Combining labeled and unlabeled data for text classification with a large number of categories

Rayid Ghani RAYID@CS.CMU.EDU
Center for Automated Learning and Discovery, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

A major concern with supervised learning techniques for text classification is that they often require a large, often prohibitive number of labeled examples to learn accurately. 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 this paper, we develop a framework to incorporate unlabeled data in the Error-Correcting Output Code setup by decomposing multiclass problems into multiple binary problems and then use Co-Training to learn the individual binary classification problems. We show that our method is especially useful for classification tasks involving a large number of categories where Co-training doesn't perform very well by itself and when combined with ECOC, outperforms several other algorithms that combine labeled and unlabeled data for text classification.

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. A major difficulty with supervised learning techniques for text classification is that they often require a large, often prohibitive number of labeled examples to learn accurately. Since a human must often do the labeling manually, collecting a large number of labeled examples can be very expensive. Lang (1995) worked on the problem of learning user preferences for newsgroup articles and found that after a person read and labeled about 1000 newsgroup postings, a classifier achieved a precision of about 50% when predicting for only the top 10% of documents about which it was most confident. Most users of a practical system, however, would not have the patience to label a thousand articles--especially to obtain only this level of precision. We would prefer algorithms that can provide accurate classifications after labeling only a few articles, 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 problems which 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.

2. Error Correcting Output Codes

Error Correcting Output Codes (ECOC) has been shown to perform extraordinarily well for text classification [Ghani 2000, Berger 1999]. 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 *n*; we will refer to these strings as codewords (Dietterich & Bakiri, 1995). Then we train *n* classifiers to predict each bit of the string. The predicted class is the one whose codeword is closest to the codeword produced by the classifiers. The distance metric we use in our experiments is the Hamming distance which counts the number of bits that the two codewords differ by. This process of mapping the output string to the nearest codeword is identical to the decoding step for error-correcting codes (Bose & Ray-Chaudhri, 1960; Hocuenghem, 1959).

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.

- 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 the base classifier 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 classifiers to the test example.
2. Combine the predictions to form a binary string of length n.
3. Classify to the class with the nearest codeword

Table 1: ECOC Algorithm

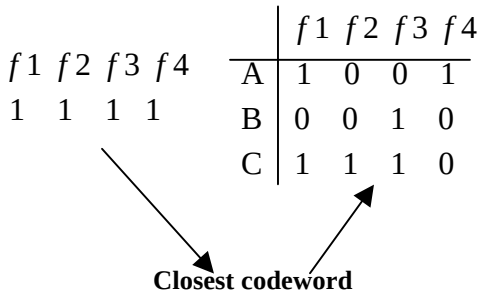

Figure 1. Test Phase of the ECOC Algorithm

ECOC has 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). 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 (Ghani 2000), we have shown that "short" random codes do not perform well in practice and have given empirical evidence for using error-correcting codes rather than random codes.

3. 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.

4. 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.

5. 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.

5.1 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:

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

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

5.2 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 2. Average classification accuracies with five-fold cross-validation for Jobs-65 Dataset

DATASET	NAÏVE BAYES (NO UNLABELED DATA)		ECOC (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

6. Datasets

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

6.1 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. The set of categories available from Hoover Online consists of 255 classes and 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 web-site 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.

6.2 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. 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 two independent and redundant feature sets for Co-Training.

7. 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)[1].

7.1 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.

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

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

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.

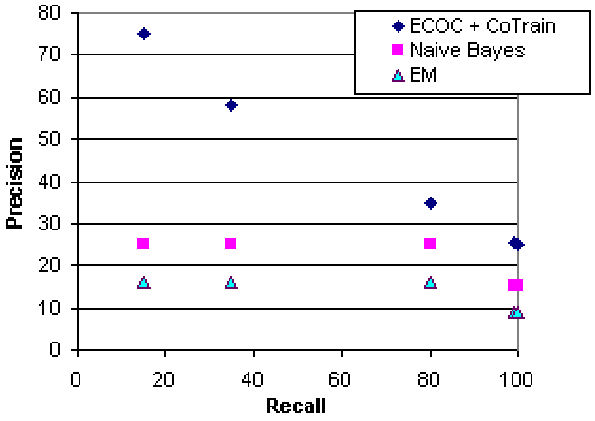

Figure 3. Precision-Recall Tradeoff for Hoovers Dataset

7.2 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.

7.2.1 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.

7.2.2 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.

8. Discussion and Future Work

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.

9. Conclusions

The results described in this paper 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.

References

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

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT '98*, 1998.

C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using SMART: TREC 4. In *Proceedings of the TREC 4 Conference*, 1996.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of EMNLP*99*, 1999.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1{38, 1977.

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

P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classi_er under zero-one loss. *Machine Learning*, 29:103{130, 1997.

R. Ghani. Using Error Correcting codes for text classification. *In the Proceedings of ICML 2000*, 2000.

Guruswami, V., & Sahai, A. (1999) Multiclass Learning, 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.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in NIPS 11*, 1999.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of ICML '99*, 1999.

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.

D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98*, 1998.

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

A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998. Tech. rep. WS-98-05, AAAI Press.

A. McCallum and K. Nigam. Employing EM and Pool-Based Active Learning for Text Classification. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML '98)*. 1998.

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

K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of CIKM-2000*, 2000.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classifcation from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103{134, 2000.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL-95*, 1995.