Feature selection using support vector machines

J. Brank[1], M. Grobelnik[1], N. Milic-Frayling[2] & D. Mladenic[1]
[1] *Jožef Stefan Institute, Ljubljana, Slovenia*
[2] *Microsoft Research, Cambridge, UK*

Abstract

Text categorization is the task of classifying natural language documents into a set of predefined categories. Documents are typically represented by sparse vectors under the vector space model, where each word in the vocabulary is mapped to one coordinate axis and its occurrence in the document gives rise to one nonzero component in the vector representing that document. When training classifiers on large collections of documents, both the time and memory requirements connected with processing of these vectors may be prohibitive. This calls for using a feature selection method, not only to reduce the number of features but also to increase the sparsity of document vectors. We propose a feature selection method based on linear Support Vector Machines (SVMs). First, we train the linear SVM on a subset of training data and retain only those features that correspond to highly weighted components (in absolute value sense) of the normal to the resulting hyperplane that separates positive and negative examples. This reduced feature space is then used to train a classifier over a larger training set because more documents now fit into the same amount of memory. In our experiments we compare the effectiveness of the SVM-based feature selection with that of more traditional feature selection methods, such as odds ratio and information gain, in achieving the desired tradeoff between the vector sparsity and the classification performance. Experimental results indicate that, at the same level of vector sparsity, feature selection based on SVM normals yields better classification performance than odds ratio- or information gain-based feature selection when linear SVM classifiers are used.

1 Introduction

Trends towards personalizing information services and client-based applications

have increased the importance of effective and efficient document categorization techniques. It is that aspect of text classification that led us to explore methods for training classifiers that optimally use the computing memory and processing cycles for the available training data. In particular, we consider tradeoffs between the quality of document classification, as measured by commonly used performance measures, and reductions of a feature set used to represent the data.

In this study we propose a method for feature selection based on Support Vector Machines (SVM) [1] with linear kernels. We explore how this and other feature selection methods can be used to make tradeoffs between the amount of training data and the sparsity of the data representation when working with a limited amount of system memory.

Our experimental results on a large collection of Reuters documents [2] show that the SVM-based feature selection provides a suitable way of preserving the classification performance while significantly reducing the size of the feature space and increasing the sparsity of data.

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

2 SVM for Feature Selection

In situations where there is an abundance of training data it is conceivable that the training of classifiers cannot be performed over the full set of data due to limited computing resources. Indeed, methods like Support Vector Machines, for example, require practically the whole training data to be stored in the main memory all the time. Thus it can become necessary to work with smaller subsets of training data instead. The question then arises how the training of a given classifier can still be performed so that the full training set is taken into account. Here we present a simple procedure that has proven quite effective, as our experimental results show (see Section 5).

Our strategy is first to train linear Support Vector Machines (SVM) on a subset of training data to create initial classifiers. In this model each classifier is, in fact, a hyperplane separating 'positive' and 'negative' examples for the class (i.e., documents belonging to this class from those not belonging to it), and can be represented by the normal (i.e., a vector perpendicular to it) and a constant. (The size of the subset used in this step may depend on the available memory, or on how much time one is willing to spend for this step.) The second step involves eliminating features that have weights close to zero in this normal, in order to achieve a specified level of data sparsity. Sparsity is here defined as the average number of non-zero components in the vector representation of data or, in other words, the average number of terms left in documents after some terms have been discarded. Finally, using only features retained after the feature selection step, we create a representation of the full training set of documents.

We retrain the linear SVM classifier in the reduced feature space and use the final model to classify the test data.

This method is designed to take advantage of the memory freed as a result of increased data sparsity and allow one to work with larger training sets while keeping the memory consumption constant. We explored how SVM-based feature selection compares with other feature selection methods when used in conjunction with the proposed training strategy. When assessing feature selection methods we look at both the final classification performance of the resulting classifiers and the sparsity of training data representations. In the following section we give a brief overview of the considered feature selection methods and refer to research related to our work.

3 Related Work

3.1 Feature Selection Methods

In text categorization, feature selection (FS) is typically performed by assigning a score or a weight to each term and keeping some number of terms with the highest scores while discarding the rest. Experiments then typically evaluate the effect that feature selection has on the classification performance.

Numerous feature scoring measures have been proposed, e.g., information gain, odds ratio, χ^2, term strength, document frequency, etc. In our experiments we compared our SVM-based feature selection approach to two well-known feature selection measures that have been reported as successful in text categorization: information gain [3] and odds ratio [4, 5, 6].

3.1.1 Normal-based feature selection

In this study we use the Support Vector Machine with linear kernels. Training examples are described by vectors $\mathbf{x}_i = (x_{i1},\ldots,x_{id})$, where d represents the dimensionality of the feature space, i.e., the number of distinct features in the model. In general, the class predictor trained by SVM has the form $prediction(\mathbf{x}) = \mathrm{sgn}[b + \sum_i a_i K(\mathbf{x}, \mathbf{x}_i)]$ but in the case of a linear kernel $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$ this can be rewritten as $\mathrm{sgn}[b + \mathbf{w}^T\mathbf{x}]$ for $\mathbf{w} = \sum_i a_i \mathbf{x}_i$, where the vector of weights $\mathbf{w} = (w_1,\ldots,w_d)$ can be computed and accessed directly. Geometrically, the predictor uses a hyperplane to separate the positive from the negative instances, and $\mathbf{w}$ is the normal to this hyperplane.

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

theoretical justification for retaining the highest weighted features in the normal has been independently derived in a somewhat different context by Sindhwani et al. [7].

In this study we also use the linear SVM classifier as the classification model since it has been shown to outperform most of other classification methods on text data [8, 9]. Interaction of the normal-based FS with other classification methods will be subject of our future work.

4 Experiment Description

4.1 Experimental strategy – Rationale

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

Thus one set of our experiments explores the relationship between the sparsity of data representation achieved by feature reduction and the performance of the classifiers trained using such a reduced feature set. The second group of experiments investigates the robustness of the classification method under a limited memory constraint, exploring tradeoffs between discarding features and discarding training documents.

In both of these experiments we use linear SVM as the learning algorithm since it has been shown to outperform most of the other classification methods on text data [8, 9]. We particularly focus on the influence that the normal-based feature selection has on the linear SVM leaving the experiments with other methods for future work.

4.2 Data Processing and Analysis

The dataset. For experimentation we used the Reuters-2000 collection [2], which includes a total of 806,791 documents, with news stories covering the period from 20 Aug 1996 through 19 Aug 1997. We divided this time interval into a training period, which includes all the 504,468 documents dated 14 April 1997 or earlier, and the test period, consisting of the remaining 302,323 documents.

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

smaller categories are still well represented (indeed the percentage of documents belonging to most small categories is greater in Train-1600 than in the corpus at large). Note that the original category membership information is preserved. Thus a document that was selected as a representative of one category but also belongs to several other categories will be used as a positive training example for each of these categories. For the experimentation purposes we further selected sub-samples of the *Train-1600* set (of 118,924 documents) of size one half, referred to as *Train-1/2*, one quarter (*Train-1/4*), one eighth (*Train-1/8*), and one sixteenth (*Train-1/16*) of the full training set, respectively.

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

We used the SvmLight program (version 3.50) by Thorsten Joachims for SVM training and classification [10]. The program is particularly appropriate for our experiments because of its optimization for working with linear kernels. It computes the normal vector $\mathbf{w}$ explicitly rather than working entirely with the dual representation $\Sigma_i \alpha_i y_i \mathbf{x}_i$ as is necessary for other kernels.

Category selection. Training classifiers for all 103 Reuters categories over relatively large sets would be a time consuming and process intensive task. Therefore we restricted our study to a sample of 16 categories: *godd, c313, gpol, ghea, c15, e121, gobit, m14, m143, gspo, e132, e13, c183, e21, e142, and c13*. These categories were selected based on the results of a preliminary experiment that involved training the SVM over a smaller training set for the complete set of Reuters categories. In choosing the categories we tried to include categories of various size and difficulty.

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

Alternatively, for a given classifier one might compute, for each category, the average F_1 over the entire test set, thus obtaining a sequence of as many values as there are categories. One can then compare two methods by taking a paired t-test over the corresponding two sequences. This approach (referred to as a "macro T-test" by Yang and Liu [11]) has been often used, although it has also been criticized because it treats performance values on different categories as random samples from a normal distribution [12].

In our experiments all types of tests tend to give similar results. However, the category-paired t-test is slightly more conservative (i.e., less likely to declare differences to be significant) than the t-test on macroaveraged F_1-values, which, on the other hand, is more conservative than the t-test on microaveraged F_1-values. For the remainder of this paper, we will report significance results from the t-tests based on macroaveraged F_1 values.

5 Experimental Results

5.1 Sparsity and the number of features

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

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

Figure 1 shows a comparison of the levels of sparsity achieved by the odds ratio (OR), information gain (IG), and normal-based feature selection for the positive and negative documents in our training data, respectively. These and the following figures show graphs that correspond to several variants of the normal-based feature weighting, depending on how large a set of documents was used to train the normal. If the full *Train-1600* training set is used we call the resulting feature weighting *normal-1;* if *Train-*$^{1}/_{k}$ is used, the resulting weighting is denoted by *normal-*$^{1}/_{k}$.

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

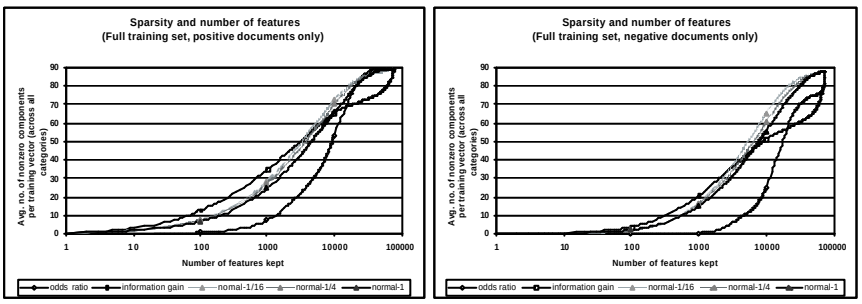

Figure 1. The graphs show how the average number of terms per document grows as increasingly many of the best scoring terms, according to various feature weighting methods, are included into the selected feature set.

These curves confirm that IG and OR behave as expected from their definitions. Information gain ranks highly those relatively common features whose presence or absence correlates well with the category membership. It ranks low the features (including many common ones) that are equally common to both members and non-members of a category. Odds ratio, as expected, shows a great preference for features typical of positive documents, even if they are very rare, and a great dislike for features typical of negative documents.

The charts also show the behavior of the normal-based feature weightings, which cannot be easily inferred from their definition. It can be seen that these weightings are less reluctant than information gain to assign high scores to less common features but they are not nearly as favorable to rare features as the odds ratio. In addition, normals trained on larger subsets of the training set are more favorable to rare features than those trained on smaller subsets. A closer look at the selected feature sets reveals that this is partly because some of the features that are rare in the whole training corpus have not been present in the smaller training subsets at all and partly because some were present there but were too rare to be ranked highly.

5.2 Dimensionality reduction and classification performance

It is also interesting to relate the reduction in dimensionality to the reduction in classification performance. From Figure 2 we see that OR needs to retain one or two orders of magnitude more features to achieve a performance that is comparable to the other FS methods when used with the linear SVM classifier. This is not surprising in view of the observations made in the previous section. Indeed, OR scores highly features that are characteristic of positive documents, including those that are rather rare and thus do not necessarily make an impact on the data representation and the SVM training process.

In many practical situations the management of computing resources is important and thus the sparsity seems to be of high practical importance. On the other hand, the number of features itself is not so important when SVM is used for training, because the dual optimization problem on which SVM is based does not explicitly mention the number of features. In the remainder of the study we will focus on sparsity and use dimensionality reduction to meet the sparsity requirements.

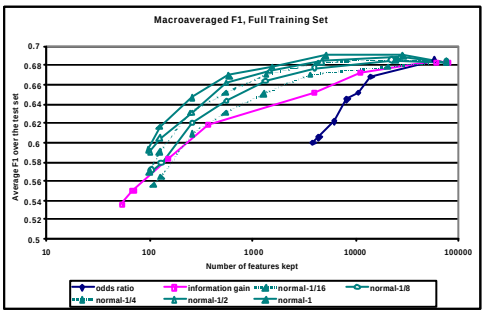

Figure 2. This chart shows the macroaveraged F_1 values achieved by various feature rankings depending on how many features are retained before the final training phase.

5.3 Experiments with the fixed memory constraint

Part of the motivation for using feature selection is to describe documents with sparser vectors and thus process larger sets of documents with a given amount of memory. However, we need to establish evidence that such an approach is

plausible and that it is possible to achieve an effective tradeoff between selecting a subset of features (to make vectors sparser) and selecting a subset of documents (to have fewer vectors). For example, if we have only a quarter of the amount of memory required for storing full vectors of the complete set of training data we could: (*i*) keep full vectors but only for 1/4 of the training documents; or (*ii*) keep 1/2 of the documents, and use feature selection to make the vectors twice as sparse; or (*iii*) keep all the documents, and use feature selection to make the vectors four times sparser; etc.

In order to explore these possibilities we evaluate the classification performance for the following type of scenarios. Let us assume that storing all N documents from *Train-1600* at S=80 (in fact full documents have approx. 88 terms on average) would require M units of memory ($S \cdot N = M$), and that we only have $M/2^K$ units available for some integer K. We are seeking the values S' (sparsity) and N' (number of documents) such that $S' \cdot N' = M/2^K$ and for which the classification performance is not significantly degraded. For simplicity we experiment with values $S' = S/2^k$ and $N' = N/2^{K-k}$, for $k = 0, \ldots, K$. Note that ideally we would vary the values of S and N on a finer scale but the current approach is already quite informative as the experiment results show (see Figure 2).

For a given K, we first obtain a ranking of features to be used as the basis for feature selection. Since the SVM model on which this ranking will be based needs to be trained over full vectors, it can use at most $N/2^K$ documents to respect the memory constraint. Then, for a given higher sparsity level $S/2^k$, $k = 0, \ldots, K$, we select as many top features from the ranking as are necessary to achieve that sparsity, and then train a new model over the largest training set that can fit into the available memory given the fact that vectors are now sparser (this is the set containing $N/2^{K-k}$ documents). Thus, for a given K, the result at sparsity 80 is based on a training set that is only half as large as the set used for the experiment with sparsity 40, only one-quarter the size of the training set used for sparsity 20, and so on.

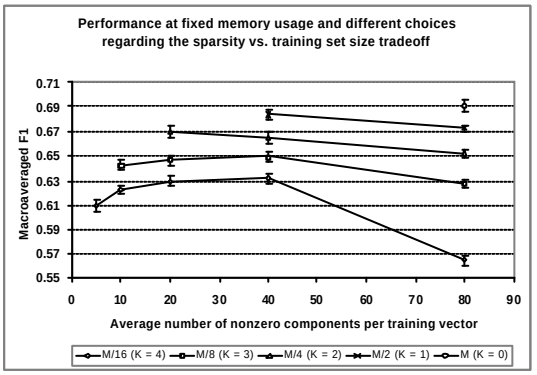

Figure 3. A comparison of different options of combining feature selection and discarding training documents if the amount of memory available for training is limited. The error bars on the graph show standard errors of F_1 macro-averages over the 10 subsets of the test set.

The results confirm that for a fixed memory constraint (fixed K) one can achieve a better performance by applying feature selection and using a larger set of training documents than by using the full set of features but a smaller training

set. For example, it is beneficial to reduce the number of nonzero components by 50% and double the number of training documents if one is working under a memory constraint (this always brought a statistically significant improvement in performance in the above experiments). That is, in some sense, expected, based on the evidence from related research that reducing the feature space typically does not dramatically affects the classification performance.

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

5.4 Experiments at fixed sparsity and training set

In these experiments we use *Train-1600* as the training set and different feature scoring methods to achieve a desired level of sparsity, i.e., a desired average number of nonzero components per training vector. In particular we look at sparsity of 2, 2.5, 5, 10, 20, 40, and 80 features per document as well as the full document vectors which have about 88 nonzero components per document on average.

For feature scoring methods we use Odds Ratio (OR), Information Gain (IG), and ranking based on SVM normals: *normal-1*, *normal-*$^1/_2$, *normal-*$^1/_4$, *normal-*$^1/_8$, and *normal-*$^1/_{16}$. The methods *normal-*$^1/_k$ use SVM normals obtained from the set *Train-*$^1/_k$, containing $^1/_k$ as many documents as the full training set (*Train-1600*); after feature selection, the final linear SVM model is trained on the full training set but with the reduced set of features. This allows us to see how robust SVM classifier is with respect to limited feature sets obtained from smaller amounts of training data. Results of these experiments are shown on Figure 4.

Comparison of the normal-1 ranking with odds ratio and information gain. From the perspective of F_1 and BEP measures, SVM-based ranking of features is most effective for the purpose of feature selection. The performance is dominated by this method for all levels of sparsity, except when only a couple of features per document are used, in which case OR seems to perform better. This is not surprising since OR retains a much larger set of features than the normal to achieve the same sparsity level (see Figure 1). Furthermore, it focuses on features that are characteristic of positive examples, which in turn enables SVM to capture the positive examples more easily. However, for the same reason, the precision of classification based on OR selection suffers, as can be seen from the precision graph. Information gain, on the other hand, ensures a consistently better precision than *normal-1* for all sparsity levels $S>5$ (these differences are statistically significant at $S=20$ and 80).

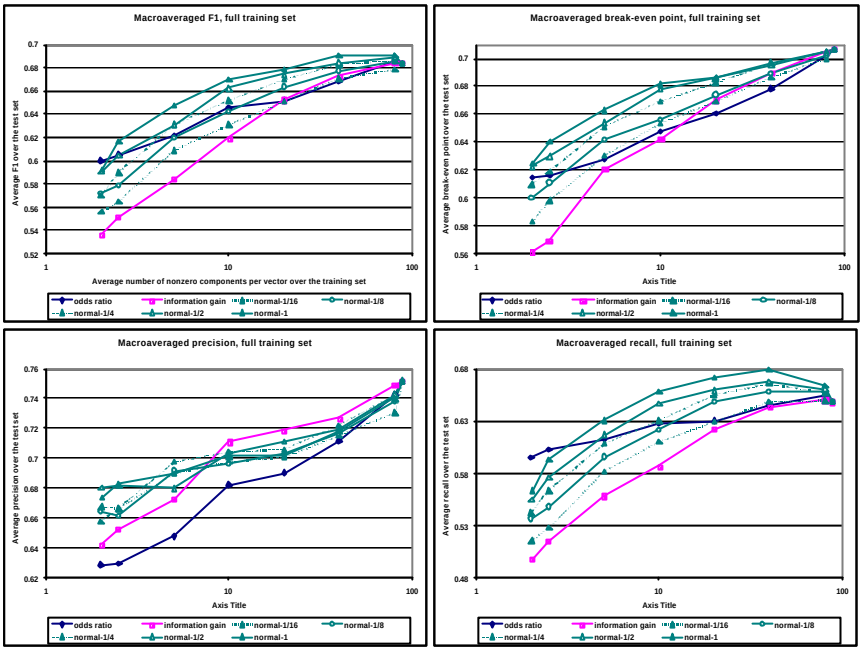

Figure 4. A comparison of several feature selection methods at different levels of sparsity. The full *Train-1600* training set was used in all cases. The test set was divided into 10 "folds", macroaveraged performance measures were computed for each fold and the averages over all 10 folds are given here.

Comparison of normal-$^1/_k$, $k=1,2,4,8,16$. SVM normals obtained from larger data sets have a clear advantage on all performance measures except for precision. Indeed they all seem to perform similarly according to that measure. Thus, if one is concerned with precision only one need not hesitate to fix the feature set to the one obtained from a smaller set of data and then retrain. Using more data in the initial phase introduces features that definitely help recall but do not affect precision. Normal-based selection seems to pick the features important for precision even from smaller sets of data.

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

5.5 Effectiveness

Although the above results show that feature selection generally does not improve performance of the linear SVM classifier, we can see that the normal-

based feature selection with sparsity S=80 (which means keeping from 60 to 75% of features) does produce a statistically significant improvement over the performance of vectors with the complete feature set. In addition, S=40 with *normal-*$^1/_4$ (keeping 5.6% of features) and *normal-*$^1/_2$, and S=20 with *normal-1* (keeping 2.0% of features) achieves performance that is not significantly worse than that of the full feature set.

6 Summary and future work

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

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

Furthermore, other linear classifiers could be similarly used to weight and select features, including Perceptron [13], Winnow [14], Bayes Point Machine [15], LLSF [16], Widrow-Hoff [17], exponentiated gradient [18, 17], and so on. (Naive Bayes is also essentially a linear classifier if we work with logarithms of probabilities. This has been exploited by e.g. Gärtner and Flach [19].) While these methods are known to be more or less successful in classifying documents, it would be interesting to see how they compare with the SVM-based feature selection method in reducing the feature space.

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

References

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

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

[3] Y. Yang, J. O. Pedersen: *A comparative study on feature selection in text categorization*. Proc. 14th ICML Conf., pp. 412–420, 1997.

[4] D. Mladenic, M. Grobelnik: *Feature selection for unbalanced class distribution and Naive Bayes*. Proc. 15th ICML Conf., pp. 258–267, 1999.

[5] D. Mladenic: *Feature subset selection in text-learning*. Proc. 10th ECML Conf. LNCS vol. 1398, pp. 95–100, 1998.

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

[7] V. Sindhwani, P. Bhattacharyya, Subrata Rakshit: *Information theoretic feature crediting in multiclass support vector machines.* First SIAM Int. Conf. on Data Mining, 2001.
[8] S. Dumais, J. Platt, D. Heckerman, M. Sahami: *Inductive learning algorithms and representations for text categorization*. Proc. 7th Int. Conf. on Information and Knowledge Management, pp. 148–155, 1998.
[9] T. Joachims: *Text categorization with support vector machines: learning with many relevant features*. Proc. 10th ECML. LNCS vol. 1398, pp. 137–142, 1998.
[10] T. Joachims: *Making large-scale support vector machine learning practical*. In: B. Schölkopf, C. J. C. Burges, A. J. Smola (Eds.): *Advances in kernel methods: Support vector learning*, MIT Press, 1999, pp. 169–184.
[11] Y. Yang, X. Liu: *A re-examination of text categorization methods.* Proc. of the 22nd ACM SIGIR Conf., pp. 42–49, 1999.
[12] D. D. Lewis: *An evaluation of phrasal and clustered representations on a text categorization task*. Proc. 15th ACM SIGIR Conf., pp. 37–50, 1992.
[13] F. Rosenblatt: *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review 65(6), pp. 386-408. Reprinted in: J. A. D. Anderson, E. Rosenfeld (Eds.), *Neurocomputing: Foundations of Research*, MIT Press, 1998, pp. 89–114.
[14] N. Littlestone: *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm.* Machine Learning, 2(4), pp. 285–318, 1988.
[15] R. Herbrich, T. Graepel, C. Campbell: *Bayes point machines*. Journal of Machine Learning Research, 1(Aug), pp. 245–279, August 2001.
[16] Y. Yang, C. Chute: *A linear least squares fit method for terminology mapping*. Proceedings of the 15th Int. Conf. on Computational Linguistics (COLING 1992), II:447–53, 1992.
[17] D. D. Lewis, R. E. Schapire, J. P. Callan, R. Papka: *Training algorithms for linear text classifiers*. Proc. 19th ACM SIGIR Conf., pp. 298–306, 1996.
[18] J. Kivinen, M. K. Warmuth: *Exponentiated gradient versus gradient descent for linear predictors*. Tech. Report UCSC-CRL-94-16, Baskin Center for Computer Engineering & Information Sciences, University of California, USA, June 21, 1994 (revised December 7, 1995).
[19] T. Gärtner, P. A. Flach: *WBC_{SVM}: Weighted Bayesian classification based on support vector machines*. Proc. 18th ICML Conf., pp. 154–161, 2001.