Text Categorization for a Comprehensive Time-Dependent Benchmark

Fred J. Damerau, Tong Zhang, Sholom M. Weiss and Nitin Indurkhya
IBM T.J. Watson Research Center,
P.O. Box 218, Yorktown Heights, NY 10598, USA
damerau@us.ibm.com, tongz@us.ibm.com, sholom@us.ibm.com, nitin@data-miner.com

Abstract

We present results for automated text categorization of the Reuters-810000 collection of news stories. Our experiments use the entire one-year collection of 810,000 stories and the entire subject index. We divide the data into monthly groups and provide an initial benchmark of text categorization performance on the complete collection. Experimental results show that efficient sparse-feature implementations of linear methods and decision trees, using a global unstemmed dictionary, can readily handle applications of this size. Predictive performance is approximately as strong as the best results for the much smaller older Reuters collections. Detailed results are provided over time periods. It is shown that a smaller time horizon does not appreciably diminish predictive quality, implying reduced demands for retraining when sample size is large.

Keywords: text categorization, machine learning, scalability, benchmark, classification of very large corpora.

1 Introduction

Automated text categorization methods have gradually improved over time, to the point where they can often rival human performanceLewis (1992); Apte et al. (1994); Yang (1999). One major contributor to progress in the field has been the availability of standard benchmarks that allow researchers to evaluate performance relative to prior work, e.g., the Reuters-21578 collection of 21,578 stories from the year 1987. A new, far larger collection of Reuters newswires is now publicly availableReu. In this paper, we make use of this data set to establish a new benchmark for evaluating text categorization performance in a high dimensional space. The new Reuters-810000 collection was examined in restricted form during the 2001 TREC Filtering TrackTre (2001), where only the first 26,150 cases were used for training, and the topics were a subset of the Reuters topics. Here we use all months except the last for training, and all original topics are considered.

The availability of a very large collection of texts with classification codes from Reuters has made possible the exploration of some open questions in the construction of classification models. One of these questions is how rapidly a classification model decays, i.e., a classification model trained with documents from t_1 to t_2 has a certain accuracy at time t_3, but presumably will have a lower accuracy at time t_n, decreasing with increasing n. Since the corpus covers news stories for one year, we are able to look at this effect over that time. Two types of learning methods were employed in our study: a decision tree classifier and a linear classifier. We examined several additional issues that influence predictive performance, including the effect of feature set size, the use of inflectional stemming and the use of titles.

2 Corpus Details

The new Reuters corpus consists of some 810 thousand stories covering the Reuters newswire over a year, August 20, 1996 to August 19, 1997. Uncompressed, it is about 2.5 gigabytes. Each story has a certain amount of associated metadata, such as creation date, publication date, source, etc. Included in the metadata are three codes: region codes, topic codes and industry codes. We focus here on the topic codes. Notice that (a) there is wide variation in the frequency of code assignment and (b) the code structure is in part hierarchical. For example. ECAT is "economics", E21 is "government/finance", E211 is "expenditure/revenues" and E212 is "government borrowing". A story classified as E211 is necessarily also classified as E21 and ECAT. However, not all stories are

classified down to a leaf node. There are stories classified as E21 which are not E211 or E212, as can be seen by looking at the respective frequencies of code assignment. Some algorithms have been proposed to exploit hierarchies in the classification process, but we have not pursued this, treating each code as a separate classification problem. Because more than one code can be assigned to a story, we treat classification as a collection of binary problems, rather than building a multi-class model.

3 Classification Methods

In our applied work for text categorization, we ordinarily use one of two models, a decision tree or a linear classifier similar to those studied in Zhang and Oles (2001). The decision tree classifier has the advantage that the tree can be reduced to a set of more or less understandable classification rules, which allow a classification model to be manually adjusted if desired. The linear model is, however, generally more accurate. Brief descriptions of our algorithms and implementations follow; details can be found in the references.

3.1 Decision tree model

In the text categorization application, interpretability of the models used is an important characteristic to be considered in addition to the accuracy achieved and the computational requirements. The requirement of interpretability can be satisfied by using a rule-based system, such as rules obtained from a decision tree. Rule-based systems are particularly appealing since a person can examine the rules and modify them. Rules can also be manually written without using any training data.

We thus include a test of a decision tree based classifier. (This package is used by IBM for text categorization.) In a typical decision tree training algorithm, there are usually two stages. The first stage is tree growing where a tree is built by greedily splitting each tree node based on a certain figure of merit. However after the first stage, the tree can overfit the training data, (i.e., the tree essentially reconstructs the original data rather than being a summary of the data. The effect is that the tree will not apply to new input because it is too specific.) Therefore a second stage involving tree pruning is invoked. In this stage, one removes overfitted branches of the tree so that the remaining portion has better predictive power. In our decision tree package, the splitting

criteria during tree growth is similar to that of the standard C4.5 program Quinlan (1993), and the tree pruning is done using a Bayesian model combination approach. See Johnson et al. (2002) for detailed description.

One useful aspect of our decision tree is that we take advantage of the sparse data representation. The algorithm requires only a small amount of memory and little time for sparse data, where each story contains a tiny fraction of the words in the dictionary. A standard decision tree package such as C4.5 that does not take advantage of sparsity in the data will not be able to handle our applications.

Although deriving rules from decision trees is efficient and the rules are usually interpretable, these methods do not provide the best performance in text categorization. One remedy is to use the so-called "boosting" procedure which votes a large number of decision trees and picks the category that receives the most votes. In fact, the best reported text categorization result on the standard Reuters-21578 evaluation dataset is achieved using boosted decision trees Weiss et al. (1999), in contrast to other methods such as nearest neighbor, single decision trees, decision rules, naive Bayes, and the like. Unfortunately, this approach requires a large number (one hundred as in Weiss et al. (1999)) of trees. The resulting system not only loses the interpretability of a single decision tree, but is also computationally too expensive to be practically interesting. We do not consider this method in this paper.

It is known that in text categorization, nearly the same level of performance achieved by boosted decision trees can be achieved by computationally more efficient linear classification methods Dumais et al. (1998); Joachims (1998); Zhang and Oles (2001). It is thus natural to include linear classification in our experiments.

3.2 Linear model

We consider a two-class categorization problem to be one that determines a label $y \in \{-1, 1\}$ from an associated vector x of input variables. Given a continuous model $p(x)$, we consider the following prediction rule: predict $y = 1$ if $p(x) \geq 0$, and predict $y = -1$ otherwise. The classification error (we shall ignore the point $p(x) = 0$, which is assumed to occur rarely) is

$$I(p(x), y) = \begin{cases} 1 & \text{if } p(x)y \leq 0, \\ 0 & \text{if } p(x)y > 0. \end{cases}$$

A useful method for solving this problem is by linear predictors. These consist of linear combinations of the input variables: $p(x) = w^T x + b$, where w is often referred to as *weight* and b as *bias*. However, in this paper, we call (w, b) the weight vector, and use the term bias for statistical bias.

A very natural way to compute a linear classifier is by finding a weight $(\hat{w}, \hat{b})$ that minimizes the average classification error in the training set:

$$(\hat{w}, \hat{b}) = \arg\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} I(w^T x_i + b, y_i).$$

Unfortunately this problem is typically NP-hard computationally. It is thus desirable to replace the classification error loss $I(p, y)$ with another formulation that is computationally more desirable. One modification, employed in a Support Vector Machine (SVM), is to minimize an upper bound of the classification error as:

$$\arg\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} g(w^T x_i + b, y_i),$$

where

$$g(p, y) = \begin{cases} 1 - py & \text{if } py \le 1, \\ 0 & \text{if } py > 1. \end{cases}$$

The resulting optimization problem is convex and thus computationally tractable. Although SVMs have been used successfully in text categorization, in this paper, we shall consider a different method which minimizes the following loss function

$$h(p, y) = \begin{cases} -2py & py < -1 \\ \frac{1}{2}(py - 1)^2 & py \in [-1, 1] \\ 0 & py > 1. \end{cases}$$

That is, our linear weights are computed by minimizing the following average loss on the training data:

$$(\hat{w}, \hat{b}) = \arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} h(w^T x_i + b, y_i). \tag{1}$$

It can be shown that using (1), (2) or (3), one computes a weight $(\hat{w}, \hat{b})$ so that $\max(\min(1, (\hat{w}^T x + \hat{b} + 1)/2), 0)$ gives an estimate of the conditional in-class probability $P(y = 1|x)$. As a contrast, it is difficult to give a reliable confidence estimate from the output of a support vector machine. The proof of these statements, which are beyond the scope of this paper, can be found in Zhang (2003).

Since for certain text categorization applications, conditional probability information can be very useful, we shall use the h-loss formulation in our experiments. The classification performance of this method is comparable to that of support vector machines. For example, our implementation of this method achieves an F-measure of about 86.5 on the Mod-Apte split of Reuters-21578. This performance is comparable to that of support vector machines where the reported range in the literature is approximate from 85 to 87 Dumais et al. (1998); Joachims (1998); Zhang and Oles (2001). Similarly, on the new Reuters corpus studied in this paper, the two methods achieve nearly the same classification performance. For this reason, we shall not include support vector machine results in the paper.

Similarly to the case of a decision tree, using (1) directly to compute a linear classifier can cause overfitting. For decision trees, one solves the problem by tree pruning, which can be regarded as a method of restricting the search space of all possible tree structures. Similarly, one may also restrict the search space of linear classifiers to avoid overfitting. This can be achieved in many different ways. In this paper we consider searching over the weight space defined as:

$$\|w\|_2^2 + b^2 \leq A,$$

where A is a parameter controlling the size of the search space. This method is quite popular and is also adopted in an SVM computation[1]. The resulting method of computing a linear weight can now be written as:

$$(\hat{w}, \hat{b}) = \arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} h(w^T x_i, y_i), \qquad \|w\|_2^2 + b^2 \leq A. \tag{2}$$

From the computational point of view, one can introduce a Lagrangian multiplier λ for the constraint $w^2 \leq A$, and obtain the following equivalent formulation:

$$(\hat{w}, \hat{b}) = \arg\min_{w,b} \left[\frac{1}{n} \sum_{i=1}^{n} h(w^T x_i + b, y_i) + \frac{\lambda}{2}(\|w\|_2^2 + b^2) \right]. \tag{3}$$

This is the formulation used in our computation. The advantage of (3) is that it transforms an constrained optimization problem into an unconstrained optimization problem which is typically

[1]We note that in the standard SVM formulation, b is often not included into the constraint. However the difference is minor and does not have impact on the classification performance.

easier to handle numerically. In this transformation, the regularization parameter A is replaced by λ, which in general can be tuned using cross-validation.

We would like to point out in order to handle large data, it is necessary to use an algorithm that can take advantage of the sparse structure in the bag of word representation of documents. Such algorithms have been suggested in Zhang and Oles (2001) for text categorization. Two types of algorithms were proposed there: one works by going through each feature and updating the corresponding weight component; the other works by going through each data point and updating the weight accordingly. The second approach is closely related to online learning which is desirable for large problems. Since such an algorithm examines the data sequentially, it does not even need to store all the training data in memory at the same time. This allows the algorithm to handle a large amount of data without potential memory issues.

Since in this paper we are dealing with large problems, we use the online-style method to solve (3). The basic idea is to transform (3) into an equivalent dual formulation which depends on a set of variables $\alpha = [\alpha_i]$, where each variable α_i $(i = 0, \ldots, n)$ is associated with a data point (x_i, y_i). In particular, the solution of (3) has the following dual representation Zhang (2002):

$$\hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i x_i y_i, \quad \hat{b} = \sum_{i=1}^{n} \hat{\alpha}_i y_i,$$

where $\hat{\alpha}$ is the solution to the following dual optimization problem:

$$\hat{\alpha} = \arg\min_{\alpha} \left[\sum_{i=1}^{n} \left(\frac{\lambda n}{2} \alpha_i^2 - \alpha_i \right) + \frac{1}{2} \left\| \sum_{i=1}^{n} \alpha_i x_i y_i \right\|_2^2 + \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y_i \right)^2 \right], \tag{4}$$
$$\text{s.t.} \ \ \forall i : \alpha_i y_i \in \left[0, \frac{2}{\lambda n} \right].$$

The equivalence follows directly from the general convex-duality formalism presented in Zhang (2002), which we shall not repeat in this paper. Following Zhang (2002), we use an alternating direction optimization method to solve (4): the idea is to cycle through all data points (x_i, y_i) $i = 1, \ldots, n$, where for each i, we update the corresponding dual variable α_i so as to decrease the objective value in (4), while keeping the remaining dual variables α_j $(j \neq i)$ fixed. Each step is thus a simple one-dimensional optimization problem respect to α_i, which can be minimized exactly in our case. However, instead of using the exact optimization at each step, we employ a gradient

descent rule as in Zhang (2002), which is more generally applicable:

$$\alpha_i \to \alpha_i - \eta[\lambda n \alpha_i - y_i - (w^T x_i + b) y_i], \qquad \text{where} \quad w = \sum_{j=1}^{n} \alpha_j x_j y_j, \; b = \sum_{j=1}^{n} \alpha_j y_j. \tag{5}$$

In the above formula, η is a small step-size quantity that is some times referred to as "learning rate" in the online learning literature. In addition, since we have the constraint $\alpha_i \in [0, 2/\lambda n]$ in (4), we shall choose η to ensure that the updated α_i satisfies this constraint (which is also equivalent to a truncation of the updated α_i onto $[0, 2/\lambda n]$). The quantity $[\lambda n \alpha_i - y_i - (w^T x_i + b) y_i]$ is the gradient of the dual objective function (4) with respect to α_i, and thus the above gradient descent update modifies α_i in the direction that decreases the objective function. In our case, there is a closed form solution of η such that the update (5) exactly minimizes (4) as a function of α_i with α_j $(j \neq i)$ fixed. However, as argued in Zhang (2002), choosing a smaller step-size η usually helps. Moreover, good performance can be achieved by simply picking a small fixed step-size η, where $\eta = 0.25$ is a reasonable generic choice Zhang (2002).

We summarize the method described above in the following algorithm which is used in all of our experiments. The algorithm is scalable and can efficiently handle very large problems.

Algorithm 1

input: training data $(x_1, y_1), \ldots, (x_n, y_n)$

output: weight vector (w, b)

let $\alpha_i = 0$ $(i = 1, \ldots, n)$

let $w = 0$ and $b = 0$

for $k = 1, \ldots, K$

  for $i = 1, \ldots, n$

    $p = (w^T x_i + b) y_i$

    $\Delta\alpha^i = \max(\min(2c - \alpha^i, \eta(\frac{c - \alpha^i}{c} - p)), -\alpha^i)$

    $w = w + \Delta\alpha^i x_i y_i$

    $b = b + \Delta\alpha^i y_i$

    $\alpha^i = \alpha^i + \Delta\alpha^i$

  end

end

In the above algorithm, $c = 1/\lambda n$. As mentioned above, we fix the learning rate parameter $\eta = 0.25$. The algorithm can be terminated when a certain stopping criterion is met. For example, one criterion is the so-called "duality gap" which is frequently used in the optimization literature, and described in Zhang (2002) for our problems. However, a simpler method of using a fixed number of iterations K generally works well Zhang (2002). In this paper, for simplicity, we shall use a fixed number $K = 40$. In our implementation, we also employ a random data ordering of (x_i, y_i). As explained in Zhang and Oles (2001), this is to avoid the situation that data in the same category (or similar data) are grouped together: in such case, an online type algorithm is likely to perform poorly since it will adjust the weight vector to overfit one category before it sees other categories.

Notice that a smaller λ in (3) corresponds to a larger A in (2). This means that with a smaller λ, a larger weight space is searched and thus the statistical bias is smaller, but one is more likely to overfit the training data. Conversely, a larger λ means searching a smaller weight space and is less likely to overfit the training data; however, since the model space is smaller we have a larger bias. It follows that an appropriate choice of λ is necessary for optimal performance. It has been

suggested in Zhang and Oles (2001) that the choice of $\lambda = 10^{-3}$ is typically good for many text data sets. We shall thus use this value as the default. However, in our experiments, we also try $\lambda = 10^{-4}$ since for extremely large data, it is less likely to overfit the data. This means that one can search over a larger weight space to reduce the bias. As we shall see, this improves the performance when the training data size becomes large.

4 Experiments

We partitioned the documents into 12 groups, each corresponding to all stories appearing during a month[2]. A key concept that we wished to explore was the rate of decay of a classification model over time. We used the 12th month as our test data, and trained classification models based on each of the 11 preceding months, the previous two months, the previous three months, the previous six months and the preceding 11 months, i.e., all the available training data. The number of documents per month is 67,500 with some seasonal variation. For example, the number of stories for the first month is 62,935, for the last training month, 66,193, and for the test set, 69,626. Our methods can exploit very large feature sets. In the case of text classification, a feature is a word and the feature value is some measure of the frequency of that word. The feature sets ranged from all possible tokens, to tokens not including numbers, to tokens which had been stemmed by an inflectional stemmer. Case information was retained for all tokens. Token set sizes ranged from almost 400,000 to 10,000 to 1000. Results from these experiments are shown in the tables which follow.

Common performance measures for text categorization are:

- precision: the proportion of classifications that are right with respect to the total of those proposed,
- recall: the proportion of classifications that are correct with respect to the true classification,
- F-measure(F): the harmonic mean of precision and recall:

$$F = 2 * (recall * precision)/(recall + precision) \tag{6}$$

It is empirically the case that in classification problems recall and and precision move in opposite

[2]A month for our Reuters data commences on the 20th calendar day and continues through the 19th day of the following month.

Num of features	Classifier type	Training Data (Months)														
		1	2	3	4	5	6	7	8	9	10	11	10:11	9:11	6:11	1:11
1K	tree	71.9	72.2	72.5	72.5	71.9	73.0	72.7	73.4	74.2	74.3	74.9	76.6	77.5	78.5	79.2
	linear	80.9	81.0	81.1	81.2	80.2	80.6	80.4	81.2	81.8	81.8	82.4	82.6	82.7	82.5	82.5
5K	tree	73.8	73.8	74.5	74.4	73.9	75.0	74.5	75.3	76.0	76.0	76.5	78.1	78.9	79.8	80.6
	linear	83.9	83.9	84.0	84.1	83.2	83.6	83.4	84.1	84.8	84.9	85.2	85.5	85.7	85.6	85.6
20K	tree	73.8	74.0	74.7	74.6	73.9	75.1	74.6	75.4	76.2	76.3	76.6	78.3	79.1	79.9	80.8
	linear	84.2	84.4	84.3	84.5	83.6	84.0	83.7	84.5	85.1	85.2	85.6	85.9	86.0	86.0	86.0
365K	tree	75.0	75.2	75.6	75.9	75.1	76.3	75.9	76.6	77.1	77.0	77.7	79.1	79.9	80.9	81.6
(Titles)	linear	84.8	84.9	84.9	85.1	84.1	84.5	84.2	85.0	85.7	85.7	86.0	86.3	86.5	86.4	86.5

Table 1: F-measure micro-average over all categories by feature set size for decision tree and linear classifier

	Classifier type	Training Data (Months)														
		1	2	3	4	5	6	7	8	9	10	11	10:11	9:11	6:11	1:11
Titles	tree	75.0	75.2	75.6	75.9	75.1	76.3	75.9	76.6	77.1	77.0	77.7	79.1	79.9	80.9	81.6
	linear	84.8	84.9	84.9	85.1	84.1	84.5	84.2	85.0	85.7	85.7	86.0	86.3	86.5	86.4	86.5
No	tree	73.9	74.0	74.6	74.8	74.0	75.1	74.7	75.4	76.1	76.3	76.5	78.2	79.0	79.9	80.7
Stemming	linear	84.2	84.3	84.3	84.5	83.6	83.9	83.7	84.5	85.1	85.1	85.6	85.8	86.0	85.9	86.0
Stemming	tree	73.7	74.2	74.4	74.6	73.6	75.0	74.5	75.3	76.2	76.2	76.5	78.3	79.1	80.2	80.9
	linear	83.9	84.0	84.1	84.2	83.3	83.7	83.5	84.3	84.9	84.9	85.4	85.6	85.7	85.7	85.7

Table 2: F-measure micro-average over all categories for decision tree and linear classifier showing effect of titles and stemming using 365K features

directions. The F-measure is a single number summary of performance which is often convenient. In this paper we report only the F-measure of equation 6. Complete tables showing recall and precision for each experiment are available from the authors. (Accuracy, the absolute number correct over all instances, is a particularly poor measure for document classification problems. Because almost all classification categories are infrequent in a corpus, one can be very accurate simply by predicting for all documents that the category in question does not apply.)

5 Results and Discussion

Consider first the results micro-averaged over all categories using titles and stemming by feature set size, Table 1. Let's first focus on the results for 365K features, the complete set of unstemmed, non-numerical words. In all cases, the linear method is substantially better than the decision tree for all models. (As we will see below, for some of the infrequently occurring categories, the decision

Parameter Choices		Training Data (Months)														
Feature Type	λ	1	2	3	4	5	6	7	8	9	10	11	10:11	9:11	6:11	1:11
TFIDF	10^{-3}	84.8	84.9	84.9	85.1	84.1	84.5	84.2	85.0	85.7	85.7	86.0	86.3	86.5	86.4	86.5
TFIDF	10^{-4}	—	—	—	—	—	—	—	—	—	—	83.9	86.6	87.1	87.5	87.8
BINARY	10^{-3}	83.2	83.5	83.7	83.7	82.7	83.5	83.2	84.1	84.6	84.7	84.9	85.8	86.1	86.4	86.5
BINARY	10^{-4}	—	—	—	—	—	—	—	—	—	—	83.6	8 5.0	85.7	86.6	87.0

Table 3: Effect of parameter adjustment on performance of the linear classifier

tree is better, but the results are still not very good. We will look in detail at one of the rarer categories, religion-GREL.) Returning to the micro average and looking at the models generated by the linear method from month 1 through the full collection of months 1-11, we can see that the F-measure is increasing, but it is not monotonically increasing. Moreover, the model built on month 1 is less than two points lower than the model built on the whole collection for the linear model. The decision tree model is substantially worse, about 5 points lower for month 1 than for the entire collection. The implications of this in an operational environment are significant and comforting. At least for this kind of data, frequent model updating does not appear to be necessary, if the original linear model was built on sufficient data. Figure 1 illustrates this trend for the corporate topic. The situation for decision tree models is not as encouraging. In this case, using more historical data provides a clear improvement over using only one of the preceding months. Whether the transparency of a rule based system merits the use of significantly more training training data depends on the application.

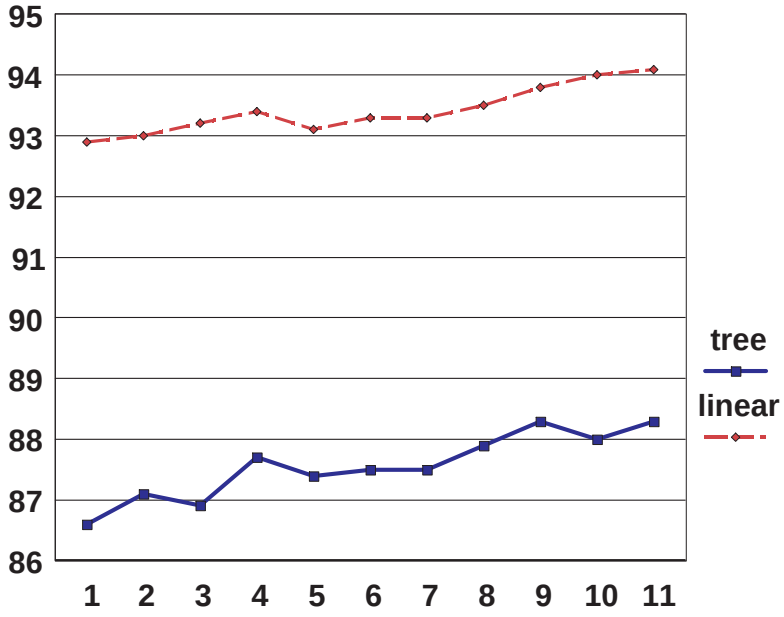

Figure 1: Trend of Error for the Corporate Topic Over 11 Months

Looking further at Table 1, we see that for the linear method there is a clear gain in going from 1000 features to 5000 features, but beyond that the gains are much smaller. Because our methods are relatively insensitive to feature set size, we will continue to focus discussion on the the results

with 365,000 features. The decision tree classifier, however, does not improve greatly when going from 1000 features to 365,000 features, showing a trend to overfit in a very large, noisy feature space.

In Table 2, we see the effects of using titles or not and using stemming or not. By stemming we mean inflectional stemming only. (Combinations of conditions are not shown in the table. Each line shows the result of applying one condition to the corpus). Titles and no stemming, the first result, is best for both the decision tree and linear methods. Without titles, stemming appears to provide a consistent small benefit for the decision tree, and a consistent but small degradation for the linear method. However, for this application, stemming is probably not worth the cost.

The default setting for linear classification can be described as follows. We use $\lambda = 10^{-3}$ as suggested in Zhang and Oles (2001). In the literature, there are two main approaches to using word count information for linear classifiers: one way is to use the so-called TFIDF scoring, as in Joachims (1998); the other is to use binary features to indicate whether a word occurs in the document or not (for example, in Dumais et al. (1998); Zhang and Oles (2001)). In the default setting, we use the TFIDF approach which gives a slightly better result. However, this method consumes more memory and requires more running time. Therefore the binary feature representation can be very useful from a practical point of view. It is also worth mentioning that we do not use any feature selection for linear classification. We assume that the method of restricting the weight space as in (2) can avoid the overfitting problem. This means that we do not have to limit the feature size explicitly (except for potential computational benefits).

Applications of this size can be readily handled using existing algorithms, such as those described in Zhang and Oles (2001); Zhang (2002). On a 500MHZ AIX machine, about four minutes of training time is required per category for the linear classifier using TFIDF, all tokens and all months from 1 to 11. In this case, loading all training data requires about 1 Giga-bytes of memory. On the same machine, the decision tree method is about twice as fast and the linear classifier with binary features is about one third faster.

The difference between the TFIDF representation and the Binary word-count representation can be seen from Table 3. It is clear that the improvement from using TFIDF is consistent but relatively small. The default regularization parameter $\lambda = 10^{-3}$ is clearly better than a smaller regularization parameter value $\lambda = 10^{-4}$ for one month of data, which corresponds to a larger parameter search space in (2). However, if we use more and more data, the choice of $\lambda = 10^{-4}$

Category	Classifier type	Training Data (Months)														
		1	2	3	4	5	6	7	8	9	10	11	10:11	9:11	6:11	1:11
corporate, industrial	tree	86.6	87.1	86.9	87.7	87.4	87.5	87.5	87.9	88.3	88.0	88.3	89.4	89.9	90.5	90.8
	linear	92.9	93.0	93.2	93.4	93.1	93.3	93.3	93.5	93.8	94.0	94.1	94.2	94.3	94.3	94.3
government, social	tree	84.6	84.6	84.2	85.1	84.5	85.4	85.0	84.9	85.3	85.2	85.3	86.6	86.9	87.7	88.3
	linear	92.6	92.7	92.5	92.4	92.4	92.6	92.7	92.8	93.0	92.7	92.8	93.0	93.2	93.3	93.3
markets	tree	85.9	85.7	85.8	86.2	85.8	86.8	87.0	87.6	89.0	88.8	89.3	90.2	90.4	91.3	91.2
	linear	93.2	93.4	93.5	93.4	93.2	93.5	93.5	93.7	94.2	94.1	94.3	94.4	94.5	94.4	94.4
regulation, policy	tree	26.2	27.7	28.7	27.2	26.4	28.9	28.8	30.0	27.8	31.9	28.9	33.3	33.8	35.8	38.0
	linear	47.5	48.9	48.0	51.6	48.3	49.6	50.0	50.4	49.4	50.2	52.7	53.0	52.3	52.1	51.4
sport	tree	86.0	86.7	85.8	87.7	86.7	89.0	88.9	89.3	88.3	88.2	89.4	92.1	91.6	93.2	93.9
	linear	97.5	97.3	97.7	97.6	97.7	97.7	97.9	97.5	97.7	97.9	98.0	98.1	97.9	98.1	98.1
war, civil war	tree	56.5	61.0	57.6	59.7	60.3	58.9	56.4	59.4	55.8	63.8	61.7	64.5	64.8	66.0	66.6
	linear	76.1	76.0	76.8	77.2	77.3	76.0	76.7	77.5	77.0	77.5	77.0	77.5	78.3	78.6	78.6
EC internal market	tree	32.2	26.4	24.4	11.8	20.4	29.2	2.4	10.5	12.8	2.3	11.2	15.1	17.8	18.7	17.0
	linear	30.2	34.2	34.0	21.6	28.1	40.0	16.7	25.6	29.0	21.4	26.5	24.5	29.5	25.6	28.8
balance of payments	tree	44.1	45.0	33.3	42.3	34.8	41.3	45.0	38.3	40.3	36.5	36.6	45.3	48.4	44.2	52.1
	linear	41.7	43.8	37.0	40.8	38.6	35.8	40.3	27.2	27.8	36.5	39.3	37.1	35.4	37.2	40.4
religion	tree	38.4	40.9	42.4	35.2	39.7	56.0	40.6	41.0	56.2	44.7	39.5	60.2	56.2	57.3	58.9
	linear	33.3	36.0	29.3	24.6	33.5	30.1	25.4	32.5	36.6	29.3	32.0	33.5	39.4	40.6	39.0
personal income	tree	0.0	6.2	6.7	0.0	6.2	0.0	0.0	6.2	18.2	0.0	6.2	0.0	12.1	26.3	35.0
	linear	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.7	0.0	0.0	0.0	0.0	0.0
fashion	tree	32.6	47.5	6.1	34.0	9.8	11.1	46.4	32.8	21.6	10.5	33.3	32.9	31.0	51.6	43.2
	linear	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	44.9	38.1	16.2	0.0	0.0
EC environment issues	tree	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	20.0	0.0	20.0	20.0	0.0	14.3
	linear	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 4: Performance on individual categories

becomes superior. The reason is that in this case the algorithm is less prone to overfitting; it is thus beneficial to search over a larger parameter space, i.e., a smaller λ, to reduce bias.

In Table 4, we see the detailed results for a selection of individual categories. Here we show results for the most frequent three categories, three others occurring about 1/10 as often, three more categories occurring about 1/100 as often, and three occurring about 1/1000 as often. It is immediately clear that the best results are in the most frequent categories and that these account for the relatively high micro-average. Some less frequent categories, e.g. "sports", do have very good categorization results and others, like "war, civil war" are fair, but many of the others are quite poor. In the case of the linear method, the results we see in Table 4 are generated with default parameter settings. Notice also that for less frequent categories, the decision tree gives a better result than the linear method. It is possible that this is only due to the optimization algorithm used for linear classifier training since the procedure we currently use has not been optimized for problems with skewed class distributions. Further experiments are necessary to completely understand this phenomenon.

A number of interesting questions are suggested by the entries in this table. One of these is why recency matters less for less frequent categories. A possible explanation is that news topics tend to occur in bursts, so training data for a particular category is localized and may not have appeared recently. While plausible, it would take detailed study to verify that this is indeed the case. We did

decide to investigate one of the many questions more closely. Some topics seem to be inherently cohesive and it seems that it should be easy to build a classifier for them. As an example, we looked at the rule set generated by the decision tree classifier for the category "religion" and did an analysis of classification failures for the first forty or so errors, and have the following comments:

- As is often the case in newswires, stories which are substantially identical are sent over the wire numerous times. Some of these stories are summaries with several topics. If one of these is related to religion but is only a brief mention, the classifier will likely not find it and because the story appears several times, what is really a single mistake appears as multiple mistakes.

- Some stories may have religion as only a subtopic of some other theme. For example, there are stories in the collection regarding human rights in China in which religion is one of the rights. This story and minor variants is also repeated, leading to multiple errors when not classed as "religion" by the classifier.

- In some cases, the classifier did not generate a rule which would have covered a number of cases. For example, there is no rule regarding multiple occurrences in a story of the word "religion", which humans might expect to be reasonable. Apparently many stories, which are not categorized as religion, also contain words typically associated with religion. In fact, only about half the occurrences in the test set of the word "religion" are in stories categorized as "religion". The remainder are in a number of different categories, the most frequent being "international relations" and "domestic politics".

6 Conclusion

The Reuters-810000 collection is nearly 40 times larger than the previous collection. The earlier version served as a source of benchmarks in text categorization. Using sparse-feature methods, we readily processed this huge collection of labeled news stories. Results are competitive with the best results for the smaller collections. Our results provide an initial benchmark for future comparisons and improvements. Whether these results apply outside the newswire domain will require further experimentation on other labeled document collections.

The larger collection of documents increases dimensions in many directions, including a much

larger feature or word space, and an increased sample size for most topics. Operating in this much larger feature space, we found that the simplest methods were effective. For example, we did not have to do any special processing to create a dictionary; rather we just gather all words in all training documents. For our sparse-data machine-learning methods, the increase in the number of words is of minor consequence. Word lookup is by hash table, and only a small percentage of words occur in any document. For this application, words with numerical content had no additional predictive value. For other applications, such as those containing specialized part numbers, they may be of critical importance.

It is clear that the amount and recency of training data needed to build a classification model for newswire texts is dependent on the model. A linear model is much less sensitive to the recency effect and requires less training data, as well as being more accurate overall as compared to a decision tree. In the case of less frequent categories, however, a decision tree can be more accurate. We made no attempt to train differently for any category. It may very well be the case that results will improve when training procedures are modified to take into account highly skewed populations for rare topics.

References

See web site for details. http://about.reuters.com/researchandstandards/corpus/available.htm.

(2001). National institutes of standards and technology. Text Retrieval Conference.

Apte, C., Damerau, F., and Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12:233–251.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management*, pages 148–155.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In *European Conference on Machine Learing, ECML-98*, pages 137–142.

Johnson, D. E., Oles, F. J., Zhang, T., and Goetz, T. (2002). A decision-tree-based symbolic rule induction system for text categorization. *IBM Systems Journal*, 41:428–437.

Lewis, D. (1992). Text representation for text classification. In Jacobs, P., editor, *Text-Based Intelligent Systems*. Lawrence Erlbaum Publisher.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Weiss, S., Apte, C., Damerau, F., Johnson, D., Oles, F., Goetz, T., and Hampp, T. (1999). Maximizing text-mining performance. *IEEE Intelligent Systems*, 14:63–69.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, 1:69–90.

Zhang, T. (2002). On the dual formulation of regularized linear systems. *Machine Learning*, 46:91–129.

Zhang, T. (2003). Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statitics*. to appear.

Zhang, T. and Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31.