KDD Lab Project Proposal

Efficiently Classifying Text Documents into 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 propose to investigate the use of Error-Correcting Output Codes (ECOC) for efficient text classification with a large number of categories. ECOC has been shown to perform well for classification including some text classification ones but it still remains an under-explored area in ensemble learning algorithms. We propose to explore approaches to create generic and domain-specific codes that are short (minimizing computational cost) but result in highly accurate classifiers. In addition, we plan to develop modifications and improvements to make ECOC more efficient and accurate such as intelligently assigning codewords to categories according to their confusability and to learn the decoding (combining the decisions of the individual classifiers) in order to adapt to different datasets. In order to reduce the need for labeled training data, we propose to develop a framework for ECOC where unlabeled data can be used to improve classification accuracy. We also propose to study why ECOC is particularly beneficial for text classification in order to gain insights into the applicability and effectiveness of this approach. This research would 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 training the classifier has to be fast.

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: the task of automatically assigning natural language texts to predefined categories. Such methods are useful for addressing problems including, but not limited to keyword tagging, word sense disambiguation, information filtering and routing, sentence parsing, clustering of related documents and classification of documents into pre-defined topics. 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. They also can be used to route texts to category-specific processing mechanisms, and even to generate fillers for some fields.

Automated categorization when used 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 catalog taxonomy changes. Automated categorization is useful for Search Engines as well. 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. This helps users resolving the ambiguity that arises from a not enough specific query terms.

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. A relatively moderate size of document sets could 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, there are lots of synonyms and it is considered good writing style to not repeated 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, new documents keep coming in at regular intervals and new categories are added, old categories and documents 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 has been one of the faster algorithms for multiclass text classification (Ghani et al. 2001[1]) 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.

We propose to 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 proposed 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 more than a factor of 100. We propose to explore the use of ECOC to not only increase performance but at no or little additional cost. In previous work, we have shown that ECOC can reduce the error on a 105 class real world classification task by 30% at no added computational cost. In this research we plan to test this on more datasets as well as develop variations of ECOC that specifically focus on efficient training and result in increased performance for a fixed code length.

Our proposed 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 propose to perform this are:

- Specifically designing codes automatically or semi-automatically that maximize accuracy while minimizing code length.
- Assigning codewords to classes according to their confusability - classes that are highly confusable get codewords that are further apart
- 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 the training set with unlabeled data

We will discuss each of these approaches in detail in Section 4.

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].

[1] In a recent study, Ghani et al. (2001) showed that naïve Bayes was the fastest algorithm for classifying a collection of hypertext documents compared to kNN and FOIL

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 have given empirical evidence for 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. Approach

In all of the work previously described, no attempt has been made to specifically develop short codes for ECOC and to develop a framework where unlabeled data can be used for classifying documents a taxonomy with a large number of categories. The focus of this proposed 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 designing 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.

3.1 Overview of ECOC

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 thatwe 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

3.2 Preliminary Experiments

In previous work (Ghani 2000), we have shown that ECOC can be applied successfully to a real-world classification task with over a hundred categories (Industry Sector Dataset). We use Naïve Bayes Classifier which has been shown to perform well for several text classification tasks.

3.2.1 NAÏVE BAYES CLASSIFIER

Naive Bayes is a simple but effective text classification algorithm [McCallum & Nigam, Lewis]. 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 in a similar fashion, counting over documents instead of words. 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:

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

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

3.2.2 Experimental Results

We used standard 63 bit BCH codes for a 105 class problem and observed a 66% decreases in error when compared to using Naive Bayes by itself with one model per class. We also observed 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 and as the codes get larger, the accuracies start leveling off. ECOC gives a higher increase over Naïve Bayes especially when the number of training examples is small and this observation can be used to utilize ECOC in domains where training data is sparse. We also experimented with several different kinds of codes, namely random codes, code suing algebraic coding theory and hand constructed codes specific to the data and classes being classified and found that the BCH Codes (constructed using coding theory) outperformed the others.

3.3 Detailed Technical Approach

Apart from applying ECOC to text classification tasks with a large number of categories, we propose to investigate several other features of ECOC. Though the domain will be text classification, these results should generalize to any classification task. The following are detailed descriptions of the approaches we plan to explore.

3.3.1 Designing Short and effective codes

We plan to start from a code that has good error-correcting properties and then modify it through searching in the space of codewords so as to minimize the length of the codewords while maximizing the accuracy of the individual bit classifiers. Several hill climbing search algorithms can be used for this purpose. One particular method for constructing such codes would be to define an objective function which consists of two terms added together (one for the length of the code and the second for the error of the classifier) and try to minimize the objective function through a search algorithm.

3.3.2 Assignment of codewords to categories

To our knowledge, all of the previous work in ECOC has involved randomly assigning codewords to categories. It is possible that a more intelligent method of assignment could lead to better performance. One way we propose to do that is to learn the multiclass problem using a standard classifier 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 each pair of classes are confused by the classifier. 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. 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. In preliminary work, 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).

3.3.3 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 regression and or neural networks 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. To our knowledge, no previous work has been done to explore the learning of the decoding function for ECOC.

3.3.4 Using labeled and unlabeled data

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. Since, for most practical applications, labeling the data is most time-consuming process, an algorithm that can reduce the amount of labeled data needed is very beneficial. ECOC has already been shown to work extremely well even when labeled data is not available in large quantities (Ghani 2000, Dietterich & Bakiri 95). Unlabeled data is relatively easy to obtain in the case of text classification tasks using a simple web crawler. However, the previous algorithms (EM and Co-Training) are not specifically designed to handle a large number of categories. In preliminary experiments, we applied both of these algorithms to a real-world classification task consisting of over 60 categories and found that they failed to take advantage of unlabeled data. In fact, adding unlabeled data resulted in hurting the classifiers that we build using the small labeled training set. We propose to develop new algorithms that can take advantage of the co-training and EM framework and combine it with the efficiency that ECOC provides. How do we add unlabeled data in the ECOC framework? One algorithm that we plan to refine that has resulted in mixed results in our preliminary experimental work was to decompose the multiclass problem using ECOC and learn the individual binary problems using Co-Training. Co-Training has previously been shown to work well for binary problems but fails for a large number of categories. This algorithm would utilize the good performance of co-training on binary problems and the efficiency of ECOC for multiclass problems. We also plan to experiments with hybrids of EM and Co-Training that are described in previous work (Nigam & Ghani 2000). One potential drawback of such an approach or rather any approach using co-training type algorithms is the need for redundant and independent feature sets. In previous work, 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 feature set into two redundantly sufficient feature sets. This would extend the applicability of our proposed approach to regular datasets.

4. Evaluation Strategy

The approaches described above will be developed and evaluated using a variety of benchmark collections of practical use and importance that have a large number of categories including:

- Industry Sector Dataset (105 classes) used by McCallum et al (1998) and Ghani (2000)
- Hoovers-WebTC collection (255 classes) used by Ghani et al (2001)
- Job Descriptions Dataset (65 classes) obtained from WhizBang labs and used in preliminary experiments.

We plan to compare our results to published ones for the top performing text classifiers such as Naïve Bayes, kNN and SVM. We will perform systematic evaluations that will characterize the scalability of the basic algorithms in terms of the number of classes, number of features, and number of training examples. It is our goal to integrate the results of this work into Rainbow, a publicly available text classification package, to facilitate its dissemination into the scientific community at large.

5. Work Plan

The infrastructure for ECOC and the underlying classifier to be used (Naïve Bayes) is already in place and so is the evaluation procedure. Some time will be spent acquiring datasets but most of the datasets are already collected and formatted accordingly. The rest of the work plan is as follows:

- Work on designing efficient codes – 2 weeks
- Work on codeword assignment problem - 1.5 weeks
- Work on Learning the Decoding Function - 2 weeks
- Work on incorporating unlabeled data in to the ECOC framework - 2 weeks
- Project/Paper writeup – 1.5 week

6. Summary

We proposed novel improvements to Error-Correcting Output Coding method for the task of efficiently text classification with a large number of categories. If successful, these improvements will not only result in algorithms that are more efficient than current ones but will also result in better classification accuracy. Further more, the proposed research will give us more insight into both ECOC and the problem of text classification when applied to a domain with a large number of categories. If successful, it will make 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 *Submitted to 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. To appear in *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.