Inductive Learning Algorithms and Representations for Text Categorization

Susan Dumais
Microsoft Research
One Microsoft Way
Redmond, WA 98052
sdumais@microsoft.com

John Platt
Microsoft Research
One Microsoft Way
Redmond, WA 98052
jplatt@microsoft.com

David Heckerman
Microsoft Research
One Microsoft Way
Redmond, WA 98052
heckerma@microsoft.com

Mehran Sahami
Computer Science Department
Stanford University
Stanford, CA 94305-9010
sahami@cs.stanford.edu

1. ABSTRACT

***Text categorization* – the assignment of natural language texts to one or more predefined categories based on their content – is an important component in many information organization and management tasks. We compare the effectiveness of five different automatic learning algorithms for text categorization in terms of learning speed, real-time classification speed, and classification accuracy. We also examine training set size, and alternative document representations. Very accurate text classifiers can be learned automatically from training examples. Linear Support Vector Machines (SVMs) are particularly promising because they are very accurate, quick to train, and quick to evaluate.**

1.1 Keywords

Text categorization, classification, support vector machines, machine learning, information management.

2. INTRODUCTION

As the volume of information available on the Internet and corporate intranets continues to increase, there is growing interest in helping people better find, filter, and manage these resources. *Text categorization* – the assignment of natural language texts to one or more predefined categories based on their content – is an important component in many information organization and management tasks. Its most widespread application to date has been for assigning subject categories to documents to support text retrieval, routing and filtering.

Automatic text categorization can play an important role in a wide variety of more flexible, dynamic and personalized information management tasks as well: real-time sorting of email or files into folder hierarchies; topic identification to support topic-specific processing operations; structured search and/or browsing; or finding documents that match long-term standing interests or more dynamic task-based interests. Classification technologies should be able to support category structures that are very general, consistent across individuals, and relatively static (e.g., Dewey Decimal or Library of Congress classification systems, Medical Subject Headings (MeSH), or Yahoo!'s topic hierarchy), as well as those that are more dynamic and customized to individual interests or tasks (e.g., email about the CIKM conference).

In many contexts (Dewey, MeSH, Yahoo!, CyberPatrol), trained professionals are employed to categorize new items. This process is very time-consuming and costly, thus limiting its applicability. Consequently there is increased interest in developing technologies for automatic text categorization. Rule-based approaches similar to those used in expert systems are common (e.g., Hayes and

Weinstein's CONSTRUE system for classifying Reuters news stories, 1990), but they generally require manual construction of the rules, make rigid binary decisions about category membership, and are typically difficult to modify. Another strategy is to use *inductive learning* techniques to *automatically* construct classifiers using labeled training data. Text classification poses many challenges for inductive learning methods since there can be millions of word features. The resulting classifiers, however, have many advantages: they are easy to construct and update, they depend only on information that is easy for people to provide (i.e., examples of items that are in or out of categories), they can be customized to specific categories of interest to individuals, and they allow users to smoothly tradeoff precision and recall depending on their task.

A growing number of statistical classification and machine learning techniques have been applied to text categorization, including multivariate regression models (Fuhr et al., 1991; Yang and Chute, 1994; Schütze et al., 1995), nearest neighbor classifiers (Yang, 1994), probabilistic Bayesian models (Lewis and Ringuette, 1994), decision trees (Lewis and Ringuette, 1994), neural networks (Wiener et al., 1995; Schütze et al., 1995), and symbolic rule learning (Apte et al., 1994; Cohen and Singer, 1996). More recently, Joachims (1998) has explored the use of Support Vector Machines (SVMs) for text classification with promising results.

In this paper we describe results from experiments using a collection of hand-tagged financial newswire stories from Reuters. We use supervised learning methods to build our classifiers, and evaluate the resulting models on new test cases. The focus of our work has been on comparing the effectiveness of different inductive learning algorithms (Find Similar, Naïve Bayes, Bayesian Networks, Decision Trees, and Support Vector Machines) in terms of learning speed, real-time classification speed, and classification accuracy. We also explored alternative document representations (words vs. syntactic phrases, and binary vs. non-binary features), and training set size.

3. INDUCTIVE LEARNING METHODS

3.1 Classifiers

A classifier is a function that maps an input attribute vector, $\vec{x} = (x_1, x_2, x_3, \ldots x_n)$, to a confidence that the input belongs to a class – that is, $f(\vec{x})$ = *confidence(class)*,. In the case of text classification, the attributes are words in the document and the classes correspond to text categories (e.g., typical Reuters categories include *acquisitions*, *earnings*, *interest*).

Examples of classifiers for the Reuters category *interest* include:

- if (interest AND rate) OR (quarterly), then *confidence(interest category)* = 0.9
- *confidence(interest category)* = 0.3*interest + 0.4*rate + 0.7*quarterly

Some of the classifiers that we consider (decision trees, naïve-Bayes classifier, and Bayes nets) are probabilistic in the sense that *confidence(class)* is a probability distribution.

3.2 Inductive Learning of Classifiers

Our goal is to learn classifiers like these using inductive learning methods. In this paper we compared five learning methods:

- Find Similar (a variant of Rocchio's method for relevance feedback)
- Decision Trees
- Naïve Bayes
- Bayes Nets
- Support Vector Machines (SVM)

We describe these different models in detail in section 2.4. All methods require only on a small amount of labeled training data (i.e., examples of items in each category) as input. This training data is used to "learn" parameters of the classification model. In the testing or evaluation phase, the effectiveness of the model is tested on previously unseen instances.

Learned classifiers are easy to construct and update. They require only subject knowledge ("I know it when I see it") and not programming or rule-writing skills. Inductively learned classifiers make it easy for users to customize category definitions, which is important for some applications. In addition, all the learning methods we looked at provide graded estimates of category membership allowing for tradeoffs between precision and recall, depending on the task.

3.3 Text Representation and Feature Selection

Each document is represented as a vector of words, as is typically done in the popular vector representation for information retrieval (Salton & McGill, 1983). For the Find Similar algorithm, tf*idf term weights are computed and all features are used. For the other learning algorithms, the feature space is reduced substantially (as described below) and only binary feature values are used – a word either occurs or does not occur in a document.

For reasons of both efficiency and efficacy, feature selection is widely used when applying machine learning methods to text categorization. To reduce the number of features, we first remove features based on overall frequency counts, and then select a small number of features based on their fit to categories. Yang and Pedersen (1997) compare a number of methods for feature selection. We used the mutual information measure. The mutual information $MI(x_i, c)$ between a feature, x_i, and a category, c is defined as:

$$MI(x_i, c) = \sum_{x_i \in \{0,1\}} \sum_{c \in \{0,1\}} P(x_i, c) \log \frac{P(x_i, c)}{P(x_i)P(c)}$$

We select the k features for which mutual information is largest for each category. These features are used as input to the various inductive learning algorithms. For the SVM and decision-tree methods we used k=300, and for the remaining methods we used k=50. We did not rigorously explore the optimum number of features for this problem, but these numbers provided good results on a training validation set so they were used for testing.

3.4 Inductive Learning of Classifiers

3.4.1 Find Similar

Our Find Similar method is a variant of Rocchio's method for relevance feedback (Rocchio,. 1971) which is a popular method for expanding user queries on the basis of relevance judgements. In Rocchio's formulation, the weight assigned to a term is a combination of its weight in an original query, and judged relevant and irrelevant documents.

$$x_j = \alpha \cdot x_{q,j} + \beta \cdot \frac{\sum_{i \in rel} x_{i,j}}{n_r} + \gamma \cdot \frac{\sum_{i \in non-rel} x_{i,j}}{N - n_r}$$

The parameters α, β, and γ control the relative importance of the original query vector, the positive examples and the negative examples. In the context of text classification, there is no initial query, so α=0. We also set γ=0 so we could easily use available code. Thus, for our Find Similar method the weight of each term is simply the average (or centroid) of its weights in positive instances of the category.

There is no explicit error minimization involved in computing the Find Similar weights. Thus, there is no learning time so to speak, except for taking the sum of weights from positive examples of each category. Test instances are classified by comparing them to the category centroids using the Jaccard similarity measure. If the score exceeds a threshold, the item is classified as belonging to the category.

3.4.2 Decision Trees

A decision tree was constructed for each category using the approach described by Chickering et al. (1997). The decision trees were grown by recursive greedy splitting, and splits were chosen using the Bayesian posterior probability of model structure. We used a *structure prior* that penalized each additional parameter with probability 0.1, and derived parameter priors from a prior network as described in Chickering et al. (1997) with an equivalent sample size of 10. A class probability rather than a binary decision is retained at each node.

3.4.3 Naïve Bayes

A naïve-Bayes classifier is constructed by using the training data to estimate the probability of each category given the document feature values of a new instance. We use Bayes theorem to estimate the probabilities:

$$P(C = c_k \mid \vec{x}) = \frac{P(\vec{x} \mid C = c_k) P(C = c_k)}{P(\vec{x})}$$

The quantity $P(\vec{x} \mid C = c_k)$ is often impractical to compute without simplifying assumptions. For the Naïve Bayes classifier (Good, 1965), we assume that the features $X_1,\ldots X_n$ are conditionally independent , given the category variable *C*. This simplifies the computations yielding:

$$P(\vec{x} \mid C = c_k) = \prod_i P(x_i \mid C = c_k)$$

Despite the fact the assumption of conditional independence is generally not true for word appearance in documents, the Naïve Bayes classifier is surprisingly effective.

3.4.4 Bayes Nets

More recently, there has been interest in learning more expressive Bayesian networks (Heckerman et al., 1995) as well as methods for learning networks specifically for classification (Sahami, 1996). Sahami, for example, allows for a limited form of dependence between feature variables, thus relaxing the very restrictive assumptions of the Naïve Bayes classifier. We used a 2-dependence Bayesian classifier that allows the probability of each feature x_i to be directly influenced by the appearance/non-appearance of at most two other features.

3.4.5 Support Vector Machines (SVMs)

Vapnik proposed Support Vector Machines (SVMs) in 1979 (Vapnik, 1995), but they have only recently been gaining popularity in the learning community. In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin – see Figure 1.

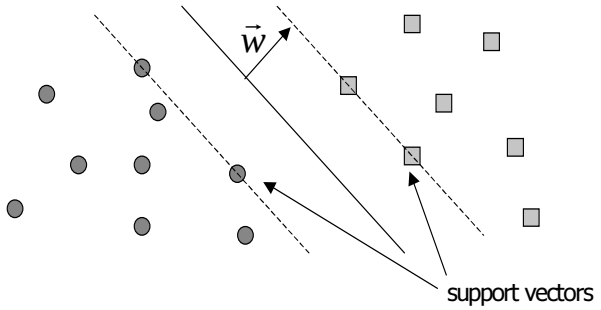

Figure 1 – Linear Support Vector Machine

The formula for the output of a linear SVM is $u = \vec{w} \cdot \vec{x} - b$, where $\vec{w}$ is the normal vector to the hyperplane, and $\vec{x}$ is the input vector.

In the linear case, the margin is defined by the distance of the hyperplane to the nearest of the positive and negative

examples. Maximizing the margin can be expressed as an optimization problem: minimize $\frac{1}{2}\|\vec{w}\|^2$ subject to $y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1, \forall i$ where x_i is the ith training example and y_i is the correct output of the SVM for the ith training example. Of course, not all problems are linearly separable. Cortes and Vapnik (1995) proposed a modification to the optimization formulation that allows, but penalizes, examples that fall on the wrong side of the decision boundary. Additional extensions to non-linear classifiers were described by Boser et al. in 1992. SVMs have been shown to yield good generalization performance on a wide variety of classification problems, including: handwritten character recognition (LeCun et al., 1995), face detection (Osuna et al., 1997) and most recently text categorization (Joachims, 1998). We used the simplest linear version of the SVM because it provided good classification accuracy, is fast to learn and fast for classifying new instances.

Training an SVM requires the solution of a QP problem Any quadratic programming (QP) optimization method can be used to learn the weights, $\vec{w}$, on the basis of training examples. However, many QP methods can be very slow for large problems such as text categorization. We used a new and very fast method developed by Platt (1998) which breaks the large QP problem down into a series of small QP problems that can be solved analytically. Additional improvements can be realized because the training sets used for text classification are sparse and binary. Once the weights are learned, new items are classified by computing $\vec{w}\cdot\vec{x}$ where $\vec{w}$ is the vector of learned weights, and $\vec{x}$ is the binary vector representing the new document to classify.

After training the SVM, we fit a sigmoid to the output of the SVM using regularized maximum likelihood fitting, so that the SVM can produce posterior probabilities that are directly comparable between categories.

4. REUTERS DATA SET

4.1 Reuters-21578 (ModApte split)

We used the new version of Reuters, the so-called Reuters-21578 collection. (This collection is publicly available at: http://www.research.att.com/~lewis/reuters21578.html.) We used the 12,902 stories that had been classified into 118 categories (e.g., corporate acquisitions, earnings, money market, grain, and interest). The stories average about 200 words in length.

We followed the ModApte split in which 75% of the stories (9603 stories) are used to build classifiers and the remaining 25% (3299 stories) to test the accuracy of the resulting models in reproducing the manual category assignments. The stories are split temporally, so the training items all occur before the test items. The mean number of categories assigned to a story is 1.2, but many stories are not assigned to any of the 118 categories, and some stories are assigned to 12 categories. The number of stories in each category varied widely as well, ranging from "earnings" which contains 3964 documents to "castor-oil" which contains only one test document. Table 1 shows the ten most frequent categories along with the number of training and test examples in each. These 10 categories account for 75% of the training instances, with the remainder distributed among the other 108 categories.

Category Name	Num Train	Num Test
Earn	2877	1087
Acquisitions	1650	719
Money-fx	538	179
Grain	433	149
Crude	389	189
Trade	369	118
Interest	347	131
Ship	197	89
Wheat	212	71
Corn	182	56

Table 1 – Number of Training/Test Items

4.2 Summary of Inductive Learning Process for Reuters

Figure 2 summarizes the process we use for testing the various learning algorithms. Text files are processed using Microsoft's Index Server. All features are saved along with their tf*idf weights. We distinguished between words occurring in the Title and Body of the stories. For the Find Similar method, similarity is computed between test examples and category centroids using all these features. For all other methods, we reduce the feature space by eliminating words that appear in only a single document (hapax legomena), then selecting the k words with highest mutual information with each category. These k-element binary feature vectors are used as input to four different learning algorithms. For SVMs and decision trees k=300, and for the other methods, k=50.

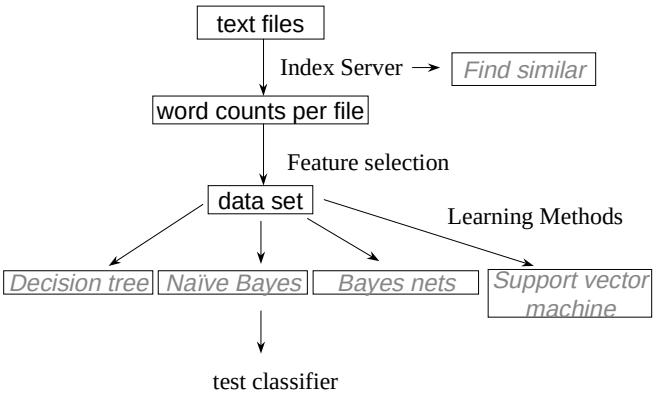

Figure 2 – Schematic of Learning Process

A separate classifier is learned for each category. New instances are classified by computing a score and comparing the score with a learned threshold. New instances exceeding the threshold are said to belong to the category. As already mentioned, all classifiers output a graded measure of category membership, so different thresholds can be set to favor precision or recall depending on the application – for Reuters we optimized the average of precision and recall (details below).

All model parameters and thresholds are set to optimize performance on a validation set and are not modified during testing. For Reuters, the training set contains 9603 stories and the test set 3299 stories. In order to decide which models to use we performed initial experiments on a subset of the training data, which we subdivided into 7147 training stories and 2456 validation stories for this purpose. We used this to set the number of features (k), decision thresholds and document representations to use for the final runs. We estimated parameters for these chosen models using the full 9603 training stories and evaluated performance on the 3299 test items. We did *not* further optimize performance by tuning parameters to achieve optimal performance in the test set.

5. RESULTS

5.1 Training Time

Training times for the 9603 training examples vary substantially across methods. We tested these algorithms on a 266MHz Pentium II running Windows NT. Unless otherwise noted times are for the 10 largest categories, because they take longest to learn. Find Similar is the fastest "learning" method (<1 CPU sec/category) because there is no explicit error minimization. The linear SVM is the next fastest (<2 CPU secs/category). These are both substantially faster than Naïve Bayes (8 CPU secs/category), Bayes Nets (~145 CPU secs/category) or Decision Trees (~70 CPU secs/category). In general, performing the mutual-information feature-extraction step takes much more time than any of the inductive learning algorithms. The linear SVM with SMO, for example, takes an average of 0.26 CPU seconds to train a category when averaged over all 118 Reuters categories.

The training speeds for the SVM are particularly impressive, since training speed has been a barrier to its wide spread applicability for large problems. Platt's SMO algorithm is roughly 30 times faster than the popular chunking algorithm on the Reuters data set (Vapnik, 1995).

5.2 Classification Speed for New Instances

In many applications, it is important to quickly classify new instances. All of the classifiers we explored are very fast in this regard – all require less than 2 msec to determine if a new document should be assigned to a particular category. Far more time is spent in pre-processing the text to extract even simple words than is spent in categorization. With the SVM model, for example, we need only compute $\vec{w} \cdot \vec{x}$, where $\vec{w}$ is the vector of learned weights, and $\vec{x}$ is feature vector for the new instance. Since features are binary, this is just the sum of up to 300 numbers.

5.3 Classification Accuracy

Many evaluation criteria for classification have been proposed. The most popular measures are based on precision and recall. Precision is the proportion of items placed in the category that are really in the category, and Recall is the proportion of items in the category that are actually placed in the category. We report the average of precision and recall (the so-called breakeven point) for comparability to earlier results in text classification. In addition, we plot precision as a function of recall in order to understand the relationship among methods at different points along this curve. Table 2 summarizes micro-averaged break even performance for the 5 different learning algorithms for the 10 most frequent categories as well as the overall score for all 118 categories.

Support Vector Machines were the most accurate method, averaging 92% for the 10 most frequent categories and 87% over all 118 categories. Accuracy for Decision Trees was 3.6% lower, averaging 88.4% for the 10 most frequent categories. Bayes Nets provided some performance improvement over Naïve Bayes as expected, but the advantages were rather small. As has previously been reported, all the more advanced learning algorithms increase performance by 15-20% compared with Rocchio-

	Findsim	NBayes	BayesNets	Trees	LinearSVM
earn	92.9%	95.9%	95.8%	97.8%	98.0%
acq	64.7%	87.8%	88.3%	89.7%	93.6%
money-fx	46.7%	56.6%	58.8%	66.2%	74.5%
grain	67.5%	78.8%	81.4%	85.0%	94.6%
crude	70.1%	79.5%	79.6%	85.0%	88.9%
trade	65.1%	63.9%	69.0%	72.5%	75.9%
interest	63.4%	64.9%	71.3%	67.1%	77.7%
ship	49.2%	85.4%	84.4%	74.2%	85.6%
wheat	68.9%	69.7%	82.7%	92.5%	91.8%
corn	48.2%	65.3%	76.4%	91.8%	90.3%
Avg Top 10	64.6%	81.5%	85.0%	88.4%	92.0%
Avg All Cat	61.7%	75.2%	80.0%	N/A	87.0%

Table 2 – Breakeven Performance for 10 Largest Categories, and over all 118 Categories.

style query expansion (Find Similar).

Both SVMs and Decision Trees produce very high overall classification accuracy, and are among the best known results for this test collection. Most previous results have used the older Reuters collection, so it is difficult to compare precisely, but 85% is the best micro-averaged breakeven point previously reported (Yang, 1997). Joachims (1998) used the new collection, and our SVM results are more accurate (87% for our linear SVM vs. 84.2% for Joachims' linear SVM and 86.5% for his radial basis function network with gamma equals 0.8) and far more efficient for both initial model learning and for real-time classification of new instances. It is also worth noting that Joachims chose optimal parameters based on the test data and used only the 90 categories that have at least one training and test item, and our results would improve some if we did the same. Apte, et al. (1998) have recently reported accuracies slightly better than ours (87.8%) for a system with 100 decision trees. Their approach involves learning many decision trees using an adaptive resampling approach (boosting) and is much more complex to learn than our one simple linear classifier.

The 92% breakeven point (for the top 10 categories) corresponds roughly to 92% precision at 92% recall. Note, however, that the decision threshold can be varied to produce higher precision (at the cost of lower recall), or higher recall (at the cost of lower precision), as appropriate for different applications. A user would be quite happy with 92% precision for information discovery tasks, but might want additional human confirmation before deleting important email messages with this level of accuracy. Figure 3 shows a representative ROC curve for the category "grain". The advantages of SVM can be seen over the entire recall-precision space.

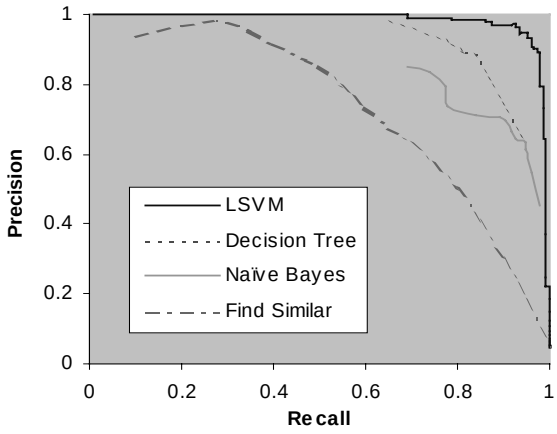

Figure 3 – Precision-Recall Curve for Category "grain"

Although we have not conducted any formal tests, the learned classifiers appear to be intuitively reasonable. For example, the SVM representation for the category "interest" includes the words prime (.70), rate (.67), interest (.63), rates (.60), and discount (.46) with large positive weights, and the words group (-.24), year (-.25), sees (-.33) world (-.35), and dlrs (-.71) with large negative weights.

5.4 Other Experiments

5.4.1 Sample Size

For an application like Reuters, it is easy to imagine developing a large training corpus of the sort we worked with (e.g., a few categories had more than 1000 positive training instances). For other applications, training data may be much harder to come by. For this reason we examined how many positive training examples were necessary to provide good generalization performance. We looked at performance for the 10 most frequent categories, varying the number of positive instances but keeping the negative data the same. For the linear SVM, using 100% of the training data (7147 stories), the micro-averaged breakeven point is 92%. For smaller training sets we took multiple random samples and report the average score. Using only 10% of the training sets data performance is 89.6%, with a 5% sample 86.2%, and with a 1% sample 72.6%. When we get down to a training set with only 1% of the positive examples, most of the categories have fewer than 5 training instances resulting in somewhat unstable performance for some categories. In general, having 20 or more training instances provides stable generalization performance.

While the number of examples needed per category will vary across application, we find these results encouraging. In addition, it is important to note that in most categorization scenarios, the distribution of instances varies tremendously across categories – some categories will have hundreds or thousands of instances, and others only a few (a kind of Zipf's law for category size). In such cases, the most popular categories will quickly receive the necessary number of training examples in the normal course of operation.

5.4.2 Simple words vs. NLP-derived phrases

For all the results reported so far, we simply used the default pre-processing provided by Microsoft's Index Server, resulting in single words as index terms. We wanted to explore how NLP analyses might improve classification accuracy. For example, the phrase "interest rate" is more predictive of the Reuters category "interest" than is either the word "interest" or "rate". We used NLP analyses in a very simply fashion to aid in the extraction of richer phases for indexing accuracy (see Lewis and Sparck Jones, 1996 for an overview of related NLP issues). We considered:

- factoids (e.g., Salomon_Brothers_International, April_8)
- multi-word dictionary entries (e.g., New_York, interest_rate)
- noun phrases (e.g., first_quarter, modest_growth)

As before, we used tf*idf weights for Find Similar and the mutual information criterion for selecting features for Naïve Bayes and SVM. Unfortunately, the NLP-derived phrases did not improve classification accuracy. For the SVM, the NLP features actually reduced performance on the 118 categories by 0.2% Because of these initial results, we did not try the NLP-derived phrases for Decision Trees or the more complex 2-dependence Bayesian network, or use NLP features in any of the final evaluations.

5.4.3 Binary vs. 0/1/2 features

We also looked at whether moving to a richer representation than binary features would improve categorization accuracy. To this end, we considered a representation that encoded words as appearing 0,1, or >=2 times in each document. Initial results using this representation with Decision Tree classifiers did not yield improved performance, so we did not pursue this further.

6. SUMMARY

Very accurate text classifiers can be learned automatically from training examples, as others have shown. The accuracy of our simple linear SVM is among the best reported for the Reuters-21578 collection. In addition, the model is very simple (300 binary features per category), and Platt's SMO training method for SVMs provides a very efficient method for learning the classifier – at least 30 times faster than the chunking method for QP, and 35 times faster than the next most accurate classifier (Decision Trees) we examined. Classification of new items is fast as well since we need only compute the sum of the learned weights for features in the test items.

We found that the simplest document representation (using individual words delimited by white spaces with no stemming) was at least as good as representations involving more complicated syntactic and morphological analysis. And, representing documents as binary vectors of words, chosen using a mutual information criterion for each category, was as good as finer-grained coding (at least for Decision Trees).

Joachims (1998) work is similar to ours in its use of SVMs for the purpose of text categorization. Our results are somewhat more accurate than his, but, more importantly, based on a much simpler and more efficient model. Joachims' best results are obtained using a non-linear radial basis function of 9962 real-valued input features (based on the popular tf*idf term weights). In contrast, we use a single linear function of 300 binary features per category.

SVMs work well because they create a classifier which maximizes the margin between positive and negative examples. Other algorithms, such as boosting (Schapire, et al., 1998), have been shown to maximize margin and are also very effective at text categorization.

We have also used SVMs for categorizing email messages and Web pages with results comparable to those reported here -- SVMs are the most accurate classifier and the fastest to train. We hope to extend the text representation models to include additional structural information about documents, as well as knowledge-based features which have been shown to provide substantial improvements in classification accuracy (Sahami et al., 1998). Finally, we will look at extending this work to automatically classify items into hierarchical category structures.

We believe that inductive learning methods like the ones we have described can be used to support flexible, dynamic, and personalized information access and management in a wide variety of tasks. Linear SVMs are particularly promising since they are both very accurate and fast.

7. REFERENCES

[1] Apte, C., Damerau, F. and Weiss, S. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems, 12(3),* 233-251, 1994.

[2] Apte, C., Damerau, F. and Weiss, S.. Text Mining with decision rules and decision trees. *Proceedings of the Conference on Automated Learning and Discovery*, CMU, June, 1998.

[3] Boser, B. E., Guyon, I. M., and Vapnik, V., A Training Algorithm for Optimal Margin Classifiers. *Fifth Annual Workshop on Computational Learning Theory*, ACM, 1992.

[4] Chickering D., Heckerman D., and Meek, C. A Bayesian approach for learning Bayesian networks with local structure. In *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.

[5] Cohen, W.W. and Singer, Y. Context-sensitive learning methods for text categorization In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 307-315, 1996.

[6] Cortes, C., and Vapnik, V., Support vector networks. *Machine Learning, 20*, 273-297, 1995.

[7] Fuhr, N., Hartmanna, S., Lustig, G., Schwantner, M., and Tzeras, K. Air/X – A rule-based multi-stage indexing system for lage subject fields. In *Proceedings of RIAO'91*, 606-623, 1991.

[8] Good, I.J. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. MIT Press, 1965.

[9] Hayes, P.J. and Weinstein. S.P. CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.

[10] Heckerman, D. Geiger, D. and Chickering, D.M. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning, 20*, 131-163, 1995.

[11] Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings 10th European Conference on Machine Learning (ECML)*, Springer Verlag, 1998. http://www-ai.cs.uni-dortmund.de/DOKIMENTE/Joachims_97a.ps.gz

[12] LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P. and Vapnik, V. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks: The Statistical Mechanics Perspective*, 261-276, 1995.

[13] Lewis, D.D.. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR'92: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 37-50, 1992.

[14] Lewis, D.D. and Hayes, P.J. (Eds.) *ACM Transactions on Information Systems – Special Issue on Text Categorization, 12(3)*, 1994.

[15] Lewis, D.D. and Ringuette, M.. A comparison of two learning algorithms for text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*, 81-93, 1994.

[16] Lewis. D.D. and Sparck Jones. K. Natural language processing for information retrieval. *Communications of the ACM, 39(1)*, 92-101, January 1996.

[17] Lewis, D.D., Schapire, R., Callan, J.P., and Papka, R. Training algorithms for linear text classifiers. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 298-306, 1996.

[18] Osuna, E., Freund, R., and Girosi, F. Training support vector machines: An application to face detection. In *Proceedings of Computer Vision and Pattern Recognition '97*, 130-136, 1997.

[19] Platt, J. Fast training of SVMs using sequential minimal optimization. To appear in: B. Scholkopf, C. Burges, and A. Smola (Eds.) *Advances in Kernel Methods – Support Vector Learning*, MIT Press, 1998.

[20] Rocchio, J.J. Jr. Relevance feedback in information retrieval. In G.Salton (Ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing*, 313-323. Prentice Hall, 1971.

[21] Sahami, M. Learning Limited Dependence Bayesian Classifiers. In *KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 335-338, AAAI Press, 1996. http://robotics.stanford.edu/users/sahami/papers-dir/kdd96-learn-bn.ps

[22] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. A Bayesian approach to filtering junk e-mail. *AAAI 98 Workshop on Text Categorization*, July 1998. http://robotics.stanford.edu/users/sahami/papers-dir/spam.ps

[23] Salton, G. and McGill, M. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[24] Schapire, R., Freund, Y., Bartlett, P. and Lee, W. S. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, to appear, 1998.

[25] Schütze, H., Hull, D. and Pedersen, J.O. A comparison of classifiers and document representations for the routing problem. In *SIGIR '95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 229-237, 1995.

[26] Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

[27] Wiener E., Pedersen, J.O. and Weigend, A.S. A neural network approach to topic spotting. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, 1995.

[28] Yang, Y. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. *SIGIR '94: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 13-22, 1994.

[29] Yang. Y. and Chute, C.G. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems, 12(3)*, 252-277, 1994.

[30] Yang, Y. and Pedersen, J.O. A comparative study on feature selection in text categorization. *In Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*, 412-420, 1997.

[31] Yang, Y. An evaluation of statistical approaches to text categorization. *CMU Technical Report*, CMU-CS-97-127, April 1997.

[32] The Reuters-21578 collection is available at: http://www.research.att.com/~lewis/reuters21578.html