Feature Selection Using Linear Support Vector Machines

Janez Brank
Marko Grobelnik
Nataša Milić-Frayling
Dunja Mladenić

12 June 2002

Technical Report
MSR-TR-2002-63

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Feature Selection Using Linear Support Vector Machines

Janez Brank, Marko Grobelnik, Dunja Mladenić
Jožef Stefan Institute, Ljubljana, Slovenia

Nataša Milić-Frayling
Microsoft Research, Cambridge, United Kingdom

1 Introduction

Trends towards personalizing information services and client-based applications have increased the importance of effective and efficient document categorization techniques. New user scenarios, including the use of devices with various specifications, have placed higher demand on performance considerations and computing resource management when designing and testing classification techniques. It is that aspect of text classification that led us to explore methods for training classifiers that optimally use the computing memory and processing cycles for the available training data. In particular, we consider tradeoffs between the quality of document classification, as measured by commonly used performance measures, and reductions of a feature set used to represent the data.

The timing of our study coincides with the availability of a larger data collection for text classification research provided by Reuters [Reu00]. While most of the past research has been constrained by a limited availability of data sets suitable for experimentation, the new Reuters collection provides opportunities for numerous experimental designs. It enables researchers and practitioners to aim at designing and testing systems that can meet the requirements of real life operations. Indeed, while the popular, smaller Reuters collection [Lew98] contained less than 30,000 documents in total, requiring less than 30 MB of disk storage, the new collection contains more than 800,000 documents and amounts to over 2 GB of text data.

In this study we propose a method for feature selection and iterative classifier training based on Support Vector Machines (SVM) with linear kernels. We explore how this and other feature selection methods can be used to make tradeoffs between the amount of training data and the sparsity of the document representation for the fixed amount of system memory.

Our experimental results show that the SVM-based feature selection provides a suitable way of preserving the classification performance while significantly reducing the size of the feature space and increasing the sparsity of data.

In the following sections we first describe the proposed feature selection and classifier training method pointing to relevant issues. We give a brief overview of feature selection methods used in the past research, describe the experiments in which we compare the effectiveness of these methods in the context of the proposed training strategy, and present the experimental results. We conclude a brief summary and the outline of future research.

2 Training with SVM Feature Selection

In situations where there is an abundance of training data it is conceivable that the training of classifiers cannot be performed over the full set of data due to limited computing resources. Thus a question arises how the training of a given classifier can still be done so that the full training set is taken into account. Here we present a simple procedure that has proven quite effective, as our experimental results show (see Section 5).

Our strategy is first to train linear Support Vector Machines (SVM) on a subset of training data to create initial classifiers. In this model each classifier is really a hyperplane separating ‘positive’ and ‘negative’ examples for the class (i.e. documents belonging to this class from those not belonging to it), and can be represented by the normal (i.e. a vector perpendicular to it) and a threshold.

The second step involves eliminating features from the normal that have low weights in order to achieve a specified level of data sparsity. Sparsity is here defined as the average number of non-zero components in the vector representation of data or, in other words, the average number of terms left in documents after some terms have been discarded.

Finally, using only features retained after the feature selection step, we create a representation of the full training set of documents. We retrain the linear SVM classifier in the fixed feature space and use the final normal to classify the test data.

This method is designed to take advantage of the memory freed as a result of increased data sparsity and include larger training sets while keeping the memory consumption constant. At the same time, we calibrate the performance controlling the possible negative impact of the reduced feature space.

In principle, the feature selection phase could include a separate validation set to suggest possible tradeoffs between sparsity and classification accuracy; alternatively, memory limitations may dictate what level of sparsity is necessary. However, here we do not investigate these decision-making aspects but instead just experiment with different levels of sparsity and observe their influence on the performance after feature selection was applied.

We should point out that in the current implementation we applied no linguistic processing to the document text. Documents were represented using single word features, weighted by the standard TF-IDF score[1] [SB88] and retaining only those that occur in at least 4 documents in the whole training set.

Intrigued by the efficacy of the SVM-based feature selection, we explored how it compares with other feature selection methods when used in conjunction with the proposed training strategy. When assessing feature selection methods we look at both the final classification performance of the resulting classifiers and the sparsity of the training data representation. In the following section we give a brief overview of the considered feature selection methods and refer to research related to our work.

3 Related Work

3.1 Feature Selection Methods

In text categorization, feature selection (FS) is typically performed by sorting linguistic features according to some weighting measure and then setting up a threshold on the weights or simply specifying a number or percentage of highly scored features to be retained. Features with lower weights are discarded as having less significance for the classification decision. Experiments then typically evaluate the effect that feature selection has on the classification performance.

Numerous feature scoring measures have been proposed, e.g., information gain, odds ratio, χ^2, term strength, etc. Even the simple document frequency has been found to perform well in conjunction with the k Nearest Neighbor method, as measured by the 11-point average precision [YP97, MG99].

It is important to note that feature weighting and selection can be more or less coordinated with the classification model, in the sense that they may be governed by the same or distinct theoretical models. It has been a common practice to explore the impact of various FS methods in combination with different classification methods. For example, an approach for weighting features based on SVM has been proposed in [GF01], where SVM is used with a special kernel to learn the weights of features for use with a Naive Bayes classifier. On the other hand, there were conscious attempts to create FS methods that are compatible with the feature scoring of a particular classifier. In that respect feature scoring using *odds ratio* is seen as a good match for the Naive Bayes classifier and has been shown to improve its performance [Mla98b].

While the idea of preserving the integrity of a theoretical framework is attractive, it is not feasible unless the framework includes an inherent mechanism for feature selection. This is typically not the case with classification methods (more specifically, it may often be possible to obtain a feature ranking or weighting, indicating which features may be preferable over others, but without clear guidelines on how many features to keep) and thus we are inevitably guided by external factors when designing the selection criteria — in our case that is the *targeted level of sparsity* of the data representation.

We compare the effect of feature selection based on an SVM normal with two feature scoring measures that have been reported successful in text categorization: *information gain* and *odds ratio*. These two

[1] There are several variants of the TF-IDF score. We use the following formulas: $\text{IDF}(t) = \log(1+N/\text{DF}(t))$ where $\text{DF}(t)$ is the "document frequency" of term *t*, i.e. the number of documents in the training set that contain the term *t*, and *N* is the total number of documents in the training set. The weight of term *t* in document *d* is then defined as $\text{TF}(t,d) \cdot \text{IDF}(t)$, where is $\text{TF}(t,d)$ the "term frequency", which is simply the number of occurrences of term *t* in document *d*.

FS methods are interesting also because they seem to prefer different types of features: the information gain is known for its tendency to prefer common features over extremely rare ones, while the odds ratio ranks rare features highly as long as they are exclusively characteristic of positive class instances. Here we provide a brief description of the three FS methods.

3.1.1 Information gain

The information gain score (IG) of a term t is calculated using the following formula:

$$\mathrm{IG} = \sum_c \sum_{\tau\in\{t,\neg t\}} \mathrm{P}(c,\ \tau) \log[\mathrm{P}(c,\ \tau)/\mathrm{P}(c)\mathrm{P}(\tau)]$$

where c ranges over the classes (in our case, where a document might belong to several categories, and we treat each category as an individual two-class problem, c can be either *positive* or *negative*) and τ ranges over the values t and $\neg t$, referring to whether the term t under consideration is present in a document or not.

This formula is equivalent to $\mathrm{H}(C) - \mathrm{H}(C|T)$ where

$$\mathrm{H}(X) = \sum_x \mathrm{P}(X=x) \log \mathrm{P}(X=x)$$

is the entropy.

It therefore measures how much we learn about C by knowing T, since $\mathrm{H}(C|T)$ measures how much remains unknown about C if we know T; hence the term "information gain".

On the other hand, it is also equivalent to $[\mathrm{H}(C) + \mathrm{H}(T)] - \mathrm{H}(C, T)$, which can be interpreted as the amount of information that each of these variables contains about the other one. For that reason this measure is also known as (*average*) *mutual information*. It should not be confused with $\log[\mathrm{P}(t|c)/\mathrm{P}(t)]$ (or the maximum or average of this value over all c), which is also sometimes called mutual information.

Information gain has been found to work well with the k-nearest neighbor algorithm on the small Reuters collection and the OHSUMED collection [YP97], where an increase in the 11-point average precision (from 0.879 to 0.892) has been observed while merely 2% of features were kept.

3.1.2 Odds ratio

The odds ratio score of a term t is calculated as follows:

$$\mathrm{OR} = \log[\mathrm{odds}(t|pos)/\mathrm{odds}(t|neg)]$$

where $\mathrm{odds}(t|c) = \mathrm{P}(t|c)/(1-\mathrm{P}(t|c))$, c denotes a class c and pos and neg refer to the number of positive and negative examples in the training data, respectively.

This measure gives a high score to features typical of positive documents and a low score to those typical of negative documents. One possible disadvantage of this scoring method is that features which occur in very few positive documents get very high scores as long as they do not occur in any negative documents. In this manner rare rather than representative features of the positive documents obtain high scores.

Odds ratio was used as a feature selection method (in combination with Naive Bayes as the training algorithm) for categorizing web pages in a collection specially built for profiling web usage [Mla98a, Mla98b] and for classifying Yahoo data into Yahoo categories [Mla98b, MG99]. In the latter case, an increase in the F_2 measure (from 0.13 to 0.6) was reported when only 0.2% and 5% of the original features were kept.

3.1.3 Normal-based feature selection

In this study we use the Support Vector Machine with linear kernels. Data instances are described by vectors $\mathbf{x}_i = (x_{i1},\ldots,x_{id})$, where d represents the dimensionality of the feature space, i.e., the number of distinct features in the model. In general, the class predictor trained by SVM has the form

$$prediction(\mathbf{x}) = \mathrm{sgn}[b + \textstyle\sum_i \alpha_i \mathrm{K}(\mathbf{x}, \mathbf{x}_i)]$$

but in the case of a linear kernel $\mathrm{K}(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\mathrm{T}\mathbf{z}$ this can be rewritten as

$$\mathrm{sgn}[b + \mathbf{w}^\mathrm{T}\mathbf{x}] \quad \text{for} \quad \mathbf{w} = \textstyle\sum_i \alpha_i \mathbf{x}_i$$

where the vector of weights $\mathbf{w} = (w_1,\ldots,w_d)$ can be computed and accessed directly. Geometrically, the predictor uses a hyperplane to separate the positive from the negative instances, and $\mathbf{w}$ is the normal to this hyperplane.

The linear classifier categorizes new data instances by testing whether the linear combination $w_1x_1 + \ldots + w_dx_d$ of the components of the vector $\mathbf{x} = (x_1,\ldots,x_d)$ is above or below some threshold $-b$ (possibly 0). In our feature selection approach we use the absolute value $|w_j|$ as the weight of a feature j. We retain features for which the value of $|w_j|$ exceeds the threshold value that is obtained from the data sparsity criteria. This type of feature weighting seems intuitively appealing because features with small values of $|w_j|$ do not have a large influence on the predictions of the classifier based on $\mathbf{w}$; this can be seen as meaning that these features are not important for classification purposes, and that consequently they could be dispensed with in the training phase as well.

A theoretical justification for retaining the highest weighted features in the normal has been independently derived in a somewhat different context in [SBR01]. The idea here is that one may consider a feature important if it significantly influences the width of the margin of the resulting hyper-plane; this margin is inversely proportional to $\|\mathbf{w}\|$, the length of $\mathbf{w}$. Since $\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i$ for a linear SVM model, one can regard $\|\mathbf{w}\|^2$ as a function of the training vectors $\mathbf{x}_1,\ldots,\mathbf{x}_l$, where $\mathbf{x}_i = (x_{i1},\ldots,x_{id})$, and thus evaluate the influence of feature j on $\|\mathbf{w}\|^2$ by looking at absolute values of partial derivatives of $\|\mathbf{w}\|^2$ with respect to x_{ij}. (Of course this disregards the fact that if the training vectors changed, the values of the multipliers α_i would also change, but the approach nevertheless seems appealing.) For the linear kernel, it turns out that

$$\sum_i |\partial\|\mathbf{w}\|^2/\partial x_{ij}| = k\,|w_j|,$$

where the sum is over support vectors and k is a constant independent of j. Thus the features with higher $|w_j|$ are more influential in determining the width of the margin. (The same reasoning could also be applied if a non-linear kernel was used, because $\|\mathbf{w}\|^2$ can still be expressed using only the training vectors $\mathbf{x}_i$ and the kernel function.)

In this study we also use the linear SVM classifier as the classification model since it has been shown to outperform most of other classification methods [DPHS98, Joa98]. Interaction of the normal-based FS with other classification methods will be subject of our future work.

Note that the normal-based approach to feature weighting and selection involves an important issue: the selection of a set of instances over which one trains the normal $\mathbf{w}$ that defines the feature weighting. Since training an SVM model requires a considerable amount of CPU time, and also practically requires all the training vectors to be present in main memory all the time, it is likely to be undesirable, or even unfeasible, to use the entire training set at one's disposal to train the normal for feature selection. Furthermore, this would be justifiable only if there is a significant gain in the classification performance or general system performance (e.g., speed, computing resource consumption, etc.) because of the cost incurred due to the retraining of the classifiers. Thus we explore the effectiveness of this method in the scenarios where one might be constrained to train on a subset of the training set.

More precisely, we use the feature weighting defined by the model trained over a subset of data to choose which features to discard. After the removal of a large number of features both the time and the memory requirements for processing the entire training set are smaller and thus make that task feasible. Of course, having used only a subset of data to obtain the weighting and selection of features may mean that this approach performs worse as a feature selection method than others which take into account the whole data set. In our experiments we will also explore the relationship between the size of the subset used to train a normal and the performance of the feature weighting based on that normal.

4 Experiment Description

4.1 Experimental strategy – Rationale

Since our main consideration is the classification performance under the constraint of limited computer resources we experiment with methods that vary the size of the training data and the sparsity level of the data representation. We compare the approach of using a smaller number of documents with a richer feature set and the alternative that involves applying a feature selection method to increase data sparsity so that a larger number of training data can be used within the fixed feature set. Recall that the sparsity is here defined as the average number of non-zero components in the vector representation of documents in the given document set, in this case the data used for training.

Our first set of experiments thus explores the relationship between feature reduction and sparsity achieved. The second set investigates the robustness of the classification method under the fixed feature set constraint: what is the impact on the classification performance if the classifiers are trained over a various data sets and the feature set is kept constant.

4.2 Data Processing and Analysis

4.2.1 Training and test data

For experimentation we used the Reuters-2000 collection [Reu00], which includes a total of 806,791 documents, with news stories covering the period from 20 Aug 1996 through 19 Aug 1997. We divided this time interval into a training period, which includes all the 504,468 documents dated 14 April 1997 or earlier, and the test period, consisting of the remaining 302,323 documents.

We used the documents from the entire test period as test data but for training we constructed a subset of data, here referred to as *Train-1600*, in the following way: for each Reuters category that contains 1600 or more documents in the training period (i.e., positive training examples), we randomly selected exactly 1600 documents. For categories with fewer than 1600 positive examples in the training period we took all such documents.

The rationale behind this decision was to select a set of training data that can be used for multiple purposes, including the experiments in which the number of positive examples per category could be fixed across categories in order to understand the effect that parameter has on the classification performance. Although the current experiments do not control that variable, the future experiments along these lines will be comparable with ones presented in this study.

Fortunately, the training set constructed in this way turns out to be also a good representative sample of the whole data set, as can be seen from Figure 1, which shows the comparison of category distributions in the *Train-1600* sample and the whole corpus.

Indeed, during sampling we treated categories as independent from each other, thus ignoring the hierarchical nature of the Reuters classification scheme and the fact that a document may be labeled by a number of distinct categories. As a result, the union of documents contains more than 1,600 positive examples for those categories whose labels typically co-occur with other categories. As can be seen from Figure 1, the achieved category distribution generally follows the trends of the full data set with only a slight modification: categories with very high distributions in the whole corpus get a slightly smaller relative representation while categories with fewer positive examples get a larger representation in the sample in comparison to the full corpus. It is interesting to note from the graph that, in fact, for two large categories (*ecat* and *gcat*), the proportion has increased because of the wide spread use of the categories.

In our experiments we use the 'natural' distribution of labeled documents in the *Train-1600* set. In other words, we did not restrict training to the 1600 positive examples originally selected for a given category. We used all the documents that carry the category label, thus including all those that were contributed to the sample by other categories.

For the experimentation purposes we further selected sub-samples of the *Train-1600* set (of 118,924 documents) of size one half, referred to as *Train-$^1/_2$*, one quarter (*Train-$^1/_4$*), one eighth (*Train-$^1/_8$*), and one sixteenth (*Train-$^1/_{16}$*) of the full training set, respectively. Random selection[2] of these subsets was strictly based on the target sample size, rather than the distribution of positive examples of individual categories. However, as Figure 1 shows, the natural distribution is followed for most of the categories and the desired distribution effects for small and large categories are achieved.

[2] Subsets of the *Train-1600* are prepared as follows: in the list of 1600 representatives for each category, each example is given a random number between 0 and 1, and is kept in the subset if this number is less than some threshold *T*. Thus, a document proposed as a representative of several categories also has a greater chance of being kept in the subset. The threshold *T* was chosen so as to yield the desired number of documents in the new sample. For example, a threshold of approximately 0.45 was used to obtain the 59,462 documents for the sample *Train-$^1/_2$* containing half the documents of *Train-1600*. Note that, as a consequence, small categories may be poorly represented in the smaller subsets. For example, *e312*, with 32 documents in the training period, has all these documents included in *Train-1600*, but only one has made it into *Train-$^1/_{16}$* (however, we did not include any such extremely small categories in our experiments, and it is therefore unclear how these sampling anomalies would affect the performance).

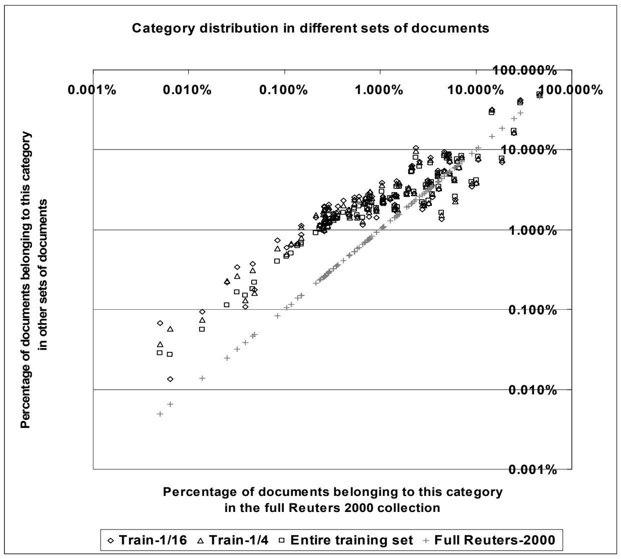


Figure 1

4.2.2 *Data Representation*

We represent documents using the bag-of-words approach, applying a stop-word filter (from a standard set of 523 stop-words) and ignoring the case of the word surface form. Features that occur less than 4 times in *Train-1600* are removed and the remaining features are weighted using the TF-IDF score and normalized so that each vector representing a document has unit length.

In order to experiment with the sparsity of document representations, we set an appropriate threshold on the number of features retained in order to obtain document vectors with a desired average number of nonzero components, i.e., desired sparsity. For each combination of the target category and the number of features to be kept, temporary copies of document vectors are made with the unwanted features removed; these copies are used for training. Finally, the models obtained during the training are used to classify the test documents. The test documents are represented using only the features kept after feature selection.

We used the SvmLight program (version 3.50) by Thorsten Joachims for SVM training and classification [Joa99]. The program is particularly appropriate for our experiments because of its optimizations for working with linear kernels. It computes the normal vector **w** explicitly rather than working entirely with the dual representation $\Sigma_i \alpha_i y_i \mathbf{x}_i$ as is necessary for other kernels.

4.2.3 *Category selection*

Training classifiers for all 103 Reuters categories over relatively large sets would be a time consuming and process intensive task. Therefore we restricted our study to a sample of 16 categories. These categories were selected based on the results of a preliminary experiment that involved training the SVM over a smaller training set *Train-200* for the complete set of Reuters categories.

The set *Train-200* (19,213 documents) was constructed in exactly the same way as the *Train-1600* except that we used only 200 documents per category. The test set was also constructed in the same way, except that documents from the test period were used, and that only 100 random representatives of each category taken; this resulted in a sample of 9,596 test documents, referred to as *Test-100*.

The selection process was based on two characteristics of the categories: the distribution of positive examples in the whole corpus and the break-even point (BEP) achieved for the category by SVM, with *Train-200* and *Test-100*. In order to obtain a sample representative of the whole set of categories, we created a 2-D histogram (with the log scale for the document distribution and 0.2 interval partition of the BEP range) and selected a subset that follows approximately the statistical distribution of the 103 category set. The 16 categories chosen include: *godd*, *c313*, *gpol*, *ghea*, *c15*, *e121*, *gobit*, *m14*, *m143*, *gspo*, *e132*, *e13*, *c183*, *e21*, *e142*, and *c13* (see Figure 2 and Appendix B for more details).

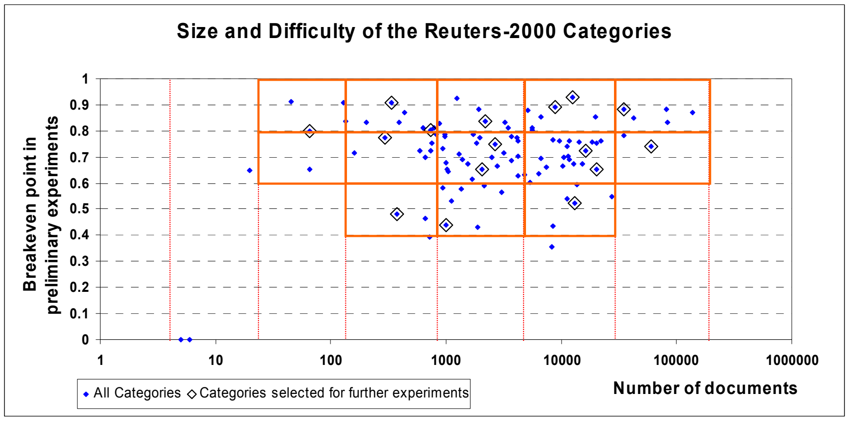


Figure 2

4.2.4 Comparison of Two Classifiers

We have partitioned the test data set into ten disjoint subsets and recorded performance measures of the classifiers, such as F_1, separately for each subset. Thus, to compare two classifiers, we can perform a paired t-test on the corresponding two groups of ten F_1 values. This is done both for micro- or macro-averaged F_1 values.

Alternately, one may perform a separate t-test for each category and then count how many times one classifier has been found to be significantly better than the other, or vice versa. These values can be quite informative, but the downside is that the comparison of two methods is now described by two numbers, rather than a single confidence value as would be obtained from a single t-test. In principle, one could reach a single confidence value by applying a sort of sign test (along the lines of the "macro sign test" from [YL99]).

Still another option is to compute, for each category, the average F_1 over the entire test set, resulting in a sequence of as many values as there are categories; then one can compare two methods by taking a paired t-test over the corresponding two sequences. This approach (referred to as a "macro T-test" in [YL99]) has been often used, although it has also been criticized because it treats performance values on different categories as random samples from a normal distribution [Lew92].

In our experience, all types of tests tend to give similar results; however, the category-paired t-test is slightly more conservative (i.e., less likely to declare differences to be significant) than the t-test on macroaveraged F_1-values, and the t-test on microaveraged F_1-values is much more liberal in considering differences to be significant. For the remainder of this paper, we will report significance results from the t-tests based on macroaveraged F_1 values.

5 Experiment Results

5.1 Sparsity and the number of features

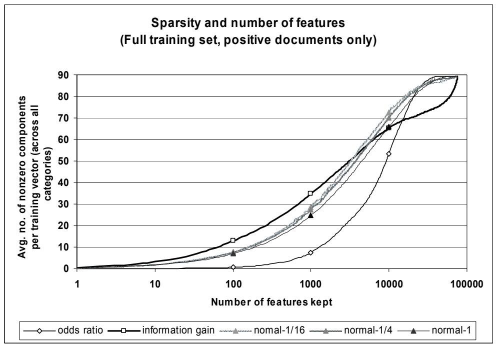
Most of the existing research has focused on the reduction of the number of features (i.e., reduction of feature space dimensionality) rather than sparsity of the resulting vectors (i.e., reduction of memory requirements for vector storage). It is interesting to explore the relationship between these two data representation aspects.

It is expected that by reducing the number of features to a certain percentage of the initial set one will increase the sparsity of vectors. However, for a fixed percentage of features to be retained, various feature scoring methods yield significantly different levels of vector sparsity.

Figures 3 and 4 show a comparison of the levels of sparsity achieved by the odds ratio (OR), information gain (IG) and normal-based feature selection on the positive and negative documents in our training data, respectively. For each of these methods we create a ranked list of features, according to the score associated by the method, and plot the curve that shows the sparsity (i.e., the average density) of the vector representation of documents in a given document set as various numbers of top ranked features are used in the vector representation of documents. In principle, for any given list of ranked features and a set of documents that contain these features we can calculate the corresponding sparsity cruve.

Appendix C contains a table of statistics showing the relationship between the number of features retained and the level of sparsity for the three methods we consider here. In the following charts, as well as in later sections, we will discuss several variants of the normal-based feature weighting, depending on how large a set of documents the normal in question has been trained on. If the full *Train-1600* training set is used to obtain the normal for feature selection, we call the resulting weighting *normal-1;* if *Train-$^1/_k$* is used, the resulting weighting is denoted by *normal-$^1/_k$*. For each of these weightings we study the corresponding sparsity curve.

Of course, each category has its own feature weightings and the actual level of sparsity attained if a fixed number of features is kept will vary from one category to another. The values shown in these charts and discussed below are averages across all categories.



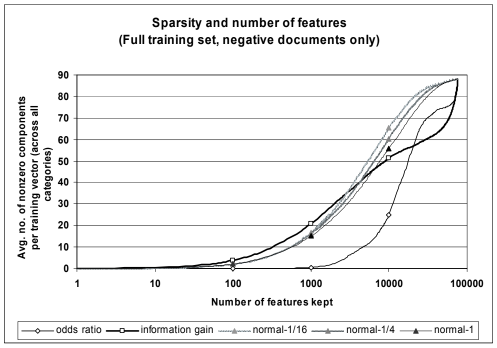


Figure 3

Figure 4

5.1.1 Odds ratio sparsity curves

As can be seen from the two charts, OR has the greatest preference for features that are not commonly present in the documents of the collection. In fact, our analysis of the *Train-1600* shows that OR needs to retain almost 1300 features to reach an average of one nonzero component per training vector. On the other hand, IG requires only 20 features to achieve the same.

Comparing OR curves from the two charts we see that the preference for 'rare' features is particularly pronounced for features typical of positive documents as the OR curve in Figure 3 ascends much sooner and more rapidly. For large numbers of features kept, the curve levels off since at that point the number of nonzero components in positive documents hardly increases, as the features added then are present almost exclusively in negative documents. Indeed, OR assigns lowest scores to features present in negative documents only. This results in the ascent that begins in the OR curve of Figure 4 after approximately 60,000 features are added. Indeed, by inspection of the feature set, we find that the remaining features practically always have ten or more occurrences in negative documents but none in the positive ones.

5.1.2 Information gain sparsity curves

IG seems to be the most inclined to rank highly features that are common in the document set. It is, however, interesting to note the fast ascents at the right end of the curves in Figures 3 and 4. They essentially show that adding features from the very bottom of the scored feature list increases the average density of vectors rapidly, indicating that these are rather common features but with low IG scores. As expected based on the IG formula, manual inspection shows that these are features that are equally common in positive and negative documents, thus with $P(t|pos) \approx P(t|neg)$.

5.1.3 Normal-based sparsity curves

We observe that the normal-based feature selectors have sparsity curves that fall between these two methods. While the curves for different training sets are very close to each other, a closer look reveals that normals trained on smaller subsets of training data have a stronger preference for common features than those trained on larger subsets. That is, it seems that as larger subsets of training data are used, more features, including relatively uncommon ones, receive a score that is high enough to promote the ranking of the feature and thus include it in the truncated normal.

In order to understand this phenomenon better we made a simple exploration of density curves associated with the feature weightings *normal-1* and *normal-$^1/_{16}$*. Since *normal-$^1/_{16}$* was obtained by training over a subset of *Train-1600* (called *Train-$^1/_{16}$*), and *normal-1* was obtained by training on the whole *Train-1600*, the features present in *normal-$^1/_{16}$* are also present in *normal-1*, although they may have different weights and different rank among the scored features. If we simply truncate the *normal-1* so that it excludes features which are not present in the *normal-$^1/_{16}$* (e.g., by setting their weights to 0), thus obtaining the *normal-1',* the sparsity curve for the resulting feature weighting (denoted in the following paragraphs) is steeper and comes closer to the curve of *normal-$^1/_{16}$* (see Figure 5). This shows that newly introduced features, some uncommon but highly ranked, account for a part of the curve 'slowdown'.

However, the difference between the sparsity curves of *normal-1'* and *normal-$^1/_{16}$* is still significant. As the feature set is the same for both this is purely due to the difference in relative ranking of features. The question is what influences the difference in ranking: the volume of training data (e.g., the statistical characteristics of the data set, regardless of the number of features used in the data representation) or the feature set used in the training phrase.

To explore this question we fix the feature set to the one obtained from *normal-$^1/_{16}$* (i.e., the features from *Train-$^1/_{16}$*) and train a linear classifier over the full training set *Train-1600*. Thus, the system uses the full *Train-1600* set of documents (the same that was used in training *normal-1*) but represented by only those features that were seen when training *normal-$^1/_{16}$* . The resulting weighting, called *normal-1",* can be compared with both *normal-$^1/_{16}$* and *normal-1'*, which contain the same set of features but with possibly different weights: *normal-$^1/_{16}$* was trained over a smaller subset, *Train-$^1/_{16}$*, while *normal-1'* is essentially *normal-l* truncated (after training) to include only those features that are present in *Train-$^1/_{16}$* (see Table 1).

Feature weighting	*normal-1*	*normal-1'*	*normal-1"*	*normal-$^1/_{16}$*
Training set	*Train-1600*	*Train-1600*	*Train-1600*	*Train-$^1/_{16}$*
Feature set used in training	*Train-1600*	*Train-1600*	*Train-$^1/_{16}$*	*Train-$^1/_{16}$*
Final feature set	*Train-1600*	*Train-$^1/_{16}$*	*Train-$^1/_{16}$*	*Train-$^1/_{16}$*

Table 1

Comparison of the corresponding sparsity curves for *normal-1'* and *normal-1" sh*ows almost no difference. Thus it is the amount of training data rather than the feature set that influences the difference in sparsity between these two normals.

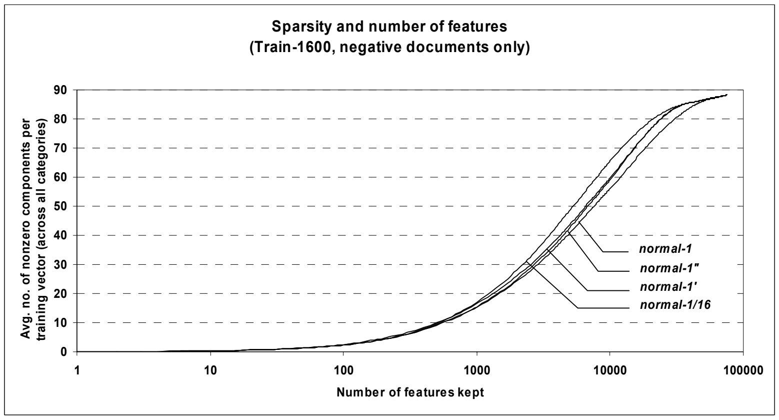


Figure 5

In summary, there are two factors that account for the differences in sparsity between normals trained over different size training sets. First, as more examples are considered, the pool of features becomes larger and many relatively uncommon but nonetheless helpful features get the opportunity to obtain a moderately high position in the ranking. (Hence the difference between *normal-1* and *normal-1'*.) Second, the statistical properties of the considered feature set change with the volume of training data used and that seems to play a significant role. Many of the less common features, even if they do appear in a smaller training set, might be largely ignored in the ranking trained on that training set only, while they achieve a much higher position in a ranking trained on a larger training set. (Hence the difference between *normal-1"* and *normal-$^1/_{16}$*.)

5.2 Dimensionality reduction and classification performance

It is also interesting to relate the reduction in dimensionality to the reduction in classification performance.

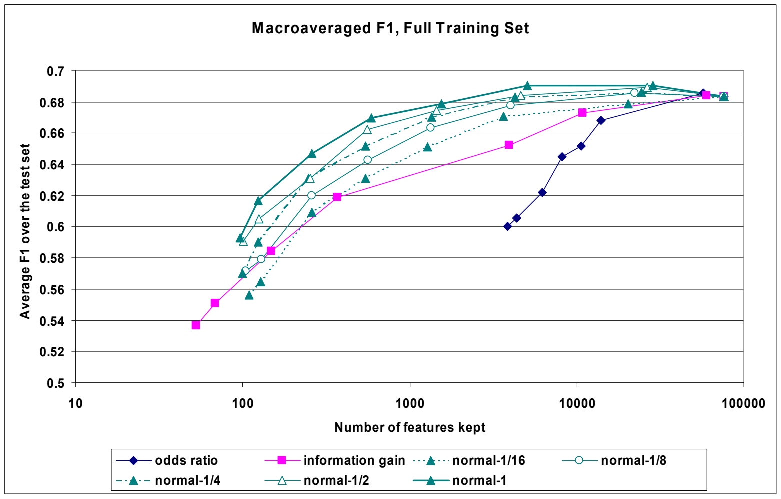


Figure 6

From Figure 6 we see that OR needs to retain one or two orders of magnitude more features to achieve a performance that is comparable to the other FS methods when used with the linear SVM classifier. This is not surprising in view of the observations we made in the previous section. Indeed, OR scores highly features that are characteristic of positive documents, among those also the ones that are 'rare' and thus do not necessarily make an impact on the document representations and the SVM training process.

In many practical situations the management of computing resources is important and thus the sparsity seems to be of high practical importance. In the remainder of the study we will focus on sparsity and use dimensionality reduction to meet the sparsity requirements.

5.3 Experiments with the fixed memory constraint

Part of the motivation for using feature selection is to describe documents with sparser vectors and thus process larger sets of documents with a given amount of memory. However, we need to establish evidence that such an approach is plausible and that it is possible to achieve an effective tradeoff between selecting a subset of features (to make vectors sparser) and selecting a subset of documents (to have fewer vectors). For example, if we have only a quarter of the amount of memory required for storing full vectors of the complete set of training data we could: (i) keep full vectors but only for 1/4 of the training documents; or (ii) keep 1/2 of the documents, and use feature selection to make the vectors twice as sparse; or (iii) keep all the documents, and use feature selection to make the vectors four times sparser; etc.

In order to explore these possibilities we evaluate the classification performance for the following type of scenarios. Let us assume that storing all N documents from *Train-1600* at sparsity[3] $S=80$ would require M units of memory ($S\cdot N=M$), and that we only have $M/2^K$ units available for some integer K. We are seeking the values S' and N' such that $S'\cdot N'=M/2^K$ and for which the classification performance is not significantly degraded. Thus, we experiment with values $S'=S/2^k$ and $N'=N/2^{K-k}$, for $k=0,\ldots,K$. Note that ideally we would vary the values of S and N on a finer scale but the current approach is already quite informative as the experiment results show (see Figure 7 and Table 1).

For a given K, we first need to obtain a ranking of features, to be used as the basis of feature selection. Since the SVM model on which this ranking will be based will have to be trained over full vectors, it can use at most $N/2^K$ documents to respect the memory constraint. Then, for a given higher sparsity level $S/2^k$, $k=0,\ldots,K,$ we select as many top features from the ranking as are necessary to achieve this sparsity level, and then train a new model over the largest training set that can fit into the available

[3] In fact, the full vectors of the training set *Train-1600* have approximately 88.3 nonzero components on average.

memory given the fact that vectors are now sparser (this is the set containing $N/2^{K-k}$ documents). Thus, for a given K, the result at sparsity 80 is based on a training set that is only half as large as the set used for the experiment with sparsity 40, only one-quarter the size of the training set used for sparsity 20, and so on.

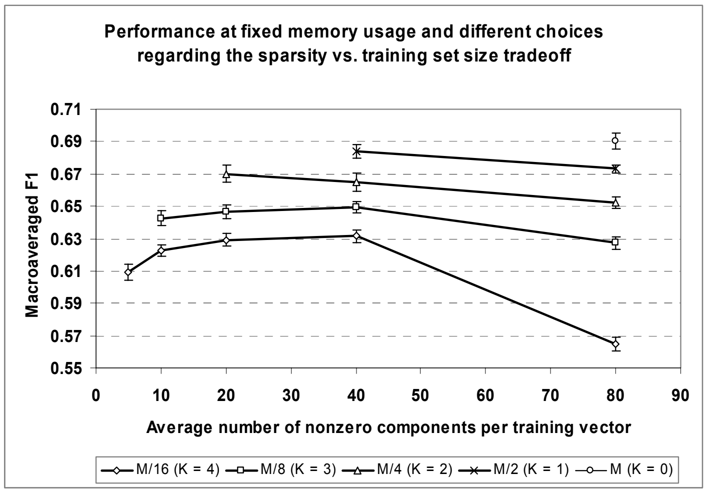


Figure 7.

Sparsity vs. Training Set Size Tradeoff								
Sparsity	K=1, Memory: $M/2$		K=2, Memory: $M/4$		K=3, Memory: $M/8$		K=4, Memory: $M/16$	
	Training Set	F_1	Training Set	F_1	Training Set	F_1	Training Set	F_1
$S = 80$	$N/2$	67.3	$N/4$	65.2	$N/8$	62.7	$N/16$	56.4
40	N	68.4	$N/2$	66.5	$N/4$	64.9	$N/8$	63.1
20			N	67.0	$N/2$	64.6	$N/4$	62.9
10					N	64.2	$N/2$	62.2
5							N	60.1

Table 2.

Figure 7 and Table 2 show the classification performance measured by the macroaverage of the F_1 measure. The error bars on the graph show standard errors of F_1 macro-averages over the 10 subsets of the test set. For comparison, if the memory constraint is removed (i.e. $K=0$) the classifier based on $S=80$ and the full training set achieves a macroaveraged F_1 of 69.0%.

The results confirm that for a fixed memory constraint (fixed K) one can achieve a better performance by applying feature selection and using a larger set of training documents than by using the full set of features but a smaller training set. For example, it is beneficial to reduce the number of nonzero components by 50% and double the number of training documents if one is working under the memory constraint (this always brought a statistically significant improvement in performance in the above experiments). That is, in some sense, expected, based on the evidence from related research that reducing the feature space typically does not dramatically affects the classification performance.

More radical reduction of the feature space (i.e., using $k>1$) generally does not bring additional improvements in performance, and eventually, when the size of the training set requires vectors to be very sparse to fit in the available memory, performance may deteriorate. This is not surprising since in this case the original set of documents used to generate the initial normal may be too small to provide useful terminology and statistics for effective feature ranking. The fixed feature set is simply not rich enough to allow generalization of the classifier to the test data.

Finally, the performance at different values of K confirms that having twice as much memory practically always allows one to achieve better performance, either by increasing the number of training vectors or by keeping more features and making the vectors less sparse.

5.4 Experiments at fixed sparsity levels

In these experiments we use *Train-1600* as the training set and different feature scoring methods to achieve a desired level of sparsity, i.e., a desired average number of nonzero components per training vector. In particular we look at sparsity of 2, 2.5, 5, 10, 20, 40, and 80 features per document as well as the full document vectors which have about 88.3 nonzero components per document on average.

For feature scoring methods we use Odds Ratio (OR), Information Gain (IG), and ranking based on SVM normals: *normal-1*, *normal-*$^1/_2$, *normal-*$^1/_4$, *normal-*$^1/_8$, and *normal-*$^1/_{16}$. The methods *normal-*$^1/_k$ use SVM normals obtained from the set *Train-*$^1/_k$, containing $^1/_k$ of the documents in the full training set (*Train-1600*); after feature selection, the final linear SVM model is trained on the full training set but with the reduced set of features. This allows us to see how robust SVM classifier is with respect to the fixed feature sets obtained from smaller amounts of training data.

Figure 8 contains graphs that show the SVM classification performance based on four measures: macro-averages of F_1, break-even-point (BEP), precision, and recall. The test set has been divided into ten folds, and the macroaveraged value (i.e., average over all categories) of each performance measure has been calculated separately for each fold. The average of this over all ten folds is then shown in the charts. (Thus, references to "average precision" in the charts should not be confused with average 11-point precision, another popular performance measure, which however we have not employed in these experiments.)

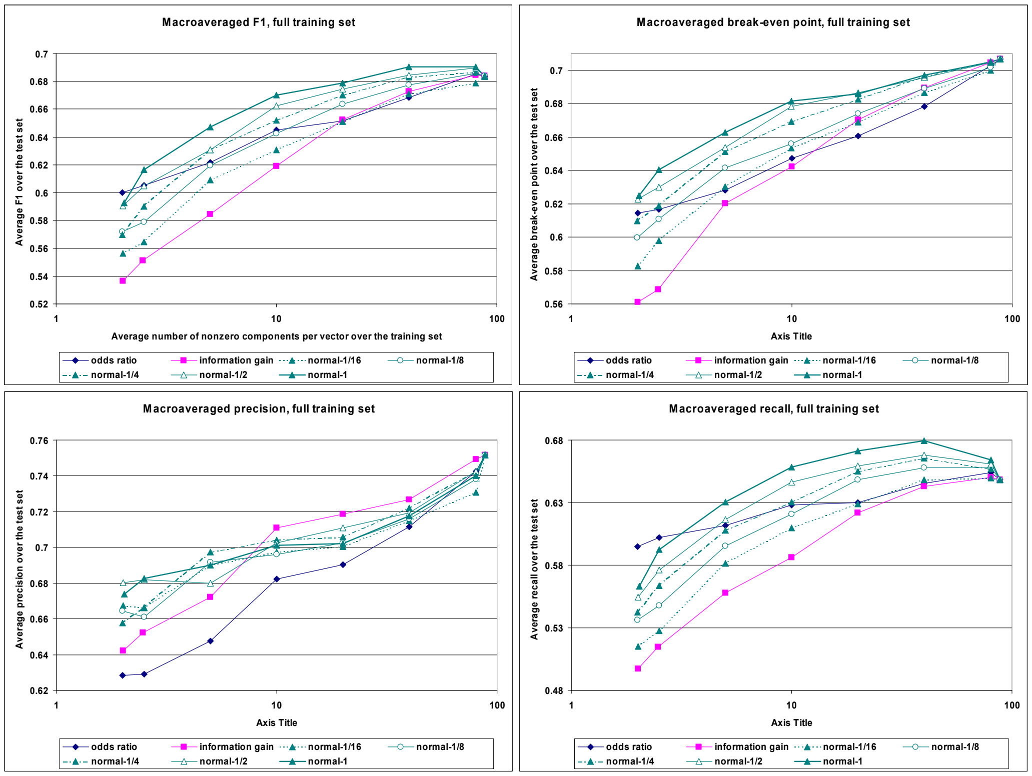


Figure 8.

5.4.1 *Comparison of the normal-1 ranking with Odds Ratio and Information Gain*

From the perspective of F_1 and BEP measures, SVM-based ranking of features is most effective for the purpose of feature selection. The performance is dominated by this method for all levels of sparsity, except when only a couple of features per document are used. OR seems to perform better in that scenario according to the F_1 measure for $S=2$. Similarly, OR provides better recall for $S\leq 2.5$.

This is not surprising since OR retains a much larger set of features than the normal to achieve the same sparsity level (see Figure 3). Furthermore, it focuses on features that are characteristic of positive examples, which in turn enables SVM to capture the positive examples more easily. However, for the same reason, the precision of classification based on OR selection suffers, as can be seen from the precision graph. Information gain, on the other hand, ensures a consistently better precision than *normal-1* for all sparsity levels $S>5$ (these differences are statistically significant at $S=20$ and 80).

5.4.2 *Comparison of normal-$^1/_k$, k=1,2,4,8,16*

SVM normals obtained from larger data sets have a clear advantage on all performance measures except for precision. Indeed they all seem to perform similarly according to that measure. Thus, if one is concerned with precision only one need not hesitate to fix the feature set to the one obtained from a smaller set of data and then retrain. Using more data in the initial phase introduces features that definitely help recall but do not affect precision. Normal-based selection seems to pick the features important for precision even from smaller sets of data.

5.4.3 *Comparison of Odds Ratio and Information Gain*

Similar to our earlier observations, the tendency of OR feature selection to prefer rare features present in positive documents causes OR to perform better than IG on recall but consistently worse according to the precision for all levels of sparsity. The trade-off between recall and precision for both methods, as seen from F_1 and BEP measures, is such that IG outperform OR for sparsity levels above 20 features per document on both of these measures.

5.5 Effectiveness

Although the above results show that feature selection generally does not improve performance of the linear SVM classifier, we can see that the normal-based feature selection with sparsity $S=80$ does produce a statistically significant improvement over the performance of vectors with the complete feature set which corresponds to $S=88.3$.

Feature selection method	Number and percentage of features retained at $S = 80$		Macroaveraged F_1 at $S=80$	P-value from t-test
Odds ratio	57,197	75.42 %	0.6857 ± 0.0053	0.881
Information gain	60,234	79.42 %	0.6843 ± 0.0047	0.685
Normal-$^1/_{16}$	20,316	26.79 %	0.6789 ± 0.0050	0.882
Normal-$^1/_8$	22,274	29.37 %	0.6859 ± 0.0048	0.901
Normal-$^1/_4$	24,338	32.09 %	0.6864 ± 0.0050	0.949
Normal-$^1/_2$	26,303	34.68 %	0.6894 ± 0.0050	**0.998**
Normal-1	28,638	37.76 %	0.6904 ± 0.0050	**0.997**
Full feature set	75,839	100.00 %	0.6837 ± 0.0049	

Table 3. Linear SVM performance with different feature selection methods at S = 80

The test set has been split into ten folds and the macro-average of F_1 computed for each fold. Averages and standard errors across these ten folds are shown. The last column shows the p-value from the t-test comparing the performance of the linear SVM classifier for each FS method at $S=80$ with the performance for full document vectors ($S=88.3$). We observe that at $S=80$ the performance is never significantly worse and for *normal-$^1/_2$* and *normal-1* it is significantly better than the performance with full vectors. A more systematic analysis of this phenomenon, e.g., exploration of the effects of sparsities $S=85$, 75, 70, etc., will be subject of future research.

Note that the number of features that needs to be retained to achieve a desired level of sparsity varies from category to category since each classifier gives rise to a different feature ranking. Shown here are averages across the 16 categories considered.

The statistics in Table 3 show that the normal-based feature selection discards as many as 60% to 75% of features to achieve the sparsity of $S=80$. In some cases, even greater sparsities can be reached without significantly decreasing the performance. This can be seen in Table 4, which shows that *normal-1* at sparsity 40 actually still performs significantly better than full vectors, while the performance of *normal-$^1/_2$* and *normal-$^1/_4$* at sparsity 40, and that of *normal-1* at sparsity 20, is not significantly different from the performance of the full feature set.

Feature selection method	Sparsity	Number and percentage of features retained		Macroaveraged F_1	P-value from t-test
Normal-$^1/_4$	40	4278	5.65 %	0.6831 ± 0.0056	0.591
Normal-$^1/_2$	40	4628	6.11 %	0.6842 ± 0.0042	0.564
Normal-1	20	1545	2.04 %	0.6786 ± 0.0037	0.876
Normal-1	40	5079	6.70 %	0.6905 ± 0.0045	0.970
Full feature set	88.3	75839	100.00 %	0.6837 ± 0.0049	

Table 4. Linear SVM performance with different feature selection methods at $S = 40$ and 20.

5.6 Training on the full training period

In the experiments presented so far, the largest training set, and the one used most of the time, was *Train-1600*, which consists of 118,924 documents, selected from the training period of documents dated 14 April 1997. However, the entire training period contains 504,468 documents, i.e., almost four times as many. Thus it is natural to ask how much we lose by limiting ourselves to training on the approximately 24% of the training documents (i.e., those included in *Train-1600*). Because it would be too time-consuming to conduct an exhaustive set of experiments on the full training period, we only trained one model for each category, without any feature selection. We compare the results with the performance of the linear SVM model trained on *Train-1600*, also without any feature selection applied.

As can be seen from Table 5, training on *Train-1600* tends to yield better performance for smaller categories but poorer for larger categories, while the contrary is true when training on the entire training period. The set of positive examples for small categories is in fact the same in both training sets, implying that it is the relative representation of the category within the training set that causes the difference in performance. Recall that the sampling procedure by which *Train-1600* was obtained tends to make sure that smaller categories are well represented at the expense of the larger ones which may cover a relatively smaller proportion of documents than they do in the full training period. This also explains why the results over *Train-1600* have a higher macroaveraged F_1 but a lower microaveraged F_1 than over the full training period.

However, part of the extremely poor F_1 performance of classifiers for some of the smaller categories (e.g., *c313*) when trained on the entire training period may also be due to a poorly chosen threshold b (see Section 3.1.3). Indeed, the breakeven point measure (also shown in Table 5), which involves calculating precision and recall of the document ranking based on scores that involve only the normal **w**, does not suffer such an extreme decrease when the full training period is involved.

	Train-1600 (118,924 docs.)				Entire training period (504,468 docs.)			
Category name	Size		F_1	BEP	Size		F_1	BEP
c13	9,895	(8.32%)	0.4891	0.5504	24,332	(4.82%)	0.5186	0.5873
c15	9,272	(7.80%)	0.8956	0.9024	89,373	(17.72%)	0.9167	0.9169
c183	2,240	(1.88%)	0.7469	0.7457	4,715	(0.93%)	0.7283	0.7572
c313	741	(0.62%)	0.3467	0.3603	741	(0.15%)	0.0804	0.2935
e121	1,444	(1.21%)	0.7442	0.7602	1,444	(0.29%)	0.7720	0.7585
e13	3,053	(2.57%)	0.7917	0.9715	4,135	(0.82%)	0.8036	0.7961
e132	602	(0.51%)	0.8109	0.8267	602	(0.12%)	0.8360	0.8453
e142	135	(0.11%)	0.4632	0.5280	135	(0.03%)	0.3582	0.4920
e21	8,484	(7.13%)	0.8129	0.8154	27,031	(5.36%)	0.8254	0.8315
ghea	2,616	(2.20%)	0.6442	0.6497	3,963	(0.79%)	0.6269	0.6532
gobit	551	(0.46%)	0.3838	0.4360	551	(0.11%)	0.1722	0.5157
godd	1,642	(1.38%)	0.2687	0.3701	1,798	(0.36%)	0.0829	0.3807
gpol	9,867	(8.30%)	0.7146	0.7260	36,650	(7.27%)	0.7437	0.7562
gspo	1,931	(1.62%)	0.9653	0.9761	22,876	(4.53%)	0.9792	0.9802
m14	9,812	(8.25%)	0.9398	0.9413	50,543	(10.02%)	0.9470	0.9478
m143	2,649	(2.23%)	0.9224	0.9257	13,121	(2.60%)	0.9264	0.9285
Macroaverage			0.6837	0.7178			0.6448	0.7150
Microaverage			0.8461				0.8635	

Table 5. Comparison of the performance of classifiers trained on *Train-1600* and those trained on the full training period. No feature selection was performed in either case.

The time needed to train all 16 models (one for each category) was 2,435 *s* when working with *Train-1600* and 15,769 *s* when working with the entire training period (these timings were obtained on a computer with a 700 MHz PIII processor and 1 GB main memory).

5.7 Combining feature weightings from several training data subsets

In comparing the sparsity behaviour of the *normal-*$^1/_{16}$ weighting, obtained from a smaller set of documents (*Train-*$^1/_{16}$), with that of *normal-1*, obtained from the full *Train-1600* set, we have seen that part of the difference stems from the unability of *normal-*$^1/_{16}$ to consider more documents at the same time, but part of it also stems from the simple fact that many features from *Train-1600* simply never occur in *Train-*$^1/_{16}$, hence *normal-*$^1/_{16}$ has no good alternative but to assign them a weight of 0 (see section 5.1.3). This in turn affects the classification performance of models based on the resulting feature ranking (see Figure 8).

Since our objective is to maximize the effectiveness of feature selection from smaller subsets of data, we explore the ways to increase the pool of features from which the final selection is made. (This will address the second of the two above-mentioned reasons of the poorer performance of weightings based on smaller subsets.) We consider the following extension of the feature weighting procedure: (1) obtain several subsets of the basic training set (i.e., subsets of *Train-1600*); (2) train a linear SVM model on each of the subsets; (3) from each model, assign a weight to each feature, equal to the absolute value of the corresponding component of the normal; (4) obtain the final weight of a feature by combining the weights obtained from models trained on individual subsets. In our preliminary experiments we used the average and the maximum as combining functions in step (4), and observed no significant differences between the two. Thus we decided to use the maximum in the following experiments.

The rationale behind this approach is that it allows us to work within the same memory limits as by limiting ourselves to a single small subset of the training set, while it also permits us to process all the documents and give all the features a chance to obtain a weight. It also helps reduce bias that would be introduced by the choice of a single subset.

To test this idea, we divided the *Train-1600* into 16 random disjoint subsets, without any particular regard for the distribution of categories within each of the 16 subsets. Thus, since the combined weightings approach needs to consider only one of the subsets at the same time, and because the subsets are approximately $^1/_{16}$ the size of the full training set, its memory requirements should be similar to those for *normal-*$^1/_{16}$.

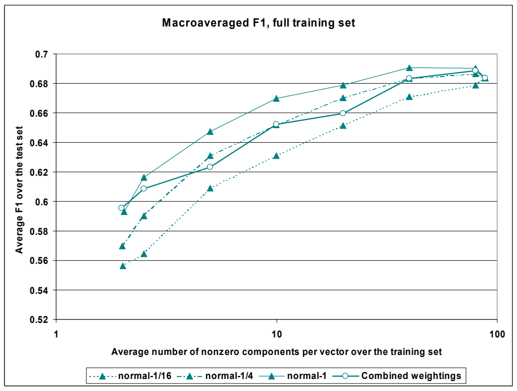


Figure 9.

The chart on Figure 9 shows a performance comparison of the combined feature weighting approach as described above and the *normal-*$^1/_{16}$, *normal-*$^1/_4$, and *normal-1* weightings. As can be seen from the chart, the performance of the combined weightings approach is roughly halfway between those for *normal-*$^1/_{16}$ and *normal-1*, and is indeed comparable to the performance of *normal-*$^1/_4$. In addition, the combined weightings approach performs particularly well at lower sparsity levels, which suggests that combining several weightings is a comparably robust feature weighting method.

Concerning the time requirements of the combined weighting approach, note that it actually takes less time to train *n* models on subsets 1/*n* the size of the full training set than to train one model on the full training set, because the training of an SVM model is not a linear-time algorithm. In order to obtain feature weightings for all 16 categories used in our experiments, the combined weighting approach requires 1,214 *s*, while *normal-1* required 3,934 *s;* however, *normal-*$^{1}/_{4}$, which is comparable in performance to the combined weighting approach, only requires 546 *s*. Incidentally, these figures suggest that, for the version of SvmLight we were using, the running time of SVM training (for N training documents) is on the order of $O(N^{1.43})$.

6 Summary and future work

In this study we emphasized the concept of sparsity as a more appropriate characteristic of the data representation than the number of features used, particularly when a variety of feature selection procedures are considered.

We also proposed a feature ranking and feature selection method based on the linear SVM that is then used in conjunction with the SVM classifier. However, this method can be combined with other classification algorithms as well. An interesting question then arises: is it still good to use feature selection based on SVM, or is it better to devise a feature selection method based on the training algorithm itself? In other words, can we say anything about the compatibility of the feature selection method with the final classification algorithm when applying the proposed iterative training strategy?

Furthermore, other linear classifiers could be similarly used to weight and select features, including Perceptron [Ros58], Winnow [Litt88], Bayes Point Machine [HGC01], LLSF [YC92], Widrow-Hoff [LSCP96], exponentiated gradient [KW94, LSCP96], and so on. (Naive Bayes is also essentially a linear classifier if we work with logarithms of probabilities. This has been exploited by e.g. [GF01].) While these methods are known to be more or less successful in classifying documents, it would be interesting to see how they compare with the SVM-based feature selection method in reducing the feature space.

Finally, it would be interesting to evaluate our approach on other data sets, perhaps on domains outside text categorization.

Appendix A: About the Reuters-2000 Corpus

The Reuters-2000 corpus, released on 3 November 2000 as "Reuters Corpus, Volume 1", contains 806,791 Reuters news articles from the period 20 August 1996 through 19 August 1997. They are distributed by Reuters in compressed form (ZIP files), occupying approximately 984 MB of space. Each document is a small XML file and when uncompressed these files have a total size of approximately 2,369 MB.

We only consider the body of each document, i.e., the contents of the ⟨*text*⟩ element in the XML file containing that document. The body of each document contains an average of 1,180.13 non-whitespace characters (median: 883), thus the total for the corpus is approximately 950 million characters (908 MB). Most of the difference between this and the 2,369 MB quoted above is due to the meta-data present in each file.

There are on average approx. 206.8 words in the body of a document (median: 150), or 118.8 (median: 88) if multiple occurrences of the same word in a document are treated as one. However, if we ignore stopwords (from a list of 523 stopwords, of which 519 actually appeared in the corpus), the average number of words in the body of a document is 118.47 (median: 88), or 88.4388 (median: 65) if multiple occurrences are treated as one. On the average, a term appears in 136.4 documents, or 96.42 if stopwords are ignored; the median is 2 in both cases (329,226 terms occur in only one document and 100,404 terms occur in only two documents).

There are 103 categories, organized hierarchically. For example, *ccat* is the common supercategory of all categories with names beginning in *c*; likewise there are *ecat*, *gcat*, and *mcat*. In addition, if one category name is the prefix of another, the latter category is a subcategory of the former. However, we did not take the hierarchical structure of the categories into account. The sizes of categories vary widely as can be seen in Table B1 in Appendix B. Further statistics of interest: a document belongs to 3.20 categories on average (the median value is 3); 2,364 documents belong to no categories at all; 70% of the documents belong to 2 or 3 categories; one document belongs to 17 categories, none belong to more than 17.

Statistics such as these given here will of course depend somewhat on the procedure used to break documents into words. Our procedure is as follows: (1) convert the document into lowercase; (2) extract, from each paragraph, maximal contiguous subsequences of non-whitespace characters; (3) for each such subsequence: (3a) if it contains no alphabetic characters, discard it; (3b) otherwise, strip leading and trailing non-alphanumeric characters, and report the remainder as a term occurring in the current document. — This approach has the potentially welcome characteristic of treating compounds consisting of two words connected with a hyphen as single units but it would do the same for two words connected by a sequence of dots, which sometimes occurs when such punctuation is used to simulate tables in the documents. However, such occurrences are relatively rare.

Appendix B: Classification of 103 categories over Train-200

This section presents the results of preliminary experiments on which our choice of 16 categories for further work was based.

For these preliminary experiments, a smaller training set, called *Train-200*, was prepared in the same way as the *Train-1600* used elsewhere in this paper (see Section 4.2.1). Similarly, a test set, called *Test-100*, was prepared from the test period data (analogously to the way *Train-200* and *Train-1600* were prepared from the training period) to evaluate the performance of the classifiers. *Train-200* consists of 19213 documents, and *Test-100* consists of 9596 documents.

After discarding features occurring in less than 4 documents from *Train-200*, each document was represented by a normalized TF-IDF vector. Each category was treated as a two-class problem, and a linear SVM model was trained for it using the documents from *Train-200*. This model was then tested on *Test-100*, and the resulting precision-recall break-even point (BEP in the table below) was used as an indicator of how difficult or easy that individual category is. Table B1 below reports the break-even points for all categories as well as the size of each category, i.e., the number of the documents (from the full Reuters-2000 corpus) that belong to the category.

Note that the full corpus contains 806,791 documents. The names of the 16 categories chosen for further experiments are displayed in italics.

Category	BEP	Size
c11	0.3552	24,325
c12	0.6299	11,944
c13	*0.5227*	*37,410*
c14	0.6645	7,410
c15	*0.7412*	*150,164*
c151	0.7833	81,875
c1511	0.7673	23,212
c152	0.5480	73,092
c16	0.7257	1,920
c17	0.7557	41,829
c171	0.6966	18,313
c172	0.7600	11,487
c173	0.6441	2,636
c174	0.9262	5,871
c18	0.7585	51,480
c181	0.6741	43,374
c182	0.6154	4,671
c183	*0.7484*	*7,406*
c21	0.4357	25,403
c22	0.4306	6,119
c23	0.5821	2,625
c24	0.5396	32,153
c31	0.5935	40,506
c311	0.6911	4,299
c312	0.5902	6,648
c313	*0.4808*	*1,115*
c32	0.7308	2,084
c33	0.6006	15,331
c331	0.8319	1,210
c34	0.7528	4,835
c41	0.7756	11,354
c411	0.7792	10,272
c42	0.6855	11,878
ccat	0.8711	374,316
e11	0.5650	8,568
e12	0.6671	27,078
e121	*0.8051*	*2,182*
e13	*0.8353*	*6,345*
e131	0.7751	5,659
e132	*0.9083*	*939*
e14	0.8119	2,086
e141	0.7155	376
e142	*0.8000*	*200*
e143	0.8684	1,206
e21	*0.7243*	*43,128*
e211	0.6368	15,768
e212	0.7596	27,405
e31	0.8300	2,342
e311	0.8101	1,701
e312	0.6500	52
e313	0.9111	111
e41	0.8030	16,900
e411	0.7246	2,136
e51	0.6612	20,722
e511	0.6798	2,933
e512	0.6261	12,634
e513	0.6522	2,290
e61	0.9100	391
e71	0.8818	5,270
ecat	0.8503	117,539
g15	0.8541	19,152
g151	0.5317	3,307
g152	0.4663	2,107
g153	0.7534	2,360
g154	0.8312	8,404
g155	0.3922	2,124
g156	0.6515	260
g157	0.7865	2,036
g158	0.5767	4,300
g159	0.0000	40
gcat	0.8845	234,873
gcrim	0.7612	32,219
gdef	0.7169	8,842
gdip	0.6732	37,739
gdis	0.8106	8,657
gent	0.7119	3,801
genv	0.6988	6,261
gfas	0.8350	313
ghea	*0.6522*	*6,030*
gjob	0.8101	17,241
gmil	0.0000	5
gobit	*0.7752*	*844*
godd	*0.4394*	*2,802*
gpol	*0.6518*	*56,878*
gpro	0.6740	5,498
grel	0.7778	2,849
gsci	0.7879	2,410
gspo	*0.9302*	*35,317*
gtour	0.8319	680
gvio	0.7003	32,615
gvote	0.7037	11,532
gwea	0.7847	3,878
gwelf	0.6967	1,869
m11	0.7549	48,700
m12	0.7041	26,036
m13	0.7610	52,972
m131	0.7411	28,185
m132	0.6885	26,752
m14	*0.8813*	*85,100*
m141	0.8522	47,708
m142	0.8779	12,136
m143	*0.8895*	*21,957*
mcat	0.8317	200,190

Table B1: Category size and break-even-point performance statistics of the linear SVM model for 103 Reuters categories over *Train-200* data set

The selected 16 categories are used in experiments that involve various sets of training and test data. The following table shows the names of the selected categories and the number (and percentage) of positive examples for each of these categories within several data sets. The document sets shown are: *Train-1600* (the largest of the sets actually used for training); *Train-*$^{1}/_{4}$ and *Train-*$^{1}/_{16}$ (subsets of *Train-1600;* see section 4.2.1 for details); the full training period (all documents dated 14 April 1997 or earlier); the full test period (all documents dated after 14 April 1997); and the entire Reuters-2000 corpus.

	Train-$^{1}/_{16}$ (7432 docs.)	*Train-*$^{1}/_{4}$ (29731 docs.)	*Train-1600* (118924 docs.)	Full training period (504468 docs.)	Full test period (302323 docs.)	Full Reuters-2000 (806971 docs.)
c13	702 (9.45%)	2,645 (8.90%)	9,895 (8.32%)	24,332 (4.82%)	13,078 (4.33%)	37,410 (4.64%)
c15	515 (6.93%)	2,199 (7.40%)	9,272 (7.80%)	89,373 (17.72%)	60,791 (20.11%)	150,164 (18.61%)
c183	128 (1.72%)	561 (1.89%)	2,240 (1.88%)	4,715 (0.93%)	2,691 (0.89%)	7,406 (0.92%)
c313	50 (0.67%)	197 (0.66%)	741 (0.62%)	741 (0.15%)	374 (0.12%)	1,115 (0.14%)
e121	101 (1.36%)	380 (1.28%)	1,444 (1.21%)	1,444 (0.29%)	738 (0.24%)	2,182 (0.27%)
e13	217 (2.92%)	883 (2.97%)	3,053 (2.57%)	4,135 (0.82%)	2,210 (0.73%)	6,345 (0.79%)
e132	47 (0.63%)	199 (0.67%)	602 (0.51%)	602 (0.12%)	337 (0.11%)	939 (0.12%)
e142	16 (0.22%)	67 (0.23%)	135 (0.11%)	135 (0.03%)	65 (0.02%)	200 (0.02%)
e21	510 (6.86%)	2,158 (7.26%)	8,484 (7.13%)	27,031 (5.36%)	16,097 (5.32%)	43,128 (5.35%)
ghea	205 (2.76%)	733 (2.47%)	2,616 (2.20%)	3,963 (0.79%)	2,067 (0.68%)	6,030 (0.75%)
gobit	44 (0.59%)	148 (0.50%)	551 (0.46%)	551 (0.11%)	293 (0.10%)	844 (0.10%)
godd	120 (1.61%)	430 (1.45%)	1,642 (1.38%)	1,798 (0.36%)	1,004 (0.33%)	2,802 (0.35%)
gpol	590 (7.94%)	2,385 (8.02%)	9,867 (8.30%)	36,650 (7.27%)	20,228 (6.69%)	56,878 (7.05%)
gspo	100 (1.35%)	449 (1.51%)	1,931 (1.62%)	22,876 (4.53%)	12,441 (4.12%)	35,317 (4.38%)
m14	557 (7.49%)	2,325 (7.82%)	9,812 (8.25%)	50,543 (10.02%)	34,557 (11.43%)	85,100 (10.55%)
m143	133 (1.79%)	608 (2.05%)	2,649 (2.23%)	13,121 (2.60%)	8,836 (2.92%)	21,957 (2.72%)

Table B2: Distribution of positive examples for the 16 selected Reuters categories in various data sets used in the experiments.

Appendix C: Number of features and sparsity statistics

The following table shows the relationship between the number of features retained in the feature set and the sparsity, i.e., the average number of non-zero components in the corresponding vector representations of documents in a given set. The statistics presented here are obtained over the *Train-1600* data set.

Number of features	Odds ratio	Information gain	*Normal-*$^{1}/_{16}$	*Normal-*$^{1}/_{4}$	*Normal-1*
10	0.002	0.542	0.271	0.270	0.301
20	0.004	0.998	0.559	0.507	0.534
30	0.005	1.446	0.786	0.796	0.777
50	0.007	2.224	1.238	1.292	1.222
75	0.009	3.172	1.772	1.804	1.733
100	0.015	4.111	2.249	2.301	2.270
200	0.037	7.123	4.187	4.266	4.328
300	0.074	9.708	6.140	6.322	6.074
500	0.152	14.040	9.674	9.703	9.132
750	0.343	17.966	13.612	13.261	12.464
1000	0.634	21.339	17.044	16.453	15.333
2000	2.583	30.405	28.025	26.385	24.319
3000	6.152	35.989	36.253	33.477	30.769
5000	11.158	43.087	48.674	44.405	40.914
7500	17.900	48.364	58.597	53.896	49.829
10000	25.689	51.566	65.621	60.790	56.088
20000	56.612	57.138	79.338	75.700	71.944
30000	70.085	60.469	83.996	82.288	79.682
40000	73.803	62.900	85.622	85.210	83.959
50000	75.002	66.192	86.587	86.535	86.137
60000	76.221	70.803	87.389	87.377	87.261
70000	79.046	78.843	87.980	88.007	87.910
75839	88.266	88.266	88.266	88.266	88.266

Table C1: Sparsity levels achieved on the *Train-1600* document set by retaining a given number of top ranked features for each of the five feature scoring methods considered

Each feature scoring method implies a ranking of features. Limiting the document representation to the highest scoring N features reduces the average number of terms that remain in a document and hence the average number of non-zero components in the vectors representing the documents. It interesting to see the difference in the achieved sparsity of the document representations for the same number of features retained for each of the five considered feature ranking methods: odds ratio, information gain, and three normal based feature selections.

References

[CV95] C. Cortes, V. Vapnik: *Support-vector networks*. Machine Learning, 20(3):273–297, September 1995.

[DPHS98] S. Dumais, J. Platt, D. Heckerman, M. Sahami: *Inductive learning algorithms and representations for text categorization*. Proceedings of the 1998 ACM 7th International Conference on Information and knowledge management, November 3–7, 1998, Bethesda, Maryland, USA, pp. 148–155.

[GF01] T. Gärtner, P. A. Flach: *WBC_{SVM}: Weighted Bayesian classification based on support vector machines*. Proceedings of the 18th International Conference on Machine Learning (ICML 2001), June 28–July 1, 2001, Williamstown, Massachusetts, USA, pp. 154–161.

[HGC01] R. Herbrich, T. Graepel, C. Campbell: *Bayes point machines*. Journal of Machine Learning Research, 1(Aug):245–279, August 2001.

[Joa98] T. Joachims: *Text categorization with support vector machines: learning with many relevant features*. Proceedings of the 10th European Conference on Machine Learning (ECML 1998), April 21–23, 1998, Chemnitz, Germany. LNCS vol. 1398, pp. 137–142.

[Joa99] T. Joachims: *Making large-scale support vector machine learning practical*. In: B. Schölkopf, C. J. C. Burges, A. J. Smola (Eds.): *Advances in kernel methods: Support vector learning*, The MIT Press, 1999, pp. 169–184

[KW94] J. Kivinen, M. K. Warmuth: *Exponentiated gradient versus gradient descent for linear predictors*. Technical Report UCSC-CRL-94-16, Baskin Center for Computer Engineering & Information Sciences, University of Califronia, Santa Cruz, CA, USA, June 21, 1994 (revised December 7, 1995).

[Lew92] D. D. Lewis: *An evaluation of phrasal and clustered representations on a text categorization task*. Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1992), June 21–24, 1992, Copenhagen, Denmark, pp. 37–50.

[Lew98] D. D. Lewis: *The Reuters-21578 Text Categorization Test Collection*. Available on: *http://www.research.att.com/~lewis/reuters21578.html.*

[Litt88] N. Littlestone: *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm*. Machine Learning, 2:285–318, 1988

[LSCP96] D. D. Lewis, R. E. Schapire, J. P. Callan, R. Papka: *Training algorithms for linear text classifiers*. Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1996), August 18–22, 1996, Zürich, Switzerland, pp. 298–306.

[MG99] D. Mladenić, M. Grobelnik: *Feature selection for unbalanced class distribution and Naive Bayes*. Proceedings of the 16th International Conference on Machine Learning (ICML 1999), June 27–30, 1999, Bled, Slovenia, pp. 258–267.

[Mla98a] D. Mladenić: *Feature subset selection in text-learning*. Proceedings of the 10th European Conference on Machine Learning (ECML 1998), April 21–23, 1998, Chemnitz, Germany. LNCS vol. 1398, pp. 95–100.

[Mla98b] D. Mladenić: *Machine Learning on non-homogeneous, distributed text data*. Ph. D. thesis, University of Ljubljana, Slovenia, 1998.

[Reu00] *Reuters Corpus, Volume 1, English Language*, 1996-08-20 to 1997-08-19. Available through *http://about.reuters.com/researchandstandards/corpus/*. Released in November 2000.

[Ros58] F. Rosenblatt: *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review 65(6), 386-408. Reprinted in: J. A. D. Anderson, E. Rosenfeld (Eds.), *Neurocomputing: Foundations of Research*, The MIT Press, 1998, pp. 89–114.

[SB88] G. Salton and C. Buckley: *Term-weighting approaches in automatic text retrieval*. Information Processing and Management, 24(5):513–523, 1988.

[YC92] Y. Yang, C. Chute: *A linear least squares fit method for terminology mapping*. Proceedings of the 15th International Conference on Computational Linguistics (COLING 1992), 23–28 July, 1992, Nantes, France, II:447–53.

[YP97] Y. Yang, J. O. Pedersen: *A comparative study on feature selection in text categorization*. Proceedings of the 14th International Conference on Machine Learning, July 8–12, Nashville, Tennessee, USA, pp. 412–420.

[YL99] Y. Yang, X. Liu: *A re-examination of text categorization methods*. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999), August 15–19, 1999, Berkeley, CA, USA, pp. 42–49.

[ZO01] T. Zhang, F. J. Oles: *Text categorization based on regularized linear classification methods*. Information Retrieval, 4(1):5–31, April 2001.