting Output Codes. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1007–1012 (2006)
14. Zhou, J., Peng, H., Suen, C.Y.: Data-driven Decomposition for Multi-class Classification. Pattern Recognition 41, 67–76 (2008)
15. Luo, D., Xiong, R.: An improved error-correcting output coding framework with kernel-based decoding. Neurocomputing 71, 3131–3139 (2008)