Journal of Universal Computer Science, vol. 14, no. 10 (2008), 1562-1596
submitted: 18/3/08, accepted: 25/5/08, appeared: 28/5/08 © *J.UCS*

Feature Selection for the Classification of Large Document Collections

Janez Brank
(Jožef Stefan Institute, Ljubljana, Slovenia
janez.brank@ijs.si)

Dunja Mladenić
(Jožef Stefan Institute, Ljubljana, Slovenia
dunja.mladenic@ijs.si)

Marko Grobelnik
(Jožef Stefan Institute, Ljubljana, Slovenia
marko.grobelnik@ijs.si)

Nataša Milić-Frayling
(Microsoft Research, Cambridge, UK
natasamf@microsoft.com)

Abstract: Feature selection methods are often applied in the context of document classification. They are particularly important for processing large data sets that may contain millions of documents and are typically represented by a large number, possibly tens of thousands of features. Processing large data sets thus raises the issue of computational resources and we often have to find the right trade-off between the size of the feature set and the number of training data that we can taken into account. Furthermore, depending on the selected classification technique, different feature selection methods require different optimization approaches, raising the issue of compatibility between the two. We demonstrate an effective classifier training and feature selection method that is suitable for large data collections. We explore feature selection based on the weights obtained from linear classifiers themselves, trained on a subset of training documents. While most feature weighting schemes score individual features independently from each other, the weights of linear classifiers incorporate the relative importance of a feature for classification as observed for a given subset of documents thus taking the feature dependence into account. We investigate how these feature selection methods combine with various learning algorithms. Our experiments include a comparative analysis of three learning algorithms: Naïve Bayes, Perceptron, and Support Vector Machines (SVM) in combination with three feature weighting methods: Odds ratio, Information Gain, and weights from the linear SVM and Perceptron. We show that by regulating the size of the feature space (and thus the sparsity of the resulting vector representation of the documents) using an effective feature scoring, like linear SVM, we need only a half or even a quarter of the computer memory to train a classifier of almost the same quality as the one obtained from the complete data set. Feature selection using weights from the linear SVMs yields a better classification performance than other feature weighting methods when combined with the three learning algorithms. The results support the conjecture that it is the sophistication of the feature weighting method rather than its compatibility with the learning algorithm that improves the classification performance.

Keywords: Feature Selection, Text Classification, Machine Learning, Support Vector Machines, Information Retrieval, Knowledge Representation, Data Preprocessing
Categories: H.3.3, I.2.4, I.2.6, M.4

1 Introduction

In classification of text documents, feature selection is typically used to achieve two objectives: (1) to reduce the size of the feature set used for data representation and thus optimize the use of computing resources and (2) to remove noise from the data in order to optimize the classification performance. A number of standard methods, such as stop word removal or linguistic normalization using stemming, are routinely applied and contribute to the reduction of the feature space, memory consumption, and processing time. In addition, further reduction is achieved by removing less 'relevant' features. This is typically a two step process. First, the features are scored in accordance with a weighting scheme believed to reflect the importance of the feature for a given task and then a subset of features, typically the top *N* scored features, are used for further processing.

A fundamental question that arises in practice is *the relationship between the scoring method for feature selection and the classification model* that further uses these features. Are there compatibility criteria that the two should satisfy in order to yield optimal classification performance? In a study of the Naïve Bayes text classifier [Mladenić, 99] the feature selection based on Odds ratio scores consistently lead to statistically significant improvements in comparison to the classification performance with the full feature set. This was attributed to the fact that the feature selection method was 'compatible' with the classification algorithm used. Indeed, the statistics used to compute Naïve Bayes and Odds ratio scores appear compatible in the sense that the features with the higher Odds ratio weights are expected to contribute more to the document scores assigned by the Naïve Bayes classifier. Thus, feature selection based on Odds ratio weights is expected to be effective in tuning the performance of Naïve Bayes. In order to gain more insight into this question, we explore whether for linear classifiers, for example, we can use the classifier itself to score and select features for classification [Mladenić, 04]. This approach has two advantages. First, it investigates the ultimate 'compatibility' between the feature weighting and document classification method since the selection and classification both use the same scoring. Second, it introduces a class of feature weighting functions which effectively take into account the whole set of features in the representation of documents and score the features relative to their contribution to the classification task.

The second practical issue is *processing of large datasets and optimizing available computing resources*. Most research of classification and feature selection techniques has been evaluated on smaller data sets, such as Reuters-21578 comprising 25 MB of text documents. Nowadays, operational systems have to deal with both large data sets and a wide range of computing platforms and devices, including those with limited computing resources. At the same time the efficiency of many learning algorithms relies upon the assumption that the data set is represented by a small feature set. Namely, in the internal representation of documents they use the fact that each document contains only a small subset of the whole vocabulary and thus the corresponding vector representation of this document (containing e.g. term

frequencies or TF-IDF weights) is sparse, i.e. most of its components have the values zero. For efficiency of processing, a sparse vector is commonly stored in a way such that only its nonzero components are represented explicitly (e.g., instead of having an array of dimension equal to the vocabulary size it is represented with a list of words that is much shorter than the vocabulary size). For the purposes of our discussion, we define the "sparsity" of a dataset as the average number of nonzero components in the vectors representing the documents of this dataset. Since only the nonzero components are stored and processed explicitly, one expects that it is the sparsity of the vector representation rather than the vocabulary size that influences the performance efficiency and the usage of the computing resources. Is there an effective way to select feature and optimize performance for the given computing resources? Using sparsity (defined as an estimated average number of non-zero vector components) to specify the cut-off criteria for feature selection is a natural step to take [Brank, 02].

Furthermore, in practice we often face the question: given a large collection of training data and a limited computing resource, what is the right trade-off between the size or sparsity of the feature space and the number of documents used for classifier training. In this paper we propose to deal with large document collections by using a subset of documents to select the features for representing the whole document collection. More precisely, we apply an iterative procedure that typically involves a subset of training data to train a classifier for the sole purpose of identifying a subset of useful features. The trained classifier is analysed to get the feature subset from the, classification model (for instance, nodes of the decision trees, normal of the linear models, features in the if-then rules). We have experimentally evaluated two linear classifiers: linear Support Vector Machines (SVM) and Perceptron. The weights of the generated linear model (i.e., the normal to the hyperplane that separates the classes) are used to rank features and select a smaller subset based on the sparsity requirement. In the second iteration the full set of data is represented in the reduced feature space and used to train the classifiers. The resulting classifiers are applied to the new test data. We observe how this type of feature selection method combines with its 'own' classification model and with others. Furthermore, we explore how feature selection methods can be used to make tradeoffs between the amount of training data and the sparsity of the document representation, given a fixed amount of system memory. Our experimental results show that selecting feature based on the linear-SVM model provides a suitable way of preserving the classification performance while significantly reducing the size of the feature space and increasing the sparsity of data.

The main contributions of our work include: (1) a broader perspective on the feature selection task that involves using linear classifiers themselves to address that issue, (2) an iterative method for tuning document representation and classifier performance that is suitable for large data sets, and (3) empirical data about the behavior of learning methods when combined with various feature selection methods. These experimental data enable us to make an informed conjecture on the compatibility of feature selection methods and the learning algorithm. They suggest that it is the sophistication of the feature selection method rather than similarity with the underlying learning algorithm that plays the crucial role in optimizing the classification performance. In particular, methods such as feature selection using

weights of a linear SVM model, which take into account statistical properties of all the features and documents simultaneously, tend to perform generally better across all the learning algorithms that we examined.

The paper is structured in the following way. We first give a brief description of the related work followed by the description of text classification methods and feature scoring that we use in our experiments. Then we describe the data, the data sampling strategy, and the experiment set-up. We discuss the experiment results and conclude with the summary of observations and an outline of the future work.

2 Related Work

Feature selection in Machine learning [Mitchell, 97] can be performed in different ways [Guyon, 03]. Simple filters, commonly used approach when dealing with a large number of features, involve pre-processing of the data to score each feature independently of each other and represent the data only by a subset of the top ranked features.

Feature scoring is usually performed in a supervised way (taking the class value into account), for instance, by measuring the expected cross entropy between the feature and the class, but often ignoring the relation with other features. Numerous scoring measures have been proposed including information gain, odds ratio, χ^2, term strength, etc. For detailed description of feature selection approaches and dimensionality reduction see [Mladenic, 06; Forman, 07].

It is important to note that feature scoring and selection can have a different effect on different classification models. Information gain, originally used in decision tree induction [Quinlan, 93] was reported to work well as a feature scoring with the *k*-nearest-neighbor classification algorithm on textual data [Yang, 97], while it performed much worse with Naïve Bayes on the binary classification problems [Mladenic, 03]. Expected cross entropy (modified Information gain) seems to work well with Naïve Bayes [Koller, 97] and in some cases significantly outperforms Information gain [Mladenic, 03]. Fisher-index was proposed for scoring features in document retrieval [Chakrabarti, 98], but no experiments in document classification have been reported. There has also been an increase of interest in the combination of multiple feature scoring heuristics [Olsson, 06; Qiu, 08]. Odds ratio and an SVM-based approach have also been applied to text classification for the purposes of literary study [Yu, 07].

Comparison of different feature selection measures in combination with the Support Vector Machines classification algorithm (SVM) has shown that using all or almost all the features usually yields the best performance [Brank, 02; Rogati, 02; Forman, 03], although in certain situations feature selection can also improve the performance of SVM. Thus [Gabrilovich, 04] found that feature selection helps in classification tasks where a small number of relevant terms are sufficient to classify the documents and the vast majority of terms are redundant or irrelevant; [Forman, 03] reports that feature scoring using Bi-Normal Separation can improve the performance of SVM on problems with a highly unbalanced class distribution; [Qiu, 08] reports improved performance of SVM with several variants of feature selection

based on the χ^2-heuristics. Thus while the SVM does not generally benefit from feature selection, sometimes it can benefit from it nevertheless.

Feature scoring 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 feature selection methods in combination with different classification methods. For example, an approach for scoring features based on SVM has been proposed in [Gärtner, 01], where SVM is used with a special kernel to learn the weights of features for use with a Naïve Bayes classifier. Some feature scoring methods coordinate feature selection and classification by taking the performance of classification model based on a set of features as the scoring value of that set of feature (e.g., [John, 94]). However, this is a rather time consuming process where evaluation of each feature set requires construction of several classification models - cross validation is commonly used to obtain average performance). On the other hand, there were conscious attempts to create feature selection 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 Naïve Bayes classifier and has been shown to improve its performance [Mladenic, 03]. 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.

As already mentioned, the feature scoring measures that are usually applied on text, assign a score to a feature independently of the other features. Our work proposes a feature scoring based on an induced classifier that we see as more sophisticated, as it considers statistical properties of all the features and documents simultaneously (unlike many widely-used feature scoring heuristics, such as information gain and odds ratio, which evaluate each feature separately from other features). This can be considered as an alternative to the traditional higher-order feature selection methods, such as those based on mutual information [Bakus, 06]. We compare the effect of feature selection based on the proposed approach (using weights from a linear model) with two feature scoring measures that have been reported successful in text classification: information gain and odds ratio (see Section 3.2 for details on the scoring measures). These two scoring measures are interesting also because they seem to prefer different types of features: 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.

3 Feature Selection on Very Large Document Collections

In situations where there is an abundance of training data it is conceivable that classifier training cannot be performed over the full set of data because of the limited

computing resources. The question then arises how to train a given classifier in a way that the whole training set is still taken into account.

Here we present a simple procedure that has proven quite effective, as our experimental results show. We first train linear Support Vector Machines (SVM) or Perceptron on a subset of training data to create an initial classifier. Then we select a subset of features to achieve a sufficiently sparse representation of data that allows us to include the whole set of documents. Finally, we train a classifier in the reduced feature space and evaluate its performance on the test data, as commonly done in machine learning.

In SVM and Perceptron models, each classifier is a hyperplane separating 'positive' and 'negative' examples for the class. The hyperplane is represented by the normal, i.e., a vector perpendicular to the hyperplane. Each feature is included in the normal with a weight that contributes to the final document score and influences the classification. In the feature selection step, we rank all the features based on their weight in the normal and retain a subset by controlling the sparsity (which is defined as the average number of non-zero components in the vector representation of data, or, equivalently, the average number of terms used to represent documents).

This approach takes advantage of the memory freed as a result of the increased data sparsity to include a larger training set while keeping the memory consumption constant. In the experimentation setting we evaluate the performance for various sparsity levels to identify the optimal tradeoff. In principle, the feature selection phase could include a separate validation set to suggest possible tradeoffs between sparsity and performance; alternatively, memory limitations may dictate what level of sparsity is necessary. Here we investigate the iterative feature selection for a spectrum of sparsity levels, focusing primarily on the interaction of feature selection and classification methods.

4 Learning Algorithms and Feature Selection Methods

In this study we investigate three learning algorithms: Naïve Bayes, Perceptron, and Support Vector Machines (SVM), and three feature scoring methods: Odds ratio, Information Gain, and weights from the linear models. In the case of the selected learning algorithms, feature selection can be considered a pre-processing phase and thus each of the feature scoring methods can be combined with each of the learning algorithms. While we are looking at a limited number of methods, we expect that studying their performance over large data sets will provide valuable insights into the nature of feature selection methods and their compatibility with classification models.

4.1 Learning Algorithms

4.1.1 Naïve Bayes

We use the multinomial model as described by [McCallum, 98]. The predicted class for a given document d is the one that maximizes the posterior probability of the class, $P(c|d) \propto P(c)\ \Pi_t\ P(t|c)^{TF(t,\ d)}$, where $P(c)$ is the prior probability that a document belongs to class c, $P(t|c)$ is the probability that a word chosen randomly in a document from class c equals t, and $TF(t, d)$ is the "term frequency", or the number of occurren-

ces of the word t in the document d. Where there are only two classes, say c_+ and c_-, maximizing $P(c|d)$ is equivalent to taking the sign of $\ln P(c_+|d)/P(c_-|d)$, which is a linear combination of $TF(t, d)$. Thus the Naïve Bayes classifier can be seen as a linear classifier as well. The training consists simply of estimating the probabilities $P(t|c)$ and $P(c)$ from the training documents, where we are using Laplace probability estimate.

4.1.2 Perceptron

The Perceptron algorithm [Rosenblatt, 88] trains a linear classifier in an incremental way as a neural unit using an additive update rule. The prediction for a document represented by the vector $\mathbf{x}$ is $\text{sgn}(\mathbf{w}^T\mathbf{x})$, where $\mathbf{w}$ is a vector of weights obtained during training. Let $\mathbf{x}_i$ be the i-th training vector, and y_i its corresponding class label ($y_i = +1$ for a positive document, $y_i = -1$ for a negative one). The training starts with $\mathbf{w} = \mathbf{0}$, then considers each training example $\mathbf{x}_i$ in turn. If the present $\mathbf{w}$ classifies $\mathbf{x}_i$ correctly it is left unchanged, otherwise it is updated according to the additive rule: $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$. To implement a variable threshold in the classifier, we extend, before training, each vector with an additional component with the value of 1; the corresponding component of the weight vector $\mathbf{w}$ then functions as a threshold.

4.1.3 Support Vector Machine

The Support Vector Machine (SVM) [Cortes, 95], when used with a linear kernel, trains a linear classifier of the form $prediction(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + b)$. That is, the space of possible vectors x is divided by the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$, where $\mathbf{w}$ is a normal of the hyperplane (i.e. a vector perpendicular to it), and b is a scalar which defines the plane's location in space. Learning is posed as an optimization problem with the goal of maximizing the margin, i.e., the distance between the separating hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ and the nearest training vectors. The vector of weights $\mathbf{w} = (w_1,...,w_d)$ 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. An extension of this formulation, known as the soft margin, also allows for a wider margin at the cost of misclassifying some of the training examples. The dual form of this optimization task is a quadratic programming problem and can be solved numerically. We used the SvmLight program [Joachims, 99] to train the SVM models. This program is particularly suitable for our purposes because it is optimizied for computations with linear kernels. In all the experiments reported in this paper, we used the linear error cost (L1) formulation of SVM. (For Naive Bayes and Perceptron, we used our own implementation of these two learning algorithms.)

We selected the linear version of the SVM and Perceptron models for our study because the existing literature indicates that for the text classification problem the nonlinear versions of these algorithms gain very little in terms of performance [Joachims, 98].

4.2 Feature Selection Methods

The three feature selection methods that we consider in our study involve the same procedure: assigning a weight to each feature, ranking the features based on the weights, and retaining only a specified number of features.

4.2.1 Odds ratio

Let $P(t|c)$ be the probability of a randomly chosen word being t, given that the document it was chosen from belongs to a class c. Then $odds(t|c)$ is defined as $P(t|c)/[1-P(t|c)]$ and the odds ratio equals to

$$OR(t) = \ln[odds(t|c_+)/odds(t|c_-)].$$

This formula is not symmetric with respect to which class we select as positive c_+ and which as negative c_-. Obviously, this scoring measure favors features that occur in positive examples rather than in negative ones. As a result a feature that occurs very few times in positive documents but never in negative documents will get a relatively high score. Thus, many features that are rare among the positive will be ranked at the top of the feature list. In this manner rare rather than representative features of the positive documents obtain high scores.

Odds ratio is known to work well with Naïve Bayes [Mladenic, 99] for categorizing web pages in a collection specially built for profiling web users [Mladenic, 02] and for classifying web pages into Yahoo! topic categories [Mladenic, 03]. In the latter case, an increase in the F_2 measure (from 0.13 to 0.6) was reported when only 0.2% to 5% of the original features were kept. Odds ratio has also been found to work particularly well in combination with bi-normal separation by Forman [Forman, 03]. A measure designed on similar principles, called "feature strength", has been used in a semi-supervised setting by Fung et al. [Fung, 05]. In our experiments we used a smoothed version of the original formula avoiding singularities, as also used in [Mladenic, 99]; namely, if $P(t|c) = 0$, the value $1/n^2$ (where n is the number of documents in the training set) is used instead of 0 when computing the odds. Thus the resulting odds are always strictly between 0 and 1.

4.2.2 Information gain

In calculating the information gain score we treat both the class membership and the presence or absence of a particular term as random variables. We compute how much information about the class membership is gained by knowing the term presence or absence statistics (as is used in the decision tree induction, e.g., in Quinlan, 1993). Indeed, if the class membership is interpreted as a random variable C with two values, *positive* and *negative*, and a term is likewise seen as a random variable T with two values, *present* and *absent*, then using the information-theoretic definition of *mutual information* we may define information gain as:

$$IG(t) = H(C) - H(C|T) = \Sigma_{\tau,c} P(C=c,T=\tau) \ln[P(C=c,T=\tau)/P(C=c)P(T=\tau)].$$

Here, τ ranges over {*present*, *absent*} and c ranges over $\{c_+, c_-\}$. As pointed out above, this is the amount of information about the class label C gained by knowing T, the presence or absence of a given word. The entropy is defined as $H(X) = \sum_x P(X=x) \ln P(X=x)$.

The information gain therefore measures how much we learn about C by knowing T, since $H(C|T)$ 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 $[H(C) + H(T)] - H(C, T)$, which can be interpreted as the amount of information that each of these variables C and T contains about the other one. For that reason this measure is also

known as (*average*) *mutual information*. It should not, however, be confused with $\log[P(t|c)/P(t)]$ or the maximum or average of this value over all c, which is sometimes also called mutual information.

Information gain has been found to work well with several algorithm including the *k*-nearest-neighbor algorithm on the small Reuters-22173 collection and the OHSUMED collection [Yang, 97] and Naïve Bayes, decision trees and linear SVM on the Reuters-21578 collection [Dumais., 98].

4.2.3 Feature selection based on linear classifiers

Using weights from the SVM classification model for feature selection was suggested in [Sindhwani, 01] and studied in detail for linear SVM in [Brank, 02] and used in [Guyon, 02; Guyon, 03]. Here we extend this idea to general linear classifiers and discuss the procedure by referring to linear SVM and Perceptron.

Both SVM and Perceptron, as linear classifiers, output predictions of the form:

$$prediction(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) = \mathrm{sgn}(\Sigma_j\, w_j x_j + b)$$

where the vector $\mathbf{w}$ and scalar b are obtained during training. Thus, a feature j with the weight w_j close to 0 has a smaller effect on the prediction than features with large absolute values of w_j. The weight vector $\mathbf{w}$ can also be seen as the normal to the hyperplane used by the classifier to separate positive from negative instances. Thus we often refer to the procedure as the *normal-based feature selection*. One speculates that since features with small $|w_j|$ are not important for categorization they may also not be important for learning and, therefore, are good candidates for removal. Thus, 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 (i.e. the threshold is set to such a value that a desired level of sparsity is obtained after discarding the features below the threshold).

A theoretical justification for retaining the highest weighted features in the normal has been independently derived in a somewhat different context in [Shih, 02]. There a feature is considered 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 normal $\mathbf{w}$.

Due to the way the SVM algorithm works, the vector $\mathbf{w}$ in a linear SVM model is always of the form $\mathbf{w} = \sum_i \alpha_i\, y_i\, \mathbf{x}_i$, where the sum goes over all training vectors ($\mathbf{x}_i$ is the i'th training vector, and y_i is its corresponding class label), and the multipliers α_i are values that have been computed during the SVM learning process. From this observation we see that one can regard $\|\mathbf{w}\|^2$ as a function of the training vectors $\mathbf{x}_1, ..., \mathbf{x}_l$, where $\mathbf{x}_i = (x_{i1}, ..., x_{id})$, 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} (for various i) Of course this disregards the fact that if the training vectors change, the values of the multipliers α_i would also change, but the approach roughly captures the essence of feature contributions. For the linear kernel, the sum of these partial derivatives turns out to be

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

where the sum is over all the support vectors and κ is a constant independent of j. Thus the features with the higher $|w_j|$ are more influential in determining the width of the margin. Analogous reasoning can also be used in the case of an SVM model based on a non-linear kernel, as $\|\mathbf{w}\|^2$ can still be expressed using only the training vectors $\mathbf{x}_i$ and the kernel function. When a kernel has been used that corresponds to mapping all vectors $\mathbf{x}_i$ into new vectors $\phi(\mathbf{x}_i)$ in a new (possibly very high-dimensional) space, the normal vector $\mathbf{w}$ is now expressed as $\mathbf{w} = \Sigma_i \alpha_i y_i \phi(\mathbf{x}_i)$, and $\|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} = \Sigma_i \Sigma_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, where K is the kernel function. If we pretend for a moment that the α_i multipliers can be held fixed while the training vectors $\mathbf{x}_i$ change by a very small amount, and if we can compute the partial derivatives of K, we can then also express the partial derivatives of $\|\mathbf{w}\|^2$ with respect to x_{ij} (see [Sindhwani, 01] for details).

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 $\mathbf{w}$ in order to arrive at the feature weights. Since training an SVM model requires a considerable amount of CPU time, and practically requires all the training vectors to be present in main memory at all times, it is desirable to use a subset of the training data in order to facilitate feature selection and then retrain the classifier over the full data set but using the reduced feature space. Furthermore, it is useful to explore how the selection of the training set affects feature selection and the performance of the final classifier.

We illustrate these issues by using different samples of the data for initial training: the full set, one quarter and one sixteenth of the full data size. After the removal of a large number of features both the time and the memory requirements for processing the entire training set are reduced and thus make that task feasible. Of course, having used only a subset of data to score and select features may render a worse performance than methods that take into account the whole data set. However, reported experiments show encouraging results. For example, feature selection based on the SVM improves performance of the Naïve Bayes classifier [Mladenic, 04]: F_1 increases from 0.38 to 0.54 on the RCV1 collection, while using on average only 5 features per document to capture its content, or equivalently, to keeping only about 0.1% of the whole feature set.

5 Data

5.1 Training and test data

For the experiments we use the Reuters Corpus Volume 1 collection (see Appendix A for details). The full Reuters corpus of 806,791 news articles amounts to over 2 GB of text data, dated from 20 August 1996 through 19 August 1997. We divided it into a "training period" that includes 504,468 articles, dated from 20 August 1996 through 14 April 1997; and a "test period" with the remaining 302,323 documents. For training we used a sample of 118,294 documents from the training period and tested the classifiers on the complete document set from the test period.

The training set was obtained using a partly stratified sampling approach (as described in detail in [Brank, 02]). From each category of RCV1, we selected a random set of up to 1600 documents (if the category contains less than 1600 documents, all of them were selected). The union of these sets is the 118,294-

document training set mentioned above; we will refer to it as *Train-1600* in the subsequent discussion. As a result of this approach, the smaller categories are not as poorly represented in our training set as they would be if a completely unstratified sampling had been used; on the other hand, the largest categories tend to be somewhat underrepresented relative to the full corpus. This is illustrated on Figure 1, which shows, for each category, the percentage of documents that belong to it, either in the full corpus or in its various subsets.

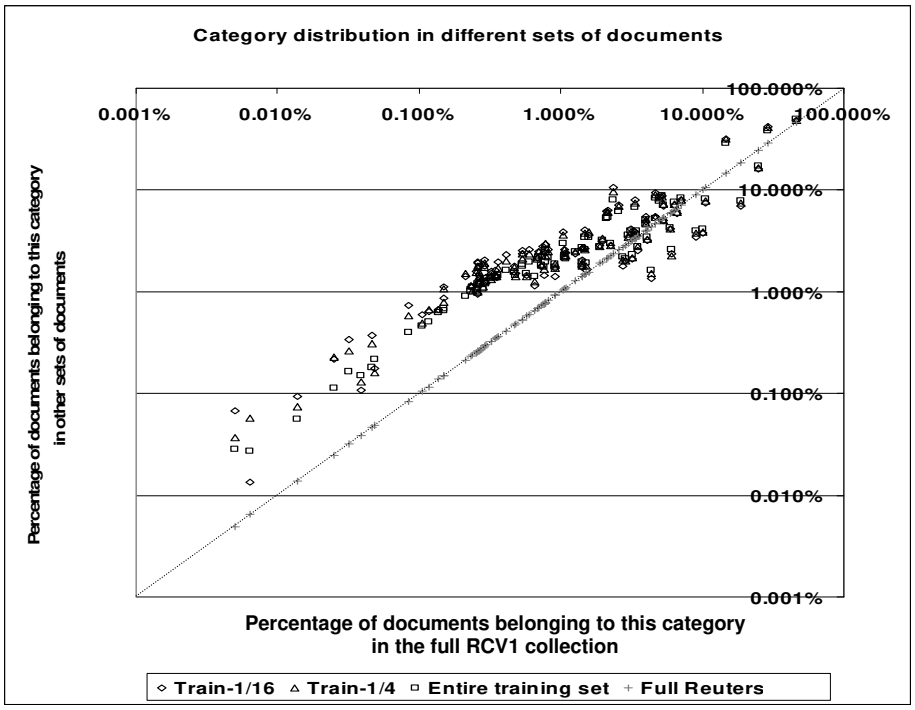

Figure 1: Distribution of documents over categories for different subsets of the full RCV1 collection. There is a symbol for each category, the x-coordinate showing the percentage of RCV1 documents belonging to this category, and the y-coordinate showing the percentage of documents belonging to this category in a subset of Retuers-2000. The dotted line shows where the two percentages would be the same. As we can see from the figure, the smaller categories are slightly overrepresented in the subsets (relative to their frequency in the full dataset), and the largest few categories are slightly underrepresented.

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. For comparison, the category distribution on these subsets is also shown on Figure 1.

5.2 Category selection

The Reuters collection uses 103 distinct categories to classify documents. However, training classifiers for all 103 Reuters categories over relatively large sets is a time consuming and process intensive task. Therefore we restricted our study to a sample of 16 categories that were selected in [Brank, 02] based on a preliminary document classification experiment that involved training the SVM over a smaller training set (19,213 documents) for the complete set of Reuters categories and classifying a test set of 9,596 documents. The training set was constructed in exactly the same way as the Train-1600 except that we used only 200 documents per category (referred to as Train-200). 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 were taken; this resulted in a sample of 9,596 test documents, referred to as Test-100.

The selection criterion was based on two characteristics of a category: the distribution of positive examples in the whole corpus and the precision-recall break-even point for the category achieved in the preliminary experiment (on Train-200 and Test-100). These statistics for the selected subset of 16 categories approximately follows the distribution for all 103 categories (see Figure 2 and Appendix B for more details). The selected set of categories includes: *godd*, *c313*, *gpol*, *ghea*, *c15*, *e121*, *gobit*, *m14*, *m143*, *gspo*, *e132*, *e13*, *c183*, *e21*, *e142*, and *c13*.

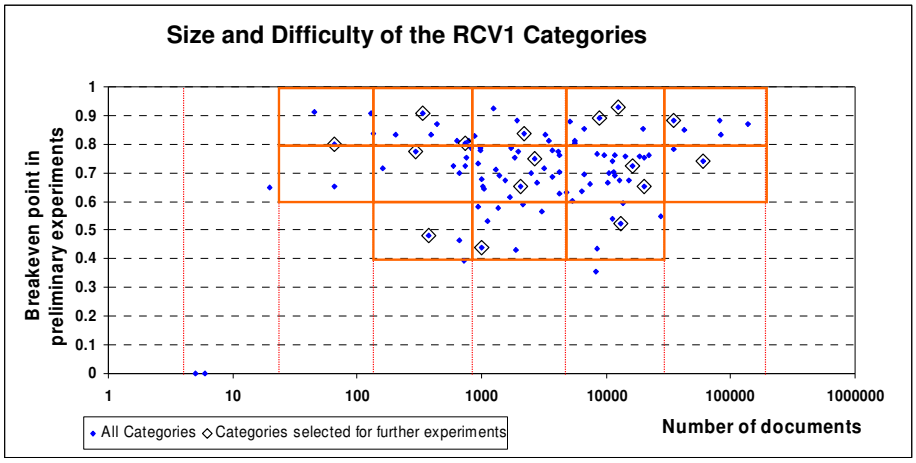

Figure 2: Size and difficulty of categories shown in a 2-d histogram with the log scale for the document distribution and 0.2 interval partition of the break-even point range.

6 Experiment Design

In this section we describe in detail the experiments that we ran with samples of data from the RCV1 corpus (Reuters Corpus Volume 1, http://about.reuters.com/researchandstandards/corpus/) and a sample of 16 categories from the set of 103 categories that constitute the Reuters classification scheme (Section 5). The

experiment design is similar to the study presented in [Brank, 02]. Our document representation is based on the bag-of-words model. We convert text of each document into lowercase, remove stop-words using a standard set of 523 stop-words, and eliminate words occurring in less than 4 training documents. Documents were represented using single word features weighted by the standard TF-IDF score [Salton, 88]. Document vectors were normalized to unit length, except for the representation used for the Naïve Bayes classifier which involved only the TF term weighting.

In order to control for memory consumption, 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.

6.1 Sparsity of data representation

It has been shown in [Brank, 02] that the document vector sparsity varies significantly when the same number of features is retained for different feature weighting methods. Similarly, a fixed sparsity level yields features sets of very different sizes. This is a rather important observation since it is expected that the performance of the data processing algorithm will depend on the representation of individual documents. Namely the number of features that we use reflects the document representation on a level of vocabulary size of all the documents, while sparsity captures the average number of features in individual document representation.

As an illustration, Figure 3 (lower chart) shows how specifying the sparsity levels for Odds ratio, Information Gain and linear classifier (Perceptron or SVM) based feature selection affects the performance of the SVM classifier on our data. Each sparsity level is, in turn, achieved by retaining a suitable number of highly scored features. We show the same performance results in two ways: with respect to the number of features retained (Figure 3, upper chart) and with respect to the average number of non-zero components in the training documents (Figure 3, lower chart) for each of the feature selection methods .

The sparsity of document vectors, on the other hand, directly affects consumption of computing resources, both the memory for storing the sparse vectors and the time needed to perform calculations [Milić-Frayling et al.]. While reducing the total number of features is expected to make the vectors sparser (i.e. the average number of nonzero components in each feature vector will become smaller and smaller as more and more terms are being discarded), the rate at which this is achieved depends on the term weighting scheme or, ultimately, the corpus properties such as term distribution (i.e., number of documents in the corpus, or labelled by a particular class, that contain the term). For example, if the number of features to be retained is fixed, weighting schemes that lead to retaining rare features generally incur a low cost to data storage and calculations. The opposite is true for weighting schemes that favor features common across documents. Thus, just specifying a fixed number of features does not allow for a reliable control over resource consumption. For that reason, we propose to

work directly with the requirement on the sparsity level as the cut-off criterion for feature selection.

6.2 Evaluation measures

In our analyses we use standard evaluation measures from the area of information retrieval. Each class is regarded as a binary classification problem (does a document belong to this class or not?), and a separate binary classifier is trained for each class. When evaluating the classifier for a particular class c, we compute the contingency table by counting the number of test documents that fall into each of the following four groups:

		Documents that...	
		Belong to c	Do not belong to c
The classifier's prediction is	positive	*TP* (true positives)	*FP* (false positives)
	negative	*FN* (false negatives)	*TN* (true negatives)

Based on these values, the precision of the classifier is defined as $p = TP/(TP + FP)$ and the recall is defined as $r = TP/(TP + FN)$. To combine these two quantities, we use the well-known F_1-measure, defined as $F_1 = 2pr/(p+r)$, i.e. the harmonic mean of precision and recall. Since the F_1 measure incorporates both the recall and the precision, we report only the experimental statistics for the F_1 measure.

To obtain an overall measure of performance of a machine learning approach over several classes, macroaveraging and microaveraging are commonly used. A macroaverage is defined as simply the average value of a performance measure (e.g. F_1) over all classes. The microaverage, on the other hand, is obtained by first adding up the contingency tables for all the classes into an aggregate contingency table (i.e. the aggregate *TP* is the sum of the *TP*s from all the individual contingency tables, etc.). The values of the aggregate contingency table are then used to compute precision, recall, and F_1 using the same formulas as above; the resulting values are called microaveraged precision, recall, and F_1. We found that analyses based on microaveraged statistics provide qualitatively similar results and thus we omit them from the paper. Instead we ensure that all the presented comparative analyses are accompanied with the evidence of statistical significance. Here we briefly discuss the approaches for establishing such evidence.

In order to compare two classifiers, we randomly partition the test data set into ten disjoint subsets and record performance measures of the classifiers, such as macro- or microaveraged F_1 statistics, separately for each subset of test data. Thus, to compare two classifiers, we can perform a paired t-test on the corresponding two groups of ten 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, across categories. These values can be quite informative, but the downside is that the comparison of two methods is then 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 some type of a sign test (e.g., a "macro sign test" from [Yang, 99]).

Another option is to compute, for each category, the value of F_1 over the entire test set. This results in a sequence of as many values as there are categories. One can thus compare the two methods by taking a paired t-test over the corresponding two sequences. This approach, referred to as a "macro T-test" in [Yang, 99], has been often used, although criticized because it treats the performance values of different categories as random samples from a normal distribution [Lewis, 92].

In our experience, all these types of tests tended to give similar results. However, it is noticeable that the category-paired t-test is slightly more conservative than the t-test on macroaveraged F_1-values in the sense that it is less likely to declare the difference to be significant. In contrast, the t-test on microaveraged F_1-values is much more liberal in considering differences to be significant. In the rest of this paper we report significance results from the t-tests based on macroaveraged F_1 values.

7 Results

7.1 Effects of feature selection on classification performance

We discuss the classification results for data representations obtained by feature ranking and selection using Odds ratio, Information gain, and normal based feature scoring from the linear SVM and Perceptron classifiers. For both the SVM and the Perceptron we apply the iterative method for classification: we train the classifier on a subset of training documents and use it to score and select features. The reduced feature space is then used for retraining the classifier over the entire document set. In order to observe the influence of the subset selection on the classification performance we defined three sets of training data: the full training set and randomly chosen subsets of $^1/_4$ and $^1/_{16}$ of the training documents. Training the classifiers on these sets results in three normal vectors for each method: *svm-1, svm-*$^1/_4$, and *svm-*$^1/_{16}$ for the linear SVM and *perceptron-1, perceptron-*$^1/_4$, *perceptron-*$^1/_{16}$ for the Perceptron. These 6 feature rankings, together with the Odds ratio and Information gain, are considered for feature selection and training of three classifiers: Naïve Bayes, the linear SVM, and Perceptron.

As already described, for each rank in the list of features we can calculate the average sparsity of vectors achieved if only features from that rank and above were retained. Similarly, given a sparsity requirement we can find the cut off rank below which the features are discarded in order to achieve the specified vector sparsity. It is interesting to note that specifying sparsity leads to non-uniform number of features across categories, depending on the distribution characteristics of features typical for the category. This is in contrast with the common practice of specifying a fixed number or percentage of top ranking features to be used for all categories.

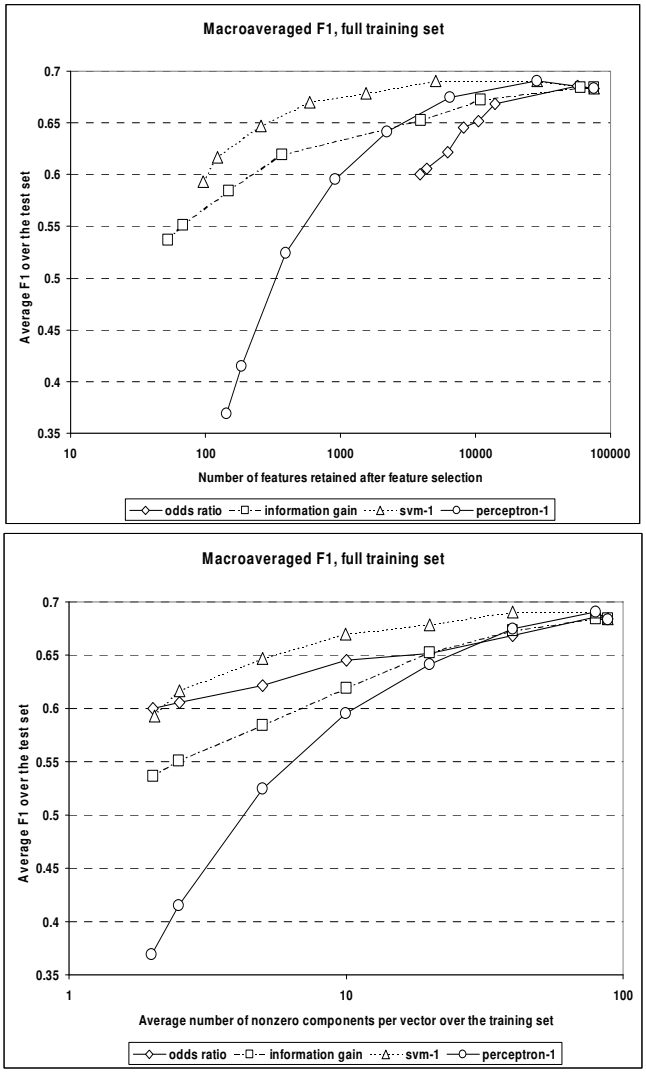

Figure 3: Macroaverages of the F_1 measure for the SVM classifier for various feature set sizes and different feature weighting methods. The upper chart shows the performance against the number of features while the lower one shows the same performance at various sparsity levels. We found that the sparsity-based charts allow for a more useful comparison of feature selection methods; e.g., Odds ratio is quite successful at very sparse representations, even though it uses a large vocabulary.

The experiment design thus includes nine sets of features (Odds ratio, Information Gain, 3 from SVM normal, 3 from Perceptron normal) used for training the learning algorithms, Naïve Bayes, Perceptron, and linear SVM, over the full set of

training documents represented by the reduced set of features. We specify the reduction level in terms of the desired sparsity of vectors.

The resulting classifiers are applied to the test data. Table 1 gives the result of Naïve Bayes for different feature selection methods for sparsity that yielded the best performance. More complete results for all three classification algorithms are shown in Figure 4, Figure 5, Figure 6 giving the macroaveraged F_1 performance for specific levels of vector sparsity on the horizontal axes: 2, 2.5, 5, 10, 20, 40, and 80 terms per document, as well as for the sparsity of the full document vectors, about 88.3 terms per document on average.

Feature selection/ranking method	Sparsity which yielded the best performance	Average number of features at that sparsity	Naïve Bayes macroaveraged F_1 at that sparsity
Odds ratio	20	10571	0.4683
Information gain	2.5	67	0.4648
SVM-1	**5**	**259**	**0.5421**
Perceptron-1	10	910	0.5005

Table 1: This table shows, for each feature selection method, the best Naïve Bayes classification performance, the sparsity at which it was achieved, and the number of features at that sparsity (averaged across all categories).

7.1.1 Naïve Bayes

Our experiments with Naïve Bayes confirm the known fact that Naïve Bayes benefits from the appropriate feature selection [Mladenic, 99]. Detailed analysis shows that this is particularly true for categories with a smaller number of training documents, while for the largest few categories there is little or no improvement from feature selection.

Odds ratio seems to work well with Naïve Bayes when documents are made moderately sparse, 10 to 20 terms per document which, on average, requires 4000 to 8000 of the best features to be retained. On the other hand, earlier research found that Odds ratio works well even if only a few hundred features were kept [Mladenic, 99]. Thus, it would be worth exploring whether sparsity is a more reliable parameter in predicting classifier performance than the number of features retained.

Information gain, on the other hand, works well when documents retain around 2.5 terms on average per document, which corresponds to keeping around 70 best features on average (the actual number varies considerably from one category to another, with *e13* needing as few as 29 and *ghea* as many as 163).

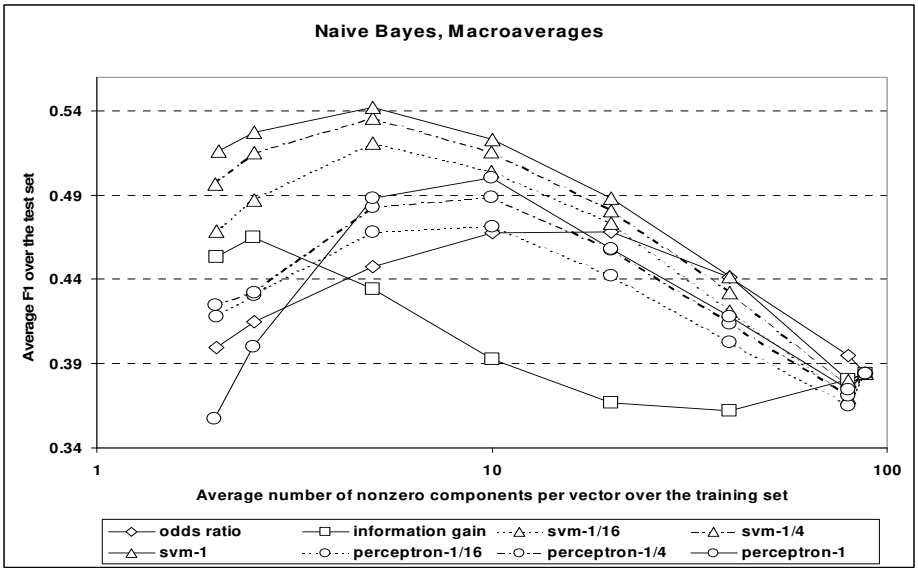

Figure 4: Macroaveraged F_1 of different feature selection methods in combination with Naïve Bayes. The horizontal axes refer to sparsity, the average number of nonzero components per training vector. In this case, this is the average number of terms per training document after terms rejected by feature selection have been removed.

Still higher performance with Naïve Bayes can be achieved with feature selection based on Perceptron and linear SVM weights. For SVM, this is true even if much smaller data sets are used in the feature selection phase (using 1/16 of the training data to get SVM-normal ranking the features). Table 1 provides detailed performance results. However, as it can be seen from Figure 4, even with the best of the tested feature selection methods, Naïve Bayes cannot match the performance of the Perceptron and the linear SVM classifiers.

7.1.2 Perceptron

Our experiments show that the Perceptron learning algorithm does not combine well with the feature selection methods considered here. The only case in which performance is actually improved is for the sparsity level of 80 terms per document with *perceptron-1*, thus a slight reduction in sparsity from 88.3 when all features are used. It is worth noting that this means the reduction of 8 features on average per document which corresponds to retaining less than 50% of the original features; more precisely around 32,500 out of 75,839 original features. The associated increase in F_1, although statistically significant, is marginal (0.8 %). Otherwise, performance drops quickly and considerably as more and more features are discarded.

When using SVM-based rankings, the performance is lower although the differences with respect to the Perceptron weighting are small. They are greatest when both feature selectors are allowed to inspect the full document set: *perceptron-1* may

achieve F_1 values up to 2.5 % above those achieved by *svm-1* at the same sparsity. The differences between *perceptron-1/4* and *svm-1/4* and between *perceptron-1/16* and *svm-1/16* are smaller and often insignificant. Thus, in scenarios of limited computer resources, when it is desirable to reduce the feature set by first training linear models over smaller training data sets, the two feature scoring methods combine equally well with the Perceptron as the classifier. Furthermore, while the SVM-based feature rankings achieve lower precision and lower F_1 values than the Perceptron ones, they tend to have higher recall and break-even point for the same subset of training documents.

In combination with the Perceptron classifier, the feature ranking by the Information gain criterion usually performs worse than the rankings of linear classifiers. The difference is slight but statistically significant. On the other hand, the feature ranking by Odds ratio performs much worse (interestingly, it combines well with the SVM; see the next section). We speculate that the Perceptron type training, which considers individual documents sequentially, is negatively affected by the tendency of the Odds ratio to favor features characteristic of positive documents, in particular those that are rare and absent from negative documents. It would be interesting to see whether Perceptron models with uneven margins are more robust in that respect [Krauth, 87].

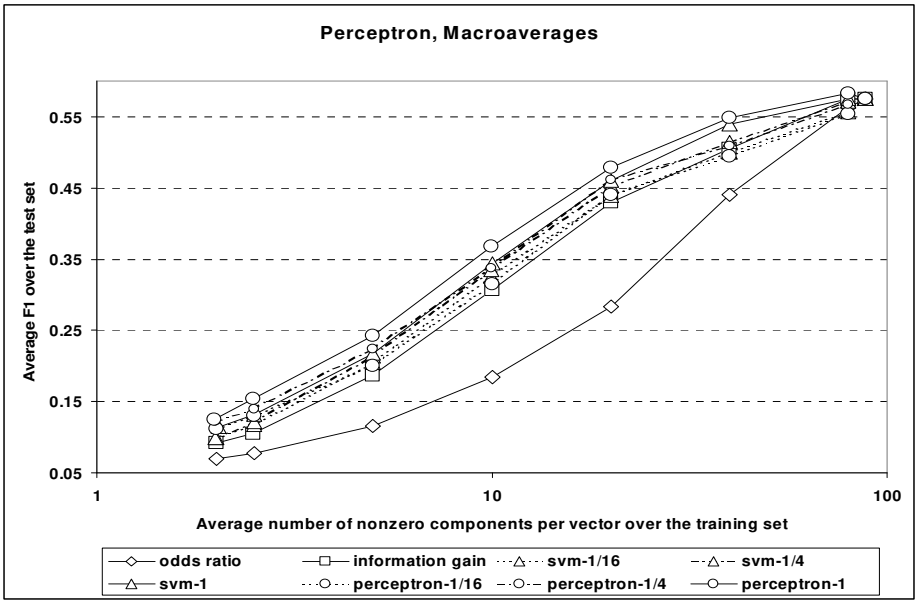

Figure 5: Macroaveraged F_1 of different feature selection methods in combination with the Perceptron classifier. The horizontal axes refer to sparsity, the average number of nonzero components per training vector. In this case, this is the average number of terms per training document after terms rejected by feature selection have been removed.

7.1.3 Linear SVM

Similarly to Perceptron, the linear SVM does not benefit from feature selection but is much less sensitive to the reduction of the feature space. Here one can reduce the feature set to 10 terms per document on average, while still losing only a few percent in terms of the F_1 performance measure. As in the case of Perceptron, statistically significant (though rather small) improvements to the F_1 performance in comparison to the full feature set are achieved only when using *svm-1* or *perceptron-1* feature sets, thus increasing the sparsity from 88.3 to 80 terms per document on average (using less than 50% of the original features).

Odds ratio works well in combination with the linear SVM classifier, particularly when very sparse documents representations are required (e.g., to capture characteristic words/phrases for under-represented categories in a very large document collections). Indeed, at sparsity ≤ 20 it is significantly better than Information gain. There it also outperforms *svm-*$^1/_{16}$. At more extreme levels of sparsity it performs better than *svm-*$^1/_4$ and even *svm-1* (for sparsity ≤ 2.5).

An interesting observation is that the Perceptron-based feature weightings perform much worse with the SVM classifier than the SVM-based weightings. Both feature weighting methods performed quite similarly in combination with the Perceptron as the classifier (see previous section). In addition, *perceptron-1* is generally not significantly better than *perceptron-*$^1/_4$, and for extremely sparse documents it is, in fact, significantly worse.

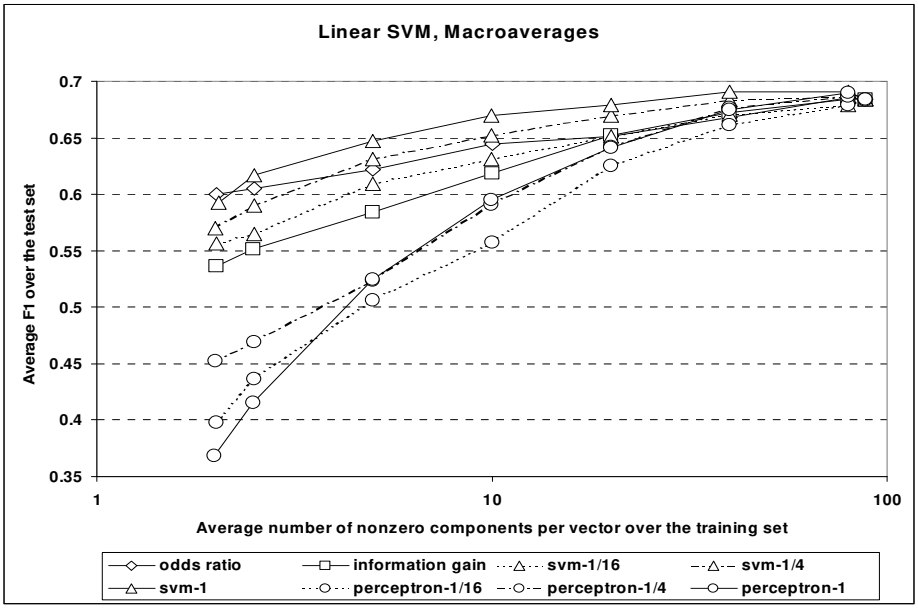

Figure 6: Macroaveraged F_1 performance of different feature selection methods in combination with SVM. The horizontal axes refer to sparsity, the average number of nonzero components per training vector. In this case, this is the average number of terms per training document after terms rejected by feature selection have been removed.

Looking at the precision and the recall separately shows that Perceptron-based feature rankings have particularly poor recall. From the sparsity curves, which provide insight in how the density of document vectors increases with the number of features retained, we conclude that the Perceptron-based feature weighting has a greater preference for rare features than the SVM-based weighting (although not nearly so great as Odds ratio), especially for features typical of positive documents. We speculate that this may be the cause of poor recall (consequently, poor F_1). Interestingly, Information gain tends to produce high precision but relatively low recall, whereas the opposite holds for Odds ratio. The SVM-based rankings, on the other hand, are good at both precision and recall, particularly the latter. Details are given in appendix D.

8 Conclusions

Our experiments show that feature scoring and selection based on the normal vector of the hyperplane obtained by the linear SVM algorithm combines very well with all the classifiers considered in the study. In particular, in combination with the Naïve Bayes classifier it seems to be far more effective than Odds ratio for all levels of sparsity below 40 features per vector. Even the Perceptron produces much better feature rankings for Naïve Bayes than Odds ratio. This supports our conjecture that the sophistication of the feature weighting method is more important for performance than its compatibility with the learning algorithm. On the basis of this observation our conjecture is that, on our dataset, the complexity and sophistication of the feature scoring algorithm plays a larger role in the success of the feature selection method than its compatibility in design with the classifier. Indeed, at the first glance it seems that Odds ratio, which closely follows the feature scoring used within Naïve Bayes, should provide the best selection for that classifier. However, a method such as SVM that tunes the normal by taking into account all the features simultaneously (rather than one at a time, as in the case of Odds ratio) seems to be the most successful in scoring features.

The SVM feature ranking combines relatively well with the Perceptron classifier considering the rather dramatic negative effect of feature selection on the Perceptron. One could argue that, because the *perceptron-1* is the best performing feature ranking with the Perceptron classifier, the conjecture we proposed in Section 1 is weakened. However, when we consider smaller subsets of the training data (e.g., $N/4$) and look for higher performance levels ($F_1 > 0.45$) of the Perceptron classifier we see that SVM-based and Perceptron-based feature selection have almost identical effect. This makes us believe that our conjecture is in the right direction.

In the experiments with the memory constraint, we have seen that the Perceptron classifier does not allow for a trade-off since it is adversely affected by feature selection while Naïve Bayes always benefits from feature selection. The SVM classifier, on the other hand, can benefit from combining feature selection and adjustment of the training set to fit the memory constraint. Namely, although the SVM classifier does not improve with feature selection itself, it yields better performance if feature selection allows a larger training set to be used to train the final model after feature selection.

Sparsity of the vectors representing the data has been found to be useful for comparing different feature selection methods, particularly because some of the learning algorithms are more sensitive to sparsity rather than to the number of features as such.

Finally, as has been found by other researchers in the field of text categorization, our experiments confirm that SVM outperforms Perceptron and Naive Bayes as a learning algorithm for text categorization. Our further work will expand this study to additional classifiers and data sets, including scenarios that do not necessary involve text data. We also conjecture that the selection of features obtained with the SVM-based feature ranking would be useful in other applications, not just in document classification, but also e.g. in clustering; investigating this conjecture through experiments is another possible direction for future work.

Acknowledgements

This work was supported by the Slovenian Research Agency, Microsoft Research and the IST Programme of the European Community under NeOn (IST-2006-027595), TAO (IST-2004-026460), PASCAL2 Network of Excellence (ICT-216886-NOE). This publication only reflects the authors' views.

References

[Bakus, 06] Bakus, J., Kamel, M. S. "Higher order feature selection for text classification". Knowledge and Information Systems, 9(4):468–91, April 2006.

[Brank, 02] Brank, J., Grobelnik, M., Milić-Frayling, N, Mladenic, D. "Feature selection using support vector machines". Proc. of the 3rd Int. Conf. on Data Mining Methods and Databases for Engineering, Finance, and Other Fields, Bologna, Italy, September 2002.

[Chakrabarti, 98] Chakrabarti, S., Dom, B., Agrawal, R., Raghavan, P. "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies". The VLDB Journal, 7, pp.163-178, Spinger-Verlag, 1998.

[Cortes, 95] Cortes, C., Vapnik, V.: "Support-vector networks". Machine Learning, 20(3):273–297, September 1995.

[Dumais, 98] Dumais, S., Platt, J., Heckerman, D., Sahami, M. "Inductive learning algorithms and representations for text categorization", Proceedings of 7th International Conference on Information and Knowledge Management, 1998

[Fung, 05] Fung, G. P. C., Yu, J. X., Lu, H., Yu, P. S. "Text classification without labeled negative documents". Proceedings of the 21st International Conference on Data Engineering (ICDE-2005), pp. 594–605.

[Forman, 03] Forman, G. "An Extensive Empirical Study of Feature Selection Metrics for Text Classification", Journal of Machine Learning Research 3, pp.1289–1305, 2003.

[Forman, 07] Forman, G. "Feature selection for text classification". In Liu, H., Motoda, H., (Eds.), "Computational Methods of Feature Selection", Chapman & Hall / CRC, 2007.

[Gabrilovich, 04] Gabrilovich, E., Markovitch, S. "Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5". Proceedings

of the 21st International Conference on Machine Learning (ICML 2004), Banff, Alberta, Canada, 2004.

[Gärtner, 01] Gärtner, T., Flach, P. A. "WBCSVM: Weighted Bayesian classification based on support vector machines". Proceedings of the 18th International Conference on Machine Learning (ICML 2001), Williamstown, Massachusetts, USA, pp. 154–161, 2001.

[Guyon, 02] Guyon, I., Weston, J., Barnhill, S., Vapnik, V. "Gene selection for cancer classification using support vector machines". Machine Learning, 46(1–3):389–422, 2002.

[Guyon, 03] Guyon, I., Elisseeff, A. "An Introduction to Variable and Feature Selection." Journal of Machine Learning Research 3, pp. 1157-1182, 2003.

[Joachims, 98] Joachims, T. "Text categorization with support vector machines: Learning with many relevant features". Proceedings of the 10th European Conference on Machine Learning (ECML-98), pp. 137–42.

[Joachims, 99] Joachims, T. "Making large-scale support vector machine learning practical". In B. Schölkopf et al. (Eds.), "Advances in kernel methods: Support vector learning". MIT Press, 1999, pp. 169–184.

[John, 94] John, G.H., Kohavi, R., Pfleger, K. "Irrelevant Features and the Subset Selection Problem". Proceedings of the 11th International Conference on Machine Learning (ICML94), pp. 121-129, 1994.

[Koller, 97] Koller, D., and Sahami, M. "Hierarchically classifying documents using very few words". Proceedings of the 14th International Conference on Machine Learning (ICML-97), pp. 170-178, 1997.

[Krauth, 87] Krauth, W., Mézard, M. "Learning algorithms with optimal stability in neural networks". Jour. Physics A 20, L745–L752, August 1987.

[Lewis, 92] Lewis, D.D. "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, June 21–24, 1992, Copenhagen, Denmark, pp. 37–50.

[McCallum, 98] McCallum, A., Nigam, K. "A comparison of event models for Naïve Bayes text categorization". AAAI Workshop on Learning for Text Categorization (pp. 41–48). AAAI Press, 1998.

[Milic-Frayling] Milic-Frayling, N., Mladenic, D., Brank, J., Grobelnik, M. "Sparsity analysis of term weighting schemes: Application to Feature Selection" (in review).

[Mitchell, 97] Mitchell, T.M., "Machine Learning". The McGraw-Hill Companies, Inc. 1997.

[Mladenic, 02] Mladenic, D. "Web browsing using machine learning on text data". In (P. S. Szczepaniak, ed.), "Intelligent exploration of the web", 111, Physica-Verlag, pp. 288–303, 2002.

[Mladenic, 04] Mladenic, D., Brank, J., Grobelnik, M., Milić-Frayling, N. "Feature Selection using Linear Classifier Weights: Interaction with Classification Models". Proceedings of the 27th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR-2004), Sheffield, UK, 2004, pp. 234–241.

[Mladenic, 99] Mladenic, D., Grobelnik, M. "Feature selection for unbalanced class distribution and Naïve Bayes". Proc. of the 16th Int. Conference on Machine Learning (ICML 1999). San Francisco: Morgan Kaufmann, pp. 258–267, 1999.

[Mladenic, 03] Mladenic, D., Grobelnik, M. "Feature selection on hierarchy of web documents". Journal of Decision support systems, 35, pp. 45-87, 2003.

[Mladenic, 06] Mladenic, D. "Feature Selection for Dimensionality Reduction". In Craig, S. et. al. (eds.), "Susbspace Latent Structure and Feature Selection techniques: Statistical and Optimisation perspectives", Springer Lecture Notes in Computer Science, vol. 3940, pp. 84-102, 2006.

[Olsson, 06] Olsson, J. S., Oard, D. W. "Combining feature selectors for text classification". Proceeding of the 15th ACM International Conference on Information and Knowledge Management (CIKM 2006), Arlington, VA, USA, pp. 798–799.

[Qiu, 08] Qiu, L.-Q., Zhao, R.-Y., Zhou, G., Yi, S.-W. "An extensive empirical study of feature selection for text categorization". Proceedings of the 7th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008), Portland, OR, USA, May 14–16, 2008.

[Quinlan, 93] Quinlan, J.R.. "Constructing decision trees". In: "C4.5: Programs for machine learning", pp. 17–26. Morgan Kaufmann, 1993.

[Rogati, 02] Rogati, M., Yang, Y. "High-performing feature selection for text classification". Proceedings of the 11th Conference on Information and Knowledge Management (CIKM-02), McLean, VA, USA, 2002, pp. 659–661.

[Rosenblatt, 88] Rosenblatt, F.. "The Perceptron: A probabilistic model for information storage and organization in the brain". Psych. Review 65(6), 386-408, 1958. Reprinted in: J. A. D. Anderson, E. Rosenfeld (Eds.), "Neurocomputing: foundations of research". Cambridge, MA: MIT Press, 1988.

[Salton, 88] Salton, G. and Buckley, C. "Term-weighting approaches in automatic text retrieval". Information Processing and Management, 24(5), pp.513–523, 1988.

[Schuetze, 95] Schuetze, H., Hull, D.A., Pedersen, O. J. "A comparison of classifiers and document representations for the routing problem". Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR-95), pp. 229-237, ACM Press, 1995

[Sindhwani, 01] Sindhwani, V., Bhattacharya, P., Rakshit, S. "Information theoretic feature crediting in multiclass Support Vector Machines". Proceedings of the 1st SIAM International Conference on Data Mining. SIAM, 2001.

[Shih, 02] Shih, L., Chang, Y., Rennie, J., Karger, D. "Not too hot, not too cold: The Bundled-SVM is just right!" Workshop on Text Learning (TextML-2002), ICML, Sydney, Australia, July 8, 2002.

[Yang, 97] Yang, Y., and Pedersen, J.O. "A Comparative Study on Feature Selection in Text Categorization". Proceedings of the 14th International Conference on Machine Learning (ICML-97), pp. 412-420, 1997.

[Yang, 99] Yang, Y., Liu, X. "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.

[Yu, 07] Yu, B., Unsworth, J. "An evaluation of text classification methods for literary study". Digital Humanities 2007, University of Illinois, Urbana-Champaign, IL, USA, June 2007.

Appendix A: About the RCV1 Corpus

The RCV1 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 stop words (from a list of 523 stop words, 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 stop words 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 super-category 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.

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

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

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 RCV1 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.

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 5.1 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 RCV1 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 RCV1 (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	*svm-1/16*	*svm-1/4*	*svm-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.

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 is 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.

Appendix D: Detailed experimental results of SVM

This appendix provides some details on experiments with SVM-based feature selection, as the best performing feature weighting measure and the linear SVM classifier, as the best performing algorithm. Figures D1, D2, D3, D4 contain graphs that show the SVM classification performance based on macro-averages of four measures: F_1, break-even-point (BEP), precision, and recall (notice that the same graph for F_1 as in Figure D2 is also given in the paper in Figure 6) . 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.)

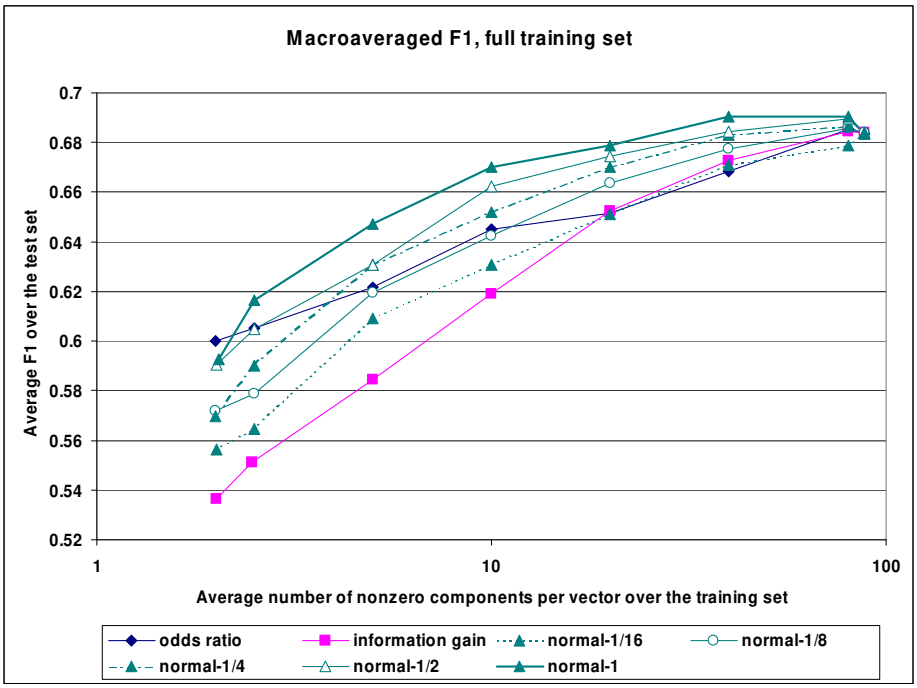

Figure D1. Macroaveraged F1 performance of different feature selection methods in combination with the SVM classifier. The horizontal axes refer to sparsity, the average number of nonzero components per training vector.

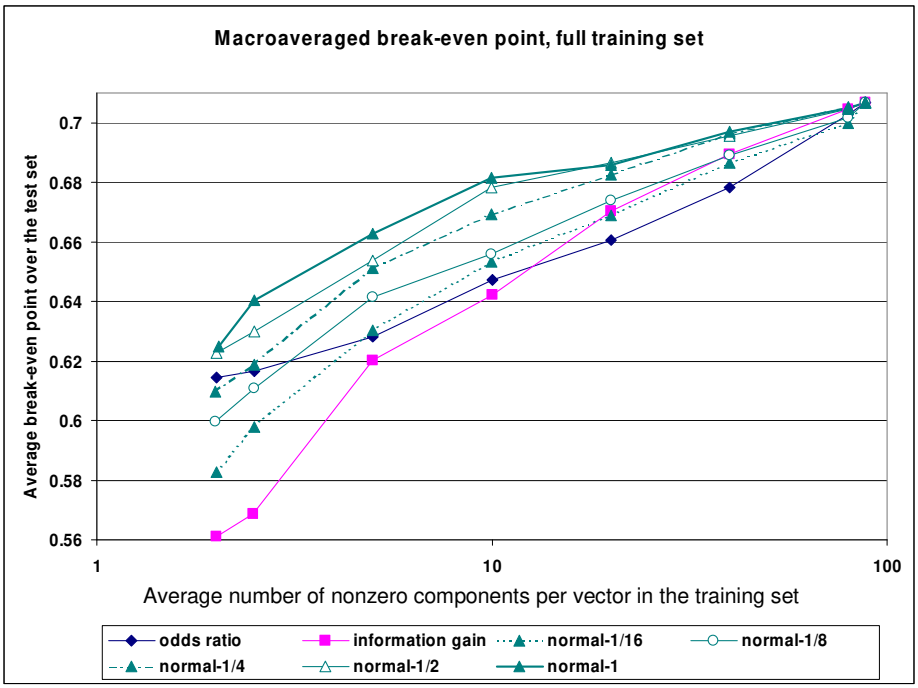

Figure D2. Macroaveraged break-even-point performance of different feature selection methods in combination with the SVM classifier. The horizontal axes refer to sparsity, the average number of nonzero components per training vector.

Although the above results show that feature selection generally does not improve performance of the linear SVM classifier, we can see that the SVM 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	0.119
Information gain	60,234	79.42 %	0.6843 ± 0.0047	0.315
Svm-$^1/_{16}$	20,316	26.79 %	0.6789 ± 0.0050	0.118
Svm-$^1/_8$	22,274	29.37 %	0.6859 ± 0.0048	0.099
Svm -$^1/_4$	24,338	32.09 %	0.6864 ± 0.0050	0.051
Svm- $^1/_2$	26,303	34.68 %	0.6894 ± 0.0050	**0.002**
Svm -1	28,638	37.76 %	0.6904 ± 0.0050	**0.003**
Full feature set	75,839	100.00 %	0.6837 ± 0.0049	

Table D1. 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 feature weighting 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 *svm-1/2* and *svm -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.

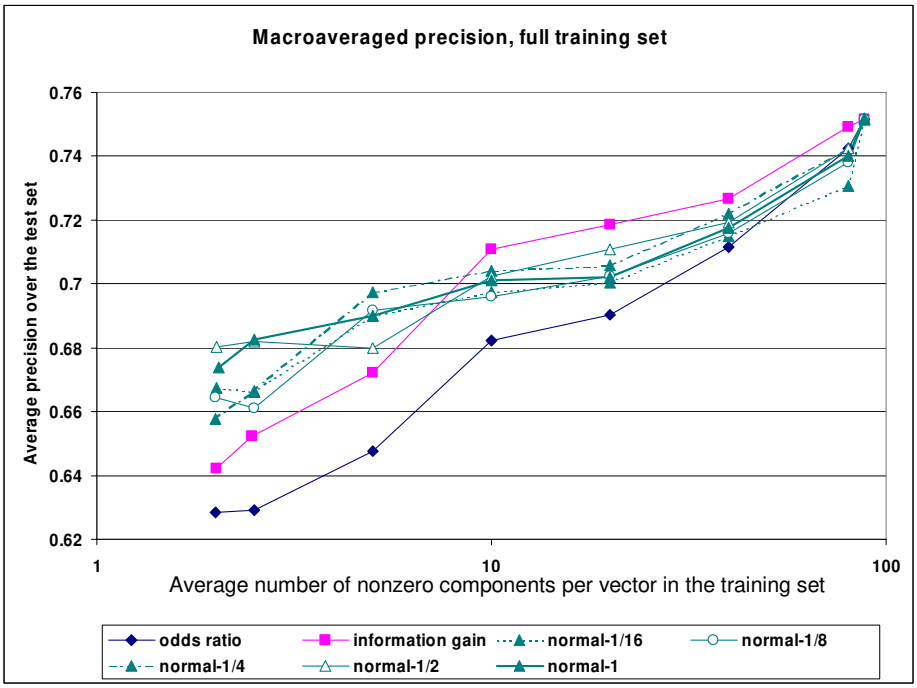

Figure D3. Macroaveraged precision performance of different feature selection methods in combination with the SVM classifier. The horizontal axes refer to sparsity, the average number of nonzero components per training vector.

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 D1 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 D2, which shows that *svm-1* at sparsity 40 actually still performs significantly better than full vectors, while the performance of *svm-1/2* and *svm-1/4* at sparsity 40, and that of *svm-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
svm-1/4	40	4278	5.65 %	0.6831 ± 0.0056	0.409
svm-1/2	40	4628	6.11 %	0.6842 ± 0.0042	0.436
svm-1	20	1545	2.04 %	0.6786 ± 0.0037	0.124
svm-1	40	5079	6.70 %	0.6905 ± 0.0045	0.030
Full feature set	88.3	75839	100.00 %	0.6837 ± 0.0049	

Table D2. Linear SVM performance with different feature selection methods at S = 40 and 20.

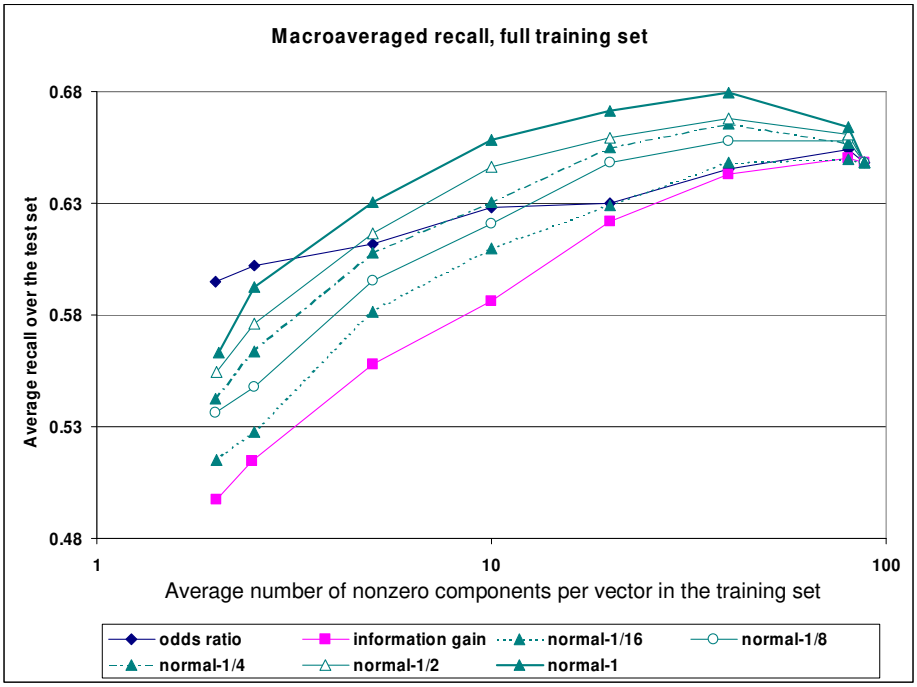

Figure D4. Macroaveraged recall performance of different feature selection methods in combination with the SVM classifier. The horizontal axes refer to sparsity, the average number of nonzero components per training vector.

D1. 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 D3, 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.

Category name	*Train-1600* (118,924 docs.) Size		F_1	BEP	Entire training period (504,468 docs.) 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 D3. Comparison of the performance on test data of the SVM classifiers trained on Train-1600 and those trained on the full training period. No feature selection was performed in either case.

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. Indeed, the breakeven point measure (also shown in Table D3), which involves calculating precision and recall of the document ranking based on scores that involve only the normal $\mathbf{w}$, does not suffer such an extreme decrease when the full training period is involved.

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).

D1. Combining feature weightings from several training data subsets

In comparing the sparsity behavior of the *svm-$^1/_{16}$* weighting, obtained from a smaller set of documents (*Train-$^1/_{16}$*), with that of *svm-1*, obtained from the full *Train-1600* set, we have seen that part of the difference stems from the inability of *svm-$^1/_{16}$* 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 *svm-$^1/_{16}$* has no good alternative but to assign them a weight of 0. This in turn affects the classification performance of models based on the resulting feature ranking.

Since one of the objectives 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 *svm-$^1/_{16}$*.

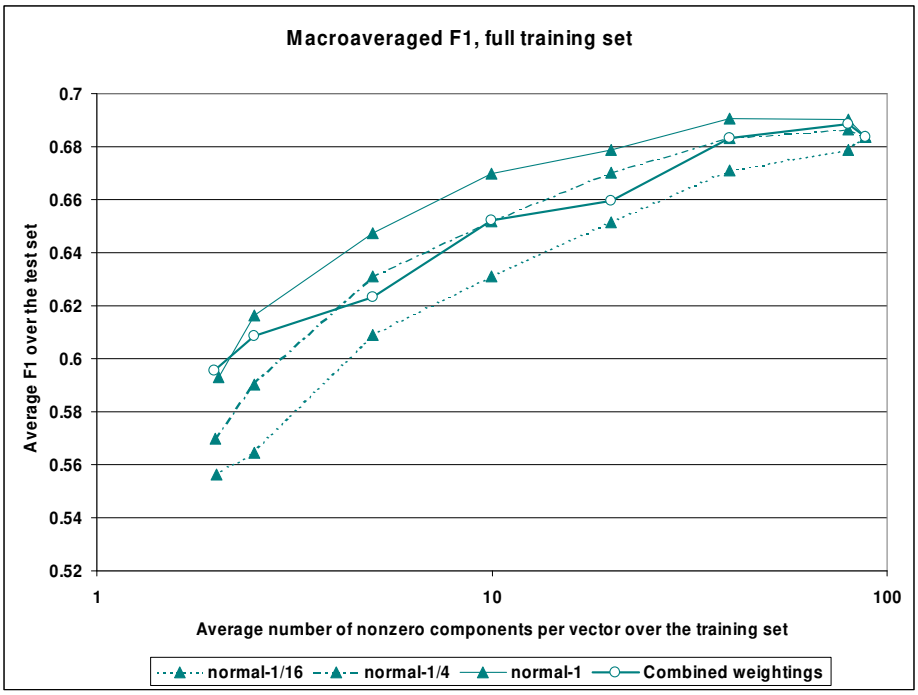

Figure D5. Macroaveraged precision performance of different feature selection methods in combination with the SVM classifier. The horizontal axes refer to sparsity, the average number of nonzero components per training vector.

The chart on Figure D5 shows a performance comparison of the combined feature weighting approach as described above and the *svm-*$^1/_{16}$, *svm-*$^1/_4$, and *svm-1* weightings. As can be seen from the chart, the performance of the combined weightings approach is roughly halfway between those for *svm-*$^1/_{16}$ and *svm-1*, and is indeed comparable to the performance of *svm-*$^1/_4$. 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 *svm-1* required 3,934 *s;* however, *svm-*$^1/_4$, 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 $O(N^{1.43})$.