Multi-Label, Multiclass Document Classification with a Mixture Model Trained by EM

Andrew Kachites McCallum
mccallum@justresearch.com
Just Research
4616 Henry Street
Pittsburgh, PA 15213

Abstract

In many important document classification tasks, documents may each be associated with multiple class labels. This paper describes a Bayesian classification approach in which the multiple class "topics" that comprise a document are represented by a mixture model. While the labeled training data indicates which classes were responsible for generating a document, it does not indicate which class was responsible for generating each word. Thus we use Expectation-Maximization to fill in this missing value, learning both the distribution over mixture weights and the word distribution in each class's mixture component. We present preliminary results with a subset of the Reuters-21578 data set showing that our new model outperforms a traditional binary classifier, sometimes reducing error by half.

Introduction

Text classification is the problem of assigning a text document into one or more topic categories or classes. In multiclass document classification, as distinguished from binary document classification, there are more than two classes. In multi-label classification each document may have more than one class label. For example, given classes North America, South America, Europe, Asia and Australia, a news article about U.S. planes bombing in Bosnia may be labeled with both the North America and Europe classes. Multi-label classification is an important problem because in the many real-world text classification problems topic labels are not mutually exclusive. This is reflected, for example, in the Reuters-21578 and OHSU-Med data sets.[1]

This paper describes a Bayesian approach to multiclass, multi-label document classification. We define a probabilistic generative model that represents the multi-label nature of a document by indicating that the words in a document are produced by a mixture of word distributions, one for each topic. The generative process begins by selecting the set of classes (instead of a single class) that will be the labels for this document; then a set of mixture weights is chosen for those classes; finally, each word in the document is generated by first selecting a class according to these mixture weights, then letting that class generate a single word.

Parameters of the model are learned by maximum a posteriori estimation from labeled training data. The labels indicate which classes were involved in generating the document, but do not indicate which class generated which individual words. Thus we use Expectation-Maximization (EM) (Dempster, Laird, & Rubin 1977) to fill in this missing value, avoiding over-fitting by performing the E-step in leave-one-document-out fashion. EM sets both the mixture weights and the word distributions in each class. We also add an extra class to which all documents belong: the "English" class. This naturally captures background language in common to all documents—in essence automatically extracting the stop list.

There are two common alternatives to this approach. The first is to build a binary classifier for each class (*e.g.* (Lewis *et al.* 1996; Yang 1998; Joachims 1998; Liere & Tadepalli 1997; Nigam *et al.* 1999)). This involves building one model of the documents having the given class label, and another model of all the documents not having the label. At test time, if the probability or score of the first model is greater than the second (or above some non-0.5 threshold), then the given label is included among those assigned to this document.

The second is to build separate models of each class (with no paired negative-class model) and also associate a threshold with each class. At test time, the document is assigned those class labels from models that resulted in above-threshold probabilities or scores.

A third, uncommon approach would consider each of the $2^{|\mathcal{C}|}$ class vectors as a separate "class", and use a traditional multiclass classifier. Training data would be so fragmented in this approach that parameter estimation would be almost impossible unless the number of classes were very small or the amount of training data were amazingly large.

The key features of the approach in this paper are four-fold. First, we conjecture that our mixture model can represent more expressive decision boundaries than can the binary classifier. In other words, the binary classifier's large and multi-faceted negative class is not well-represented by a single word distribution model.

[1] See http://www.research.att.com/ lewis/reuters21578.html and ftp://medir.ohsu.edu/pub/ohsumed

This paper's mixture model does not attempt to do this; instead it represents each class individually without the need for a corresponding negative class model.

Second, note that binary classifiers tacitly assume that labels can be assigned independently. When one label provides information about another, the binary classifier fails to capture this. For example, two classes may have very similar word distributions (and hence be equally likely according to the binary classifiers), but only one of them may co-occur with a certain other class label. Our generative model correctly classifies such data because it represents the dependencies between class labels.

Third, there are no thresholds to be tuned or learned. Threshold learning is typically quite problematic because document length and other attributes affect the score, making it difficult to relate scores across different documents. Document length normalization heuristics only partially address this problem.

Finally, we have the advantages of a formal probabilistic approach, with a well-defined generative model. The formal foundations also make available future enhancements based on the large tool chest of powerful statistical parameter estimation techniques. For example, if the classes can be arranged in a directed acyclic graph or a hierarchy or related topics, then shrinkage could be used to more robustly estimate model parameters in the face of sparse training data (Carlin & Louis 1996; McCallum *et al.* 1998).

This paper presents preliminary experimental results on a subset of the Reuters-21578 data set. We find that the mixture model outperforms the approach based on a collection of binary classifiers, reducing classification error on almost all labels, reducing error by more than 50% on one label, and reducing overall error by one third. Further experiments are needed to fully evaluate the technique.

Multi-label Classification with a Mixture Model

The Generative Model

The generative model contains a set of classes, $\mathcal{C} = \{c_1, c_2, ...c_n\}$. Each document is associated with some subset of these classes, which can be thought of as a binary bit-vector, $\vec{c}$, with ones indicating the classes in the subset. The power set of all possible class subsets is written $\mathcal{C}^\star$. Associated with each individual class, c_j is a word distribution $\mathrm{P}(w_k|c_j)$, for all words in the vocabulary, $\mathcal{V} = \{w_1, w_2, ...w_m\}$. Given a set of classes, each document is generated by a mixture of these word distributions, with mixture weights $\vec{\lambda}$, where components of $\vec{\lambda}$ associated with classes not in $\vec{c}$ are forced to be zero. The mixture weights themselves are selected from a distribution over mixture weights written $\mathrm{P}(\vec{\lambda}|\vec{c})$.

Labeled training data consists of a set of documents $\mathcal{D} = \{d_1, d_2, ...d_N\}$. The vector of class labels associated with document d_i is written $\vec{l}_i$.

The generative process for a document is as follows. First select a set of classes from distribution $\mathrm{P}(\vec{c})$. Then select mixture weights among these classes according to $\mathrm{P}(\vec{\lambda}|\vec{c})$. Then start generating words in the document. For each word select a class according the mixture weights in $\vec{\lambda}$, (the components of which are written λ_c, which can be thought of as $\mathrm{P}(c|\vec{\lambda})$), then let that class generate a single word according to its multinomial word probability distribution $\mathrm{P}(w_k|c_j)$.

Thus the probability of a document is

$$\mathrm{P}(d) = \sum_{\vec{c}\in\mathcal{C}^\star} \int d\vec{\lambda}\, \mathrm{P}(\vec{\lambda}|\vec{c}) \prod_{w\in d} \sum_{c\in\mathcal{C}} \vec{\lambda}_c \mathrm{P}(w|c). \qquad (1)$$

Once this generative model is defined, multi-label Bayesian classification falls out naturally. Given a test document we wish to select the set of classes that is most probable: $\vec{c}^+ = \arg\max_{\vec{c}} \mathrm{P}(\vec{c}|d)$. To express $\mathrm{P}(\vec{c}|d)$ in terms of the generative model parameters, we begin by applying Bayes' rule:

$$\mathrm{P}(\vec{c}|d) = \frac{\mathrm{P}(\vec{c})\mathrm{P}(d|\vec{c})}{\mathrm{P}(d)}. \qquad (2)$$

The generative process for the document d is not defined in terms of $\vec{c}$ only, but requires the mixture weights. We introduce the mixture weights by a sum of total probability, (and also drop the $\mathrm{P}(d)$, which is just a normalization factor that makes the $\mathrm{P}(\vec{c}|d)$'s sum to one):

$$\mathrm{P}(\vec{c}|d) \propto \mathrm{P}(\vec{c}) \int d\vec{\lambda}\, \mathrm{P}(\vec{\lambda}|\vec{c})\mathrm{P}(d|\vec{\lambda}, \vec{c}). \qquad (3)$$

This is an integral over an infinite number of possible real-valued vectors $\vec{\lambda}$. Finding the closed-form solution to this integral is the subject of future work. For now, we approximate the integral using the single a priori most probable $\vec{\lambda}$.[2] (See Equation 9.)

Let $\vec{\lambda}^{(\vec{c})}$ be the single most likely mixture weights given the set of classes $\vec{c}$:

$$\vec{\lambda}^{(\vec{c})} \equiv \arg\max_{\lambda} \mathrm{P}(\lambda|\vec{c}), \qquad (4)$$

then we approximate Equation 3 by

$$\mathrm{P}(\vec{c}|d) \approx \mathrm{P}(\vec{c})\mathrm{P}(d|\vec{\lambda}^{(\vec{c})}, \vec{c}). \qquad (5)$$

[2]Using the single a priori most probable mixture weights for each class set is one solution, however early experimental evidence indicates that quite different mixture weights are most likely for different documents labeled with the same class set, thus indicating that this uni-modal approximation could be improved. An interesting alternative to the closed form solution is a non-parametric, "memory-based" representation for $\mathrm{P}(\vec{\lambda}|\vec{c})$: Instead of summing over all possible mixture weights $\vec{\lambda}$, sum over the most likely mixture weights associated with each of the training documents labeled with class set $\vec{c}$. This could be even better than a closed-form solution when the parametric form of of $\mathrm{P}(\vec{\lambda}|\vec{c})$ (such as Dirichlet) is a poor match for the data.

Assuming word independence, and then expanding the mixture model gives

$$\mathrm{P}(\vec{c}|d) \approx \mathrm{P}(\vec{c}) \prod_{w \in d} \sum_{c \in \mathcal{C}} \vec{\lambda}_c^{(\vec{c})} \mathrm{P}(w|c). \tag{6}$$

There remains the question how to represent the function $\mathrm{P}(\vec{c})$. In the current implementation the function is represented in completely unfactored form (*i.e.* with separate estimates for each vector $\vec{c}$). While this is an expressive form, there are many degrees of freedom, and smoothing of these estimates is important. See the next sub-section for more details.

Classification is performed by calculating the probability of each class set $\vec{c}$ and selecting the set that is most probable. In the current implementation the size of this set is limited to three, and thus the number of possible sets is not too large to evaluate exhaustively. When the set size is unrestricted, efficient yet accurate approximations will be necessary. Here are three possible methods of efficient approximation.

The first is to greedily add classes to the set one at a time. Begin restricting $\vec{c}$ to size one, and finding $\arg\max_{\vec{c}} \mathrm{P}(\vec{c}|d)$, then restrict $\vec{c}$ to size two in which one of the classes must be the class selected in the first step. Continue this process, until all classes have been added, and then select the combination that is most likely. It is believed that stopping the process before all classes are added and soon after $\mathrm{P}(\vec{c}|d)$ begins to drop will not reduce accuracy.

The second method of efficient classification with unrestricted class set size is to find the mixture weights λ that are most likely to have produced the document (described as the E-step in the next sub-section), and then evaluate $\mathrm{P}(\vec{c}|d)$ only for those classes with mixture weights $\vec{\lambda}^{(\vec{c})}$ that are similar to to λ, (measuring similarly, for example, by Kullback-Liebler divergence).

The third is related to both of the previous methods. Greedily add classes to the set one at a time. Begin by finding the mixture weights λ that are most likely to have produced the document (again, described as the E-step in the next sub-section), and initializing $\vec{c}$ to contain only the class with the highest mixture weight. Then find modified mixture weights among those classes not in the current $\vec{c}$, where the modified mixture weights indicate to what extent the class gives higher probability to words in the document than do the classes already in $\vec{c}$. Add to $\vec{c}$ the class with the highest mixture weight, and repeat. This process is quite similar to step-wise regression. Additional accuracy may be obtained by performing a limited amount of exploration around this greedy path.

Parameter Estimation

Now we turn to the subject of learning the parameters of the generative model from labeled training data. We determine parameters by maximum a posteriori estimation.

The class set prior distribution is set by maximum a posteriori estimation with a uniform prior:

$$\mathrm{P}(\vec{c}) = \frac{m + N(d, \vec{c})}{m|\mathcal{C}^\star| + |\mathcal{D}|}, \tag{7}$$

where $N(d, \vec{c})$ is the number of labeled training documents labeled with exactly the vector $\vec{c}$, $|\mathcal{C}^\star|$ is the size of the power set of the set of classes $\mathcal{C}$, $|\mathcal{D}|$ is the size of the set of all training documents, and m is a smoothing parameter determining the strength of the uniform prior. Setting $m = 1$ results in Laplace smoothing.

All other parameters cannot be estimated directly from training data since all aspects of the generative process are not provided by the given labels. Specifically, the labels on the training document indicate which classes were involved in generating each document, but not which classes were responsible for generating each individual word. Thus we turn to Expectation-Maximization (EM) (Dempster, Laird, & Rubin 1977) in order to fill in this missing value. In the E-step we estimate for each word in a training document which class was responsible for generating it. In the M-step we use these estimates to straightforwardly determine maximum a posteriori estimates of the most likely mixture weight for each class ($\vec{\lambda}^{(\vec{c})}$) and per-class word probability distributions ($\mathrm{P}(w|c)$).

The E-step uses the current model parameters to estimate the probability that word w in document d was generated by class c. We write this $\mathrm{P}(c|w)$, and calculate it by

$$\mathrm{P}(c|w \in d_i) = \frac{\vec{\lambda}_c^{(\vec{l}_i)} \mathrm{P}(w|c)}{\sum_{c'} \vec{\lambda}_{c'}^{(\vec{l}_i)} \mathrm{P}(w|c')}. \tag{8}$$

The M-step then re-estimates model parameters using the results of the last E-step. The mostly likely mixture weights for a class set are simply the proportions of the number of words generated by each class among those training documents labeled with the given class set (smoothed with a Laplace prior):

$$\vec{\lambda}_c^{(\vec{c})} = \frac{1 + \sum_{w \in d_i \in \vec{c}} \mathrm{P}(c|w \in d_i)}{|\mathcal{C}| + \sum_{c' \in \vec{c}} \sum_{w \in d_i \in \vec{c}} \mathrm{P}(c'|w \in d_i)}. \tag{9}$$

If there are no training documents for a particular set of classes, then we backoff to the average mixture weights for all sets of classes of the equivalent size. One could also imagine backoff schemes that took the identities of the classes into consideration.

The per-class word distributions are determined by the (probabilistically-weighted) counts of the number of times each word was generated by a class. We do not smooth with a Laplace prior here because of the mixture with the uniform word distribution as described in the next sub-section. Thus the probability the word w_i is generated by class c is

$$\mathrm{P}(w_i|c) = \frac{\sum_{w \in d \in \mathcal{D}} \mathrm{P}(c|w)}{Z_c}, \tag{10}$$

where Z_c is the normalization factor is necessary to make the probabilities for all words in class c sum to one, $\sum_{w \in \mathcal{V}} \mathrm{P}(w|c) = 1$.

Additional Features

To guard against over-fitting, the E-step above is performed in leave-one-out fashion. That is, when the estimating which classes generated the words in a particular training document, the influence of that training document is temporarily removed from the parameter estimates used to perform the E-step. Because the parameters are based on (probabilistically weighted) counts, this removal is quite easy. The leave-one-out E-step is also used in deleted interpolation for speech recognition (Jelinek & Mercer 1980) and in work on shrinkage for document classification (McCallum *et al.* 1998).

In addition to the classes in $\mathcal{C}$, we also add an extra class to which all documents belong. This can be thought of as the "English" class. Due to the leave-one-out E-step, this class gathers the words that are common to all classes—in essence automatically finding the task-specific stop list.

Furthermore, the uniform word distribution is also added as another mixture component. Mixture weights $\vec{\lambda}$ are learned for this mixture components, but obviously the word distribution is not, since it is fixed. In conjunction with the leave-one-out E-step, this serves to automatically determine the "optimal" amount of smoothing with uniform distribution.[3] For example, when calculating the E-step, a word that only occurs once in the training data will have zero probability in all mixtures except the uniform distribution since, in accordance with the leave-one-out E-step, the document in which that word occurred has been removed from the training data. The mixture weight λ for the uniform distribution will thus be increased, properly set to maximize the probability of this left-out document.

Experimental Results

The Reuters 21578 Distribution 1.0 was used to evaluate the performance of the multi-label mixture model in comparison with a binary classification approach. As in several other studies, only the ten most populous classes were used (Joachims 1998; Liere & Tadepalli 1997; Nigam *et al.* 1999), and results are evaluated with the ModApte test/train split. Additionally, since the current implementation is limited to class sets of size three, the five documents having more than three labels of out the ten were removed. In future work, this class set size restriction will be removed, and we will perform evaluations with other data sets.

The text of Reuters-21578 was tokenized by removing SGML tags, removing "REUTER" tags from the end of documents, gathering strings alphabetic characters, downcasing them and removing words on the SMART stoplist. Additionally all strings of digits were mapped to a single common token. No stemming was used.

[3]It is optimal in the sense that is maximizes the probability of all the training data in leave-one-out fashion.

grain	wheat	corn	ship	trade
$d^\star$	$d^\star$	$d^\star$	$d^\star$	trade
grain	wheat	corn	gulf	billion
tonnes	tonnes	tonnes	ships	$d^\star$
mln	mln	mln	strike	japan
agriculture	export	maize	port	deficit
year	agriculture	pct	shipping	year
usda	offer	usda	seamen	japanese
crop	department	acres	march	surplus
farmers	soviet	year	union	exports
farm	march	production	iran	countries

Table 1: The ten most probable words in multinomial word distributions of five topic classes from the ten used in the Reuters-21578 data set. The notation $d^\star$ refers to a string of digits.

The multi-label mixture model was trained with the ModApte training set. EM was run to convergence, which took ten iterations and about 4 minutes of wall clock time.

The performance of this model is compared with the traditional method consisting of a collection of binary classifiers, one for each class, as used in several other studies (Joachims 1998; Liere & Tadepalli 1997; Nigam *et al.* 1999). Each binary classifier is multinomial naive Bayes with Laplace smoothing (McCallum & Nigam 1998). Feature selection, when used, is performed by choosing the words that have highest mutual information with the class variable (Cover & Thomas 1991; Yang & Pederson 1997).

The results of EM estimation of the word probability distribution in each of the classes are quite interesting. As expected, words associated with the different class topics gather in the appropriate classes. Even classes that never occur as the sole label for a document, such as wheat, tend to gather those words that are in common among the wheat documents, excluding words that are explained by other topics. Some word distributions for several example classes are given in Table 1. Notice that some high-probability words still overlap among classes, such as "tonnes" in in grain, wheat and corn. This suggests that a hierarchical or directed acyclic graph structure among classes could capture some of this redundancy and perhaps help generalization; see the discussion of future work in the last section of this paper.

A comparison of classification accuracy is shown in Table 2. This paper's multi-label mixture model outperforms the traditional collection of binary classifiers for all classes except one (ship), sometimes reducing error by more than half (*e.g.* wheat). An unfair advantage is given to the traditional binary classifier in that we select the vocabulary size with which it does best on the test set. Running the multi-label mixture model involved no such tuning or feature selection—the full vocabulary was used. The bottom line shows classifica-

class	binary NB classifier accuracy (vocab size)	multi-label mixture accuracy
acq	0.9707 (23433)	0.9783
corn	0.9751 (20)	0.9834
crude	0.9751 (23433)	0.9807
earn	0.9718 (2000)	0.9823
grain	0.9627 (20)	0.9795
interest	0.9639 (23433)	0.9827
money	0.9536 (23433)	0.9689
ship	0.9867 (23433)	0.9815
trade	0.9524 (20)	0.9654
wheat	0.9618 (23433)	0.9811
full set	0.7569	0.8392

Table 2: Classification accuracy on each of the ten labels in the subset of Reuters-21578. The multi-label mixture model sometimes reduces error by half (wheat). The last line shows accuracy on the full multi-label task, where credit is given only for class sets that are an exact match with label set. Here the multi-label mixture model reduces error by one third.

tion accuracy on the full multi-label task, where credit is given only for class sets that are an exact match with label set. Here the multi-label mixture model reduces error by one third.

Related Work

Various related work has used mixture models for modeling text. The Nymble system from BBN uses an HMM with uniform transition probabilities to perform information extraction of "named entities" (Bikel *et al.* 1997). Similar models have been used for text segmentation and topic tracking (van Mulbregt *et al.* 1998; Yamron *et al.* 1998). Topic-based mixture models have also been used to improve speech recognition (Seymore & Rosenfeld 1997).

Perhaps the most related work is that of Imai *et al.* (1997), in which a mixture model is used for multi-label document classification. However, their work does not have a class-set conditional mixture distribution, does not use leave-one-out evaluation in the E-step, does not do "optimal smoothing" by mixing with the uniform distribution, and classifies test documents by searching the space of class sets using a method based on a heuristic threshold. Their model is applied to topic discrimination on broadcast news data.

Schapire and Singer's BoosTexter algorithm (Schapire & Singer 1999) is a collection of enhancements to AdaBoost (Freund & Schapire 1997) that enable its application to multiclass, multi-label document classification problems. Unlike traditional Boosting, the labels as well as the training instances are re-weighted. BoosTexter aims to predict all the correct labels by ranking them so that the correct labels receive the highest ranks.

The mixture model described in this paper is similar to the "aspect" clustering model described by Hofmann & Puzicha (1998). There also, a document is generated by a mixture of per-topic models. Instead of supervised classification, their model is used for unsupervised clustering, and has been shown to improve language modeling and information retrieval (Gildea & Hofmann 1999; Hofmann 1999).

Conclusions

This paper presents a Bayesian approach to multiclass, multi-label document classification. We formally define a generative model, explain how classification can be performed with this model and describe how EM is used to find maximum a posteriori parameter estimates.

The model provides significant improvement in classification accuracy on a subset of the Reuters-21578 data set. In addition to improved accuracy, the model is also more space-efficient than the traditional collection of binary classifiers, and avoids the problematic threshold-tuning associated with the alternative traditional method.

In future work we will perform further experiments with Reuters-21578 as well as with other data sets. We will also combine this model with one that takes advantage of a hierarchy of topics (McCallum *et al.* 1998), or a directed acyclic graph of topics. This model also lends itself to improving parameter estimates by incorporating unlabeled data, as in the work by Nigam *et al.* (1999). We will also explore more complex functions and priors for the $\mathrm{P}(\vec{c})$ and $\mathrm{P}(\vec{\lambda}|\vec{c})$ functions. The model may additionally be improved by using multiple mixture components for each topic, the word distribution of each learned in a completely unsupervised fashion.

Acknowledgements

Thanks to Thomas Minka and Doug Baker for providing many useful comments on an earlier draft. Thanks to Lyle Ungar for pointing out the similarity between the third greedy class-set search technique and step-wise regression.

References

Bikel, D. M.; Miller, S.; Schwartz, R.; and Weischedel, R. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, 194–201.

Carlin, B., and Louis, T. 1996. *Bayes and Empirical Bayes Methods for Data Analysis.* Chapman and Hall.

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory.* New York: John Wiley and Sons.

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

Freund, Y., and Schapire, R. E. 1997. A decision-theorectic generalization of on-line learning and an application t boosting. *Journal of Computer and System Sciences* 55(1):119–139.

Gildea, D., and Hofmann, T. 1999. Topic based language models using em. In *Proceedings of 6th European Conference On Speech Communication and Technology (Eurospeech'99)*.

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

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*.

Imai, T.; Schwartz, R.; Kubala, F.; and Nguyen, L. 1997. Improved topic discrimination of broadcast news using a model of multiple simultaneous topics. In *Proceedings of the IEEE ICASSP*, volume 2, 727–730.

Jelinek, F., and Mercer, R. 1980. Interpolated estimation of Markov source parameters from sparse data. In Gelsema, S., and Kanal, L. N., eds., *Pattern Recognition in Practice*, 381–402.

Joachims, T. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *Machine Learning: ECML-98, Tenth European Conference on Machine Lea rning*, 137–142.

Lewis, D. D.; Schapire, R. E.; Callan, J. P.; and Papka, R. 1996. Training algorithms for linear text classifiers. In Frei, H.-P.; Harman, D.; Schauble, P.; and Wilkinson, R., eds., *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 298–306. Hartung-Gorre Verlag.

Liere, R., and Tadepalli, P. 1997. Active learning with committees for text categorization. In *Proceedings of the Fourteenth National Conference on Artificial Int elligence (AAAI-97)*, 591–596.

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

McCallum, A.; Rosenfeld, R.; Mitchell, T.; and Ng, A. 1998. Improving text clasification by shrinkage in a hierarchy of classes. In *International Conference on Machine Learning (ICML)*, 359–367.

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

Schapire, R. E., and Singer, Y. 1999. Boostexter: A boosting-based system for text categorization. *Machine Learning Journal*. To appear.

Seymore, K., and Rosenfeld, R. 1997. Using story topics for language model adaptation. In *Eurospeech'97*.

van Mulbregt, P.; Carp, I.; Gillick, L.; and Yamron, J. 1998. Text segmentation and topic tracking on broadcast news via a hidden markov model approach. In *Proceedings of the International Conference on Spoken Language Processing*.

Yamron, J.; Carp, I.; Gillick, L.; Lowe, S.; and van Mulbregt, P. 1998. A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of the IEEE ICASSP*.

Yang, Y., and Pederson, J. 1997. Feature selection in statistical learning of text categorization. In *International Conference on Machine Learning (ICML)*.

Yang, Y. 1998. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*. To appear.