Text Classification by Bootstrapping with Keywords, EM and Shrinkage

Andrew McCallum[‡†]
mccallum@justresearch.com

[‡]Just Research
4616 Henry Street
Pittsburgh, PA 15213

Kamal Nigam[†]
knigam@cs.cmu.edu

[†]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

When applying text classification to complex tasks, it is tedious and expensive to hand-label the large amounts of training data necessary for good performance. This paper presents an alternative approach to text classification that requires no labeled documents; instead, it uses a small set of keywords per class, a class hierarchy and a large quantity of easily-obtained unlabeled documents. The keywords are used to assign approximate labels to the unlabeled documents by term-matching. These preliminary labels become the starting point for a bootstrapping process that learns a naive Bayes classifier using Expectation-Maximization and hierarchical shrinkage. When classifying a complex data set of computer science research papers into a 70-leaf topic hierarchy, the keywords alone provide 45% accuracy. The classifier learned by bootstrapping reaches 66% accuracy, a level close to human agreement.

1 Introduction

When provided with enough labeled training examples, a variety of text classification algorithms can learn reasonably accurate classifiers (Lewis, 1998; Joachims, 1998; Yang, 1999; Cohen and Singer, 1996). However, when applied to complex domains with many classes, these algorithms often require extremely large training sets to provide useful classification accuracy. Creating these sets of labeled data is tedious and expensive, since typically they must be labeled by a person. This leads us to consider learning algorithms that do not require such large amounts of labeled data.

While labeled data is difficult to obtain, *unlabeled* data is readily available and plentiful. Castelli and Cover (1996) show in a theoretical framework that unlabeled data can indeed be used to improve classification, although it is exponentially less valuable than labeled data. Fortunately, unlabeled data can often be obtained by completely automated methods. Consider the problem of classifying news articles: a short Perl script and a night of automated Internet downloads can fill a hard disk with unlabeled examples of news articles. In contrast, it might take several days of human effort and tedium to label even one thousand of these.

In previous work (Nigam et al., 1999) it has been shown that with just a small number of labeled documents, text classification error can be reduced by up to 30% when the labeled documents are augmented with a large collection of unlabeled documents.

This paper considers the task of learning text classifiers with *no* labeled documents at all. Knowledge about the classes of interest is provided in the form of a few keywords per class and a class hierarchy. Keywords are typically generated more quickly and easily than even a small number of labeled documents. Many classification problems naturally come with hierarchically-organized classes. Our algorithm proceeds by using the keywords to generate preliminary labels for some documents by term-matching. Then these labels, the hierarchy, and all the unlabeled documents become the input to a bootstrapping algorithm that produces a naive Bayes classifier.

The bootstrapping algorithm used in this paper combines hierarchical shrinkage and Expectation-Maximization (EM) with unlabeled data. EM is an iterative algorithm for maximum likelihood estimation in parametric estimation problems with missing data. In our scenario, the class labels of the documents are treated as missing data. Here, EM works by first training a classifier with only the documents

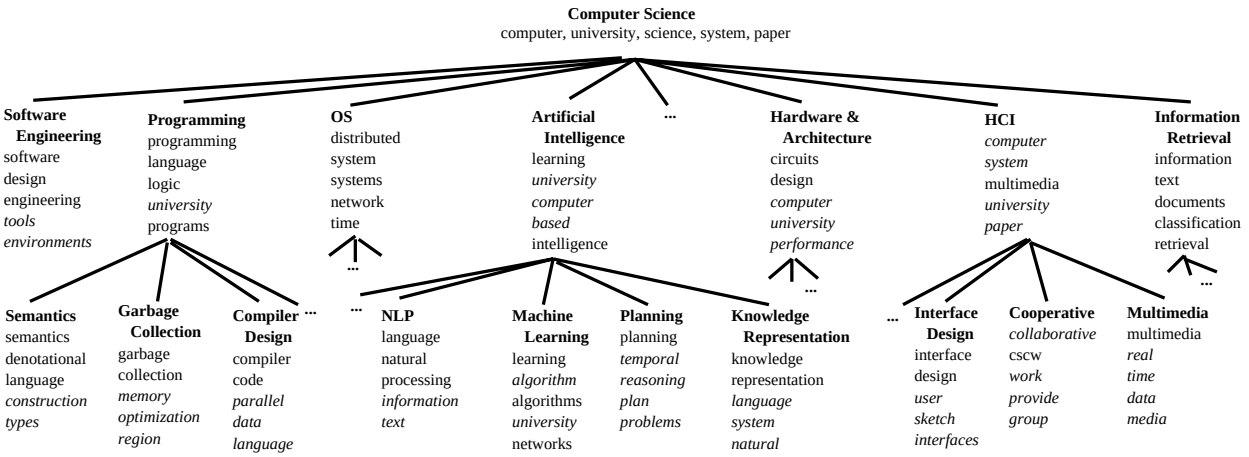

Figure 1: A subset of Cora's topic hierarchy. Each node contains its title, and the five most probable words, as calculated by naive Bayes and shrinkage with vertical word redistribution (Hofmann and Puzicha, 1998). Words among the initial keywords for that class are indicated in plain font; others are in italics.

preliminarily-labeled by the keywords, and then uses the classifier to re-assign probabilistically-weighted class labels to all the documents by calculating the expectation of the missing class labels. It then trains a new classifier using all the documents and iterates.

We further improve classification by incorporating *shrinkage*, a statistical technique for improving parameter estimation in the face of sparse data. When classes are provided in a hierarchical relationship, shrinkage is used to estimate new parameters by using a weighted average of the specific (but unreliable) local class estimates and the more general (but also more reliable) ancestors of the class in the hierarchy. The optimal weights in the average are calculated by an EM process that runs simultaneously with the EM that is re-estimating the class labels.

Experimental evaluation of this bootstrapping approach is performed on a data set of thirty-thousand computer science research papers. A 70-leaf hierarchy of computer science and a few keywords for each class are provided as input. Keyword matching alone provides 45% accuracy. Our bootstrapping algorithm uses this as input and outputs a naive Bayes text classifier that achieves 66% accuracy. Interestingly, this accuracy approaches estimated human agreement levels of 72%.

The experimental domain in this paper originates as part of the *Ra* research project, an effort to build domain-specific search engines on the Web with machine learning techniques. Our demonstration system, Cora, is a search engine over computer science research papers (McCallum et al., 1999). The bootstrapping classification algorithm described in this paper is used in Cora to place research papers into a Yahoo-like hierarchy specific to computer science. The search engine, including this hierarchy, is publicly available at *www.cora.justresearch.com*.

2 Generating Preliminary Labels with Keywords

The first step in the bootstrapping process is to use the keywords to generate preliminary labels for as many of the unlabeled documents as possible. Each class is given just a few keywords. Figure 1 shows examples of the number and type of keywords given in our experimental domain—the human-provided keywords are shown in the nodes in non-italic font.

In this paper, we generate preliminary labels from the keywords by term-matching in a rule-list fashion: for each document, we step through the keywords and place the document in the category of the first keyword that matches. Finding enough keywords to obtain broad coverage while simultaneously finding sufficiently specific keywords to obtain high accuracy is very difficult; it requires intimate knowledge of the data and a lot of trial and error.

As a result, classification by keyword matching is both an inaccurate and incomplete. Keywords tend to provide high-precision and low-recall; this brittleness will leave many documents unlabeled. Some documents will match keywords from the wrong class. In general we expect the low recall of the keywords to be the dominating factor in overall error. In our experimental domain, for example, 59% of the unlabeled documents do not contain any keywords.

Another method of priming bootstrapping with keywords would be to take each set of keywords as a

labeled mini-document containing just a few words. This could then be used as input to any standard learning algorithm. Testing this, and other keyword labeling approaches, is an area of ongoing work.

3 The Bootstrapping Algorithm

The goal of the bootstrapping step is to generate a naive Bayes classifier from the inputs: the (inaccurate and incomplete) preliminary labels, the unlabeled data and the class hierarchy. One straightforward method would be to simply take the unlabeled documents with preliminary labels, and treat this as labeled data in a standard supervised setting. This approach provides only minimal benefit for three reasons: (1) the labels are rather noisy, (2) the sample of preliminarily-labeled documents is skewed from the regular document distribution (*i.e.* it includes only documents containing keywords), and (3) data are sparse in comparison to the size of the feature space. Adding the remaining unlabeled data and running EM helps counter the first and second of these reasons. Adding hierarchical shrinkage to naive Bayes helps counter the first and third of these reasons. We begin a detailed description of our bootstrapping algorithm with a short overview of standard naive Bayes text classification, then proceed by adding EM to incorporate the unlabeled data, and conclude by explaining hierarchical shrinkage. An outline of the entire algorithm is presented in Table 1.

- **Inputs:** A collection $\mathcal{D}$ of unlabeled documents, a class hierarchy, and a few keywords for each class.
- Generate preliminary labels for as many of the unlabeled documents as possible by term-matching with the keywords in a rule-list fashion.
- Initialize all the λ_j's to be uniform along each path from a leaf class to the root of the class hierarchy.
- Iterate the EM algorithm:
  - **(M-step)** Build the maximum likelihood multinomial at each node in the hierarchy given the class probability estimates for each document (Equations 1 and 2). Normalize all the λ_j's along each path from a leaf class to the root of the class hierarchy so that they sum to 1.
  - **(E-step)** Calculate the expectation of the class labels of each document using the classifier created in the M-step (Equation 3). Increment the new λ_j's by attributing each word of held-out data probabilistically to the ancestors of each class.
- **Output:** A naive Bayes classifier that takes an unlabeled document and predicts a class label.

Table 1: An outline of the bootstrapping algorithm described in Sections 2 and 3.

3.1 The naive Bayes framework

We build on the framework of multinomial naive Bayes text classification (Lewis, 1998; McCallum and Nigam, 1998). It is useful to think of naive Bayes as estimating the parameters of a probabilistic generative model for text documents. In this model, first the class of the document is selected. The words of the document are then generated based on the parameters for the class-specific multinomial (*i.e.* unigram model). Thus, the classifier parameters consist of the class prior probabilities and the class-conditioned word probabilities. For formally, each class, c_j, has a document frequency relative to all other classes, written $\mathrm{P}(c_j)$. For every word w_t in the vocabulary V, $\mathrm{P}(w_t|c_j)$ indicates the frequency that the classifier expects word w_t to occur in documents in class c_j.

In the standard supervised setting, learning of the parameters is accomplished using a set of labeled training documents, $\mathcal{D}$. To estimate the word probability parameters, $\mathrm{P}(w_t|c_j)$, we count the frequency with which word w_t occurs among all word occurrences for documents in class c_j. We supplement this with Laplace smoothing that primes each estimate with a count of one to avoid probabilities of zero. Let $N(w_t, d_i)$ be the count of the number of times word w_t occurs in document d_i, and define $\mathrm{P}(c_j|d_i) \in \{0, 1\}$, as given by the document's class label. Then, the estimate of the probability of word w_t in class c_j is:

$$\mathrm{P}(w_t|c_j) = \frac{1 + \sum_{d_i \in \mathcal{D}} N(w_t, d_i)\mathrm{P}(c_j|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{d_i \in \mathcal{D}} N(w_s, d_i)\mathrm{P}(c_j|d_i)}. \quad (1)$$

The class prior probability parameters are set in the same way, where $|\mathcal{C}|$ indicates the number of classes:

$$\mathrm{P}(c_j) = \frac{1 + \sum_{d_i \in \mathcal{D}} \mathrm{P}(c_j|d_i)}{|\mathcal{C}| + |\mathcal{D}|}. \quad (2)$$

Given an unlabeled document and a classifier, we determine the probability that the document belongs in class c_j using Bayes' rule and the naive Bayes assumption—that the words in a document occur independently of each other given the class. If we denote $w_{d_{i,k}}$ to be the kth word in document d_i, then classification becomes:

$$\begin{aligned} \mathrm{P}(c_j|d_i) &\propto \mathrm{P}(c_j)\mathrm{P}(d_i|c_j) \\ &\propto \mathrm{P}(c_j)\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{i,k}}|c_j). \end{aligned} \quad (3)$$

Empirically, when given a large number of training documents, naive Bayes does a good job of classifying text documents (Lewis, 1998). More complete presentations of naive Bayes for text classification are provided by Mitchell (1997) and McCallum and Nigam (1998).

3.2 Adding unlabeled data with EM

In the standard supervised setting, each document comes with a label. In our bootstrapping scenario, the preliminary keyword labels are both incomplete and inaccurate—the keyword matching leaves many many documents unlabeled, and labels some incorrectly. In order to use the entire data set in a naive Bayes classifier, we use the Expectation-Maximization (EM) algorithm to generate probabilistically-weighted class labels for all the documents. This results in classifier parameters that are more likely given all the data.

EM is a class of iterative algorithms for maximum likelihood or maximum a posteriori parameter estimation in problems with incomplete data (Dempster et al., 1977). Given a model of data generation, and data with some missing values, EM iteratively uses the current model to estimate the missing values, and then uses the missing value estimates to improve the model. Using all the available data, EM will locally maximize the likelihood of the parameters and give estimates for the missing values. In our scenario, the class labels of the unlabeled data are the missing values.

In implementation, EM is an iterative two-step process. Initially, the parameter estimates are set in the standard naive Bayes way from just the preliminarily labeled documents. Then we iterate the E- and M-steps. The E-step calculates probabilistically-weighted class labels, $\mathrm{P}(c_j|d_i)$, for every document using the classifier and Equation 3. The M-step estimates new classifier parameters using all the documents, by Equations 1 and 2, where $\mathrm{P}(c_j|d_i)$ is now continuous, as given by the E-step. We iterate the E- and M-steps until the classifier converges. The initialization step from the preliminary labels identifies each mixture component with a class and seeds EM so that the local maxima that it finds correspond well to class definitions.

In previous work (Nigam et al., 1999), we have shown this technique significantly increases text classification accuracy when given limited amounts of labeled data and large amounts of unlabeled data. The expectation here is that EM will both correct and complete the labels for the entire data set.

3.3 Improving sparse data estimates with shrinkage

Even when provided with a large pool of documents, naive Bayes parameter estimation during bootstrapping will suffer from sparse data because naive Bayes has so many parameters to estimate ($|V||C| + |C|$). Using the provided class hierarchy, we can integrate the statistical technique *shrinkage* into the bootstrapping algorithm to help alleviate the sparse data problem.

Consider trying to estimate the probability of the word "intelligence" in the class NLP. This word should clearly have non-negligible probability there; however, with limited training data we may be unlucky, and the observed frequency of "intelligence" in NLP may be very far from its true expected value. One level up the hierarchy, however, the Artificial Intelligence class contains many more documents (the union of all the children). There, the probability of the word "intelligence" can be more reliably estimated.

Shrinkage calculates new word probability estimates for each leaf class by a *weighted average* of the estimates on the path from the leaf to the root. The technique balances a trade-off between specificity and reliability. Estimates in the leaf are most specific but unreliable; further up the hierarchy estimates are more reliable but unspecific. We can calculate mixture weights for the averaging that are guaranteed to maximize the likelihood of held-out data with the EM algorithm.

One can think of hierarchical shrinkage as a generative model that is slightly augmented from the one described in Section 3.1. As before, a class (leaf) is selected first. Then, for each word position in the document, an ancestor of the class (including itself) is selected according to the shrinkage weights. Then, the word itself is chosen based on the multinomial word distribution of that ancestor. If each word in the training data were labeled with which ancestor was responsible for generating it, then estimating the mixture weights would be a simple matter of maximum likelihood estimation from the ancestor emission counts. But these ancestor labels are not provided in the training data, and hence we use EM to fill in these missing values. We use the term *vertical EM* to refer to this process that calculates ancestor mixture weights; we use the term *horizontal EM* to refer to the process of filling in the missing

class (leaf) labels on the unlabeled documents. Both vertical and horizontal EM run concurrently, with interleaved E- and M-steps.

More formally, let $\{\mathrm{P}^1(w_t|c_j), \ldots, \mathrm{P}^k(w_t|c_j)\}$ be word probability estimates, where $\mathrm{P}^1(w_t|c_j)$ is the maximum likelihood estimate using training data just in the leaf, $\mathrm{P}^2(w_t|c_j)$ is the maximum likelihood estimate in the parent using the training data from the union of the parent's children, $\mathrm{P}^{k-1}(w_t|c_j)$ is the estimate at the root using all the training data, and $\mathrm{P}^k(w_t|c_j)$ is the uniform estimate ($\mathrm{P}^k(w_t|c_j) = 1/|V|$). The interpolation weights among c_j's "ancestors" (which we define to include c_j itself) are written $\{\lambda_j^1, \lambda_j^2, \ldots, \lambda_j^k\}$, where $\sum_{a=1}^{k} \lambda_j^a = 1$. The new word probability estimate based on shrinkage, denoted $\check{\mathrm{P}}(w_t|c_j)$, is then

$$\check{\mathrm{P}}(w_t|c_j) = \lambda_j^1 \mathrm{P}^1(w_t|c_j) + \ldots + \lambda_j^k \mathrm{P}^k(w_t|c_j). \quad (4)$$

The λ_j vectors are calculated by the iterations of EM. In the E-step we calculate for each class c_j and each word of unlabeled held out data, $\mathcal{H}$, the probability that the word was generated by the ith ancestor. In the M-step, we normalize the sum of these expectations to obtain new mixture weights λ_j. Without the use of held out data, all the mixture weight would concentrate in the leaves.

Specifically, we begin by initializing the λ mixture weights for each leaf to a uniform distribution. Let $\beta_j^a(d_{i,k})$ denote the probability that the ath ancestor of c_j was used to generate word occurrence $d_{i,k}$. The E-step consists of estimating the β's:

$$\beta_j^a(d_{i,k}) = \frac{\lambda_j^a \mathrm{P}^a(w_{d_{i,k}}|c_j)}{\sum_m \lambda_j^m \mathrm{P}^m(w_{d_{i,k}}|c_j)}. \quad (5)$$

In the M-step, we derive new and guaranteed improved weights, λ, by summing and normalizing the β's:

$$\lambda_j^a = \frac{\sum_{d_{i,k} \in \mathcal{H}} \beta_j^a(d_{i,k}) \mathrm{P}(c_j|d_i)}{\sum_{d_{i,k} \in \mathcal{H}} \mathrm{P}(c_j|d_i)}. \quad (6)$$

The E- and M-steps iterate until the λ's converge. These weights are then used to calculate new shrinkage-based word probability estimates, as in Equation 4. Classification of new test documents is performed just as before (Equation 3), where the Laplace estimates of the word probability estimates are replaced by shrinkage-based estimates.

A more complete description of hierarchical shrinkage for text classification is presented by McCallum et al. (1998).

4 Related Work

Other research efforts in text learning have also used bootstrapping approaches. The co-training algorithm (Blum and Mitchell, 1998) for classification works in cases where the feature space is separable into naturally redundant and independent parts. For example, web pages can be thought of as the text on the web page, and the collection of text in hyperlink anchors to that page.

A recent paper by Riloff and Jones (1999) bootstraps a dictionary of locations from just a small set of known locations. Here, their mutual bootstrap algorithm works by iteratively identifying syntactic constructs indicative of known locations, and identifying new locations using these indicative constructs.

The preliminary labeling by keyword matching used in this paper is similar to the seed collocations used by Yarowsky (1995). There, in a word sense disambiguation task, a bootstrapping algorithm is seeded with some examples of common collocations with the particular sense of some word (*e.g.* the seed "life" for the biological sense of "plant").

5 Experimental Results

In this section, we provide empirical evidence that bootstrapping a text classifier from unlabeled data can produce a high-accuracy text classifier. As a test domain, we use computer science research papers. We have created a 70-leaf hierarchy of computer science topics, part of which is shown in Figure 1. Creating the hierarchy took about 60 minutes, during which we examined conference proceedings, and explored computer science sites on the Web. Selecting a few keywords associated with each node took about 90 minutes. A test set was created by expert hand-labeling of a random sample of 625 research papers from the 30,682 papers in the Cora archive at the time we began these experiments. Of these, 225 (about one-third) did not fit into any category, and were discarded—resulting in a 400 document test set. Labeling these 400 documents took about six hours. Some of these papers were outside the area of computer science (*e.g.* astrophysics papers), but most of these were papers that with a more complete hierarchy would be considered computer science papers. The class frequencies of the data are not too skewed; on the test set, the most populous class accounted for only 7% of the documents.

Each research paper is represented as the words of the title, author, institution, references, and abstract. A detailed description of how these segments are automatically extracted is provided elsewhere (McCallum et al., 1999; Seymore et al., 1999).

Method	# Lab	# P-Lab	# Unlab	Acc
Keyword	—	—	—	45%
NB	100	—	—	30%
NB	399	—	—	47%
NB+EM+S	—	12,657	18,025	66%
NB	—	12,657	—	47%
NB+S	—	12,657	—	63%
Human	—	—	—	72%

Table 2: Classification results with different techniques: keyword matching, human agreement, naive Bayes (NB), and naive Bayes combined with hierarchical shrinkage (S), and EM. The classification accuracy (Acc), and the number of labeled (Lab), keyword-matched preliminarily-labeled (P-Lab), and unlabeled (Unlab) documents used by each method are shown.

Words occurring in fewer than five documents and words on a standard stoplist were discarded. No stemming was used. Bootstrapping was performed using the algorithm outlined in Table 1.

Table 2 shows classification results with different classification techniques used. The rule-list classifier based on the keywords alone provides 45%. (The 43% of documents in the test set containing no keywords cannot be assigned a class by the rule-list classifier, and are counted as incorrect.) As an interesting time comparison, about 100 documents could have been labeled in the time it took to generate the keyword lists. Naive Bayes accuracy with 100 labeled documents is only 30%. With 399 labeled documents (using our test set in a leave-one-out-fashion), naive Bayes reaches 47%. When running the bootstrapping algorithm, 12,657 documents are given preliminary labels by keyword matching. EM and shrinkage incorporate the remaining 18,025 documents, "fix" the preliminary labels and leverage the hierarchy; the resulting accuracy is 66%. As an interesting comparison, agreement on the test set between two human experts was 72%.

A few further experiments reveal some of the inner-workings of bootstrapping. If we build a naive Bayes classifier in the standard supervised way from the 12,657 preliminarily labeled documents the classifier gets 47% accuracy. This corresponds to the performance for the first iteration of bootstrapping. Note that this matches the accuracy of traditional naive Bayes with 399 labeled training documents, but that it requires less than a quarter the human labeling effort. If we run bootstrapping without the 18,025 documents left unlabeled by keyword matching, accuracy reaches 63%. This indicates that shrinkage and EM on the preliminarily labeled documents is providing substantially more benefit than the remaining unlabeled documents.

One explanation for the small impact of the 18,025 documents left unlabeled by keyword matching is that many of these do not fall naturally into the hierarchy. Remember that about one-third of the 30,000 documents fall outside the hierarchy. Most of these will not be given preliminary labels by keyword matching. The presence of these outlier documents skews EM parameter estimation. A more inclusive computer science hierarchy would allow the unlabeled documents to benefit classification more.

However, even without a complete hierarchy, we could use these documents if we could identify these outliers. Some techniques for robust estimation with EM are discussed by McLachlan and Basford (1988). One specific technique for these text hierarchies is to add extra leaf nodes containing uniform word distributions to each interior node of the hierarchy in order to capture documents not belonging in any of the predefined topic leaves. This should allow EM to perform well even when a large percentage of the documents do not fall into the given classification hierarchy. A similar approach is also planned for research in topic detection and tracking (TDT) (Baker et al., 1999). Experimentation with these techniques is an area of ongoing research.

6 Conclusions and Future Work

This paper has considered building a text classifier without labeled training documents. In its place, our bootstrapping algorithm uses a large pool of unlabeled documents and class-specific knowledge in the form of a few keywords per class and a class hierarchy. The bootstrapping algorithm combines Expectation-Maximization and hierarchical shrinkage to correct and complete preliminary labeling provided by keyword matching. Experimental results show that accuracies close to human agreement can be obtained by the bootstrapping algorithm.

In future work we plan to refine our probabilistic model to allow for documents to be placed in interior hierarchy nodes, documents to have multiple class assignments, and classes to be modeled with multiple mixture components. We are also investigating principled methods of re-weighting the word features for "semi-supervised" clustering that will provide better discriminative training with unlabeled data.

Acknowledgements

Kamal Nigam was supported in part by the Darpa HPKB program under contract F30602-97-1-0215.

References

D. Baker, T. Hofmann, A. McCallum, and Y. Yang. 1999. A hierarchical probabilistic model for novelty detection in text. Technical report, Just Research. http://www.cs.cmu.edu/∼mccallum.

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

V. Castelli and T. M. Cover. 1996. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2101–2117.

W. Cohen and Y. Singer. 1996. Context-sensitive learning methods for text categorization. In *SIGIR '96*.

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

T. Hofmann and J. Puzicha. 1998. Statistical models for co-occurrence data. Technical Report AI Memo 1625, AI Lab, MIT.

T. Joachims. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *ECML-98*.

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

A. McCallum and K. Nigam. 1998. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*. Tech. rep. WS-98-05, AAAI Press. http://www.cs.cmu.edu/∼mccallum.

A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. 1998. Improving text clasification by shrinkage in a hierarchy of classes. In *ICML-98*.

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 1999. Using machine learning techniques to build domain-specific search engines. In *IJCAI-99*. To appear.

G.J. McLachlan and K.E. Basford. 1988. *Mixture Models*. Marcel Dekker, New York.

T. M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, New York.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 1999. Text classification from labeled and unlabeled documents using EM. *Machine Learning*. To appear.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction using multi-level bootstrapping. In *AAAI-99*. To appear.

K. Seymore, A. McCallum, and R. Rosenfeld. 1999. Learning hidden Markov model structure for information extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*. To appear.

Y. Yang. 1999. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*. To appear.

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